Automating Off-Site Construction: Integrating BIM and Industrial Robotics through the RoboSimX Framework for Smart Manufacturing

by

Anas Itani

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Civil (Cross-disciplinary)

Department of Civil and Environmental Engineering
University of Alberta

ABSTRACT

Off-site construction (OSC) has been slow to adopt automation and digitization, particularly since OSC requires manufacturing flexibility due to a need for diverse product designs in dynamic environments. While automation is crucial for OSC operations, integrating digital technologies with design and manufacturing processes is becoming increasingly complex. This complexity is deepened by rapid technological advancements and data integration challenges, which hinder effective implementation. Moreover, the shift toward industrialized construction requires advanced fabrication lines, flexible algorithms, and enhanced methods for assembling various product variants and streamlining fabrication processes. Building Information Modelling (BIM) has become prominent in digitizing the design and automating construction workflows. Similarly, Industrial Robotics (IR) in construction is gaining recognition for enhancing productivity, safety, and efficiency while minimizing lean waste. However, the integration of BIM and IR within OSC remains underexplored. This research introduces the RoboSimX framework, which incorporates a flexible Multi-Robotic Operational System (ROS) to automate and simulate processes. The project starts with a Systematic Literature Review (SLR) to evaluate the integration of BIM-IR, followed by the development of a Robotic Assembly Task Motion Planner (RATMP) for intelligent and zero-labour robotic manufacturing cells. A Virtual Commissioning (VC) system is employed to simulate and optimize manufacturing strategies. Additionally, the framework utilizes an XAI engine, which provides interpretable, model-agnostic explanations to integrate and analyze data from BIM-IR activities, thus creating an automated decision-making tool. This tool aims to enhance manufacturing productivity, design flexibility, environmental adaptability, tool interchangeability, and overall system effectiveness. Using the Design Science Research (DSR) methodology, the study demonstrates that BIM-IR integration is achieved in OSC through the RoboSimX framework. Validated across panelized and modular sectors in the wood and steel

industries, the process includes software simulation, laboratory experiments, and expert surveys. This research offers the industry a comprehensive, adaptable BIM-IR-based solution for OSC manufacturing automation.

PREFACE

This thesis represents the original work of Anas Itani. It includes three journal papers and three conference papers that have been prepared for publication. These papers are sequentially listed below, in the order they appear in this thesis. The thesis is structured in a paper format, adhering to the paper-based thesis guidelines.

(1) **Itani, A**., Bouferguene, A., and Al-Hussein, M. (2024). Innovative Approaches in Assembly Sequence Planning: A Review of Soft Computing and AI Methodologies. **(Ready for Submission– Conference Proceedings)**

(2) **Itani, A**., Bouferguene, A., and Al-Hussein, M. (2024). Integrated BIM-Robotic Assembly Sequence Planning Tool: Bridging the Gap from CAD to CAM in Offsite Construction Manufacturing. **(Ready for Submission – Journal Publication)**

(3) **Itani, A**., Bouferguene, A., and Al-Hussein, M. (2024). Advances in Multi-Robot Task Allocation: From Manual Coordination to Algorithmic Optimization. **(Ready for Submission – Conference Proceedings)**

(4) **Itani, A**., Bouferguene, A., and Al-Hussein, M. (2024). From Heuristics to Learning-Based Methods: Evolving 3D Motion Planning for Robotics. **(Ready for Submission – Conference Proceedings)**

(5) **Itani, A**., Bouferguene, A., and Al-Hussein, M. (2024). Enhanced Simulated Motion Planning Framework for Robotic Assembly in Offsite Construction **(Ready for Submission – Journal Publication)**

(6) **Itani, A.**, Bouferguene, A., and Al-Hussein, M. (2024). XAI-LIME and BIM-IR: A Synergistic Framework for Smart Manufacturing Decision-Making and System Effectiveness. **(Ready for Submission – Journal Publication)**

ACKNOWLEDGMENT

First and foremost, I am deeply grateful to God for providing me with the strength, intellect, well-being, and supportive individuals necessary for the successful completion of my research.

I would like to extend my profound appreciation to my supervisor, Dr. Mohamed Al-Hussein, and my co-supervisor, Ahmed Bouferguene, for their unwavering guidance, support, motivation, and influence throughout this research. Their involvement has been indispensable, and I cannot envision having completed this academic journey without their invaluable contributions.

I also appreciate the backing of the administrative and research staff within Dr. Al-Hussein's group at the University of Alberta. Their technical expertise and emotional encouragement have been crucial. Special thanks go to the administrative staff and research group members for their unwavering support and constructive feedback during my studies. Additionally, I acknowledge the financial support provided by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Lastly, I extend my deepest gratitude to my wonderful family, who has been an unwavering pillar of support over the years. A special acknowledgment goes to my wife, Inaam, for her significant contribution, marked by her unconditional love, companionship, and steadfast belief in me, which greatly aided in reaching this milestone in my life. Also, to my Dad, Mom, Mia, Ahmad, Sarah, and Jad, your continuous encouragement throughout this journey has been pivotal. This achievement would not have been possible without your enduring support and boundless love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D/3D	Two- Three Dimensional
ACO	Ant Colony Optimization
ADC	Assembly Direction Change
AEC	Architecture, Engineering, and Construction
AE	Assembly Energy
AIS	Artificial Immune System
ALB	Assembly Line Balancing
ALO	Ant Lion Optimizer
AOT	Assembly Operation Type
API	Application Programming Interface
APF	Artificial Potential Field
ASP	Assembly Sequence Planning
ASG	Assembly Sequence Generation
ATC	Assembly Tool Change
ATSP	Asymmetric Travelling Salesman Problem
B-rep	Boundary Representation
BIM	Building Information Modelling
BIP	Binary Integer Programming
BLSA	Breakout Local Search Algorithm
Bnb	Branch and Bound
BCM	Building Component Manufacturing
BC	Bacterial Chemotaxis
BCO	Bee Colony Optimization
CAD	Computer-Aided Design

CAM	Computer-Aided Manufacturing
CAS	Constraint Assembly State
CBBA	Consensus-Based Bundle Algorithm
CD	Constraint Direction
CETSP	Close-Enough Travelling Salesman Problem
CHOMP	Covariant Hamiltonian Optimisation for Motion Planning
CNC	Computer Numerical Control
CNN	Convolutional Neural Networks
COG	Center of Gravity
CPSO	Chaotic Particle Swarm Optimization
CSV	Comma-Separated Values
CV	Coordination View
CVAE	Conditional Variational AutoEncoder
DC	Directional Changes
DeepSMP	Deep Sampling-based Motion Planner
DFA	Design for Assembly
DFMA	Design for Manufacturing and Assembly
D-H	Denavit-Hartenberg
DOF	Degrees of Freedom
DPSO	Discrete Particle Swarm Optimization
DSR	Design Science Research
EDO	Evolutionary Direction Operator
EA	Evolutionary Algorithms
ERP	Enterprise Resource Planning
EMA	Embedded Multifusion Algorithms

FE	Fact Extraction
FEDS	Framework for Evaluation in Design Science
FK	Forward Kinematics
FLA	Frog Leaping Algorithm
FL	Fitness Function
FN	False Negative
FOL	First Order Logic
FP	False Positive
FTr.	Fact Transformation
GA	Genetic Algorithm
GAN	Generative Adversarial Network
GNN	Gaussian Mixture Models
GSA	Gravitational Search Algorithm
GTSP	Generalized Travelling Salesman Problem
GTSPN	Generalized Travelling Salesman Problem with Neighborhoods
GWO	Grey Wolf Optimizer
HA	Hybrid Algorithm
HA	Hungarian Algorithm
HC	Horn Clause
HJB	Hamilton-Jacobi-Bellman
ICA	Imperialist Competitive Algorithm
IFC	Industry Foundation Classes
IK	Inverse Kinematics
ILP	Integer Linear Programming
IOA	Immune Optimization Approach

IR	Industrial Robotics
ISO	International Organization for Standardization
JSDAI	Java Standard Data Access Interface
KNN	K-Nearest Neighbour
KP	Kinodynamic Planning
K-PRM	K Neighbours Probabilistic Road Map
LAP	Linear Assignment Problem
LAP*	Lifelong Planning A*
LiDAR	Light Detection and Ranging
LbD	Learning by demonstration
LOD	Level of Development/Detail
LSTM	Long Short-Term Memory
MCAS	Minimal Constraint Assembly State
MDP	Markov Decision Process
MINP	Mixed-Integer Nonlinear Program
MRS	Multi-Robot Systems
MRTA	Multi-Robot Task Allocation
MTPGR	Multi-Target Pathfinding for Goal Regions
MVD	Model View Definition
NLP	Natural Language Processing
NN	Neural Network
NP	Nondeterministic Polynomial
OSC	Off-site Construction
OSB	Oriented Strand Board
OWL	Web Ontology Language

PM	Precedence Matrix
PMX	Partially Matched Crossover
PSO	Particle Swarm Optimization
PTP	Point-to-Point
RAP	Robotic Assembly Planning
RATMP	Robotic Assembly Task Motion Planner
RBF	Rule-Based Fact
RE	Regular Expression
RL	Reinforcement Learning
RMA	Ranked Multifusion Algorithms
RMP	Robotic Motion Planning
RNN	Recurrent Neural Network
ROD	Robot Oriented Design
ROS	Robot Operating System
RPA	Robotic Process Automation
RPS	Robotic Prefabrication System
RRT	Rapidly Exploring Random Tree
RRT*	Rapidly Exploring Random Tree Star
RRG	Rapidly-Exploring Random Graph
RTP	Task Allocation Planning
SA	Simulated Annealing
SBA	Sampling-Based Algorithms
SBMP	Sampling-Based Motion Planning
SLR	Systematic Literature Review
SOL	Second Order Logic

SOP	Sequential Ordering Problem
SRP	Safari Route Problem
STEP	Standard for Exchange of Product
STOMP	Stochastic Trajectory Optimisation for Motion Planning
SVM	Support Vector Machines
SWRL	Semantic Web Rule Language
TCP	Tool Center Point
TC	Tool Changes
TN	True Negative
TMP	Task and Motion Planning
TP	True Positive
TSP	Travelling Salesman Problem
TSPN	Travelling Salesman Problem with Neighbourhoods
UI	User Interface
UR	Universal Robots
URDF	Unified Robotic Description Format
VDC	Virtual Design and Construction
VC	Virtual Commissioning
ZRP	Zookeeper Route Problem

Chapter 1 : Introduction

1.1 Initial Research Context

In 2018, Canada records a hike in the residential construction sector where 215,725 housing starts were recorded (Statistics Canada). It is expected that the residential construction for housing stays close to 200,000 units in 2019 (Housing Market Outlook, Fall 2018). On the other hand, commercial construction is expected in 2018 to grow by 6%; this is due to the Digital technologies are reshaping the Architecture, Engineering, and Construction (AEC) industry, significantly altering its digital landscape, as highlighted by Alwisy et al. (2019). However, the adoption of digital tools remains inconsistent between design and construction phases. The industry still relies heavily on manual labor and traditional equipment, particularly in North America, where labor shortages exacerbate productivity issues (Poirier et al., 2018; García de Soto et al., 2022). This reliance on outdated methods underscores the pressing need to bridge the digital divide between design and construction to improve efficiency and promote industrialized construction techniques.

In the digital era, data collection has evolved from a complex task to one enhanced by modern design tools (Carpo, 2017). Building Information Modeling (BIM) significantly improves upon traditional Computer-Aided Design (CAD) by integrating comprehensive information functions that support informed modeling and effective data management (Shepherd, 2019). BIM also boosts collaboration by centralizing information accessible to all project stakeholders, addressing prevalent inefficiencies and suboptimal outcomes in the AEC industry (Race, 2019). Despite its capabilities, BIM's full potential in digital production, especially in integration with Industrial Robotics (IR) to optimize construction processes, remains underutilized (Yin et al., 2019).

IR has been prominent in manufacturing for over fifty years and is increasingly used as interest in automated production grows across industries (Gurgul, 2018; Dachs et al., 2019). Yet, IR adoption

in the AEC industry is limited due to incompatibilities between Computer-Aided Manufacturing (CAM) tools, designed for robotic assembly sequence and motion control, and the variable, reconfigurable nature of construction tasks. Many construction robots use proprietary programming languages that struggle with rapid design changes, presenting barriers for construction professionals (Garcia del Castillo Lopez, 2019). Additionally, the integration of robotic programming tools with BIM software is still in development (Davtalab et al., 2018). These tools often operate in isolation, creating a gap between design and robotic programming. This disconnection between BIM and IR tools underscores an urgent need for research focused on integrating these technologies.

Integration, according to ISO/IEC 2382 Information Technology Vocabulary, involves diverse functional units' capacity to communicate, execute programs, or transfer data, requiring minimal user knowledge of the units' specific characteristics (ISO, 2022). Originally a technological term, integration now includes multiple dimensions tailored to the systems in question. This study focuses on three key integration tools that connect design and manufacturing, as illustrated in Figure 1.

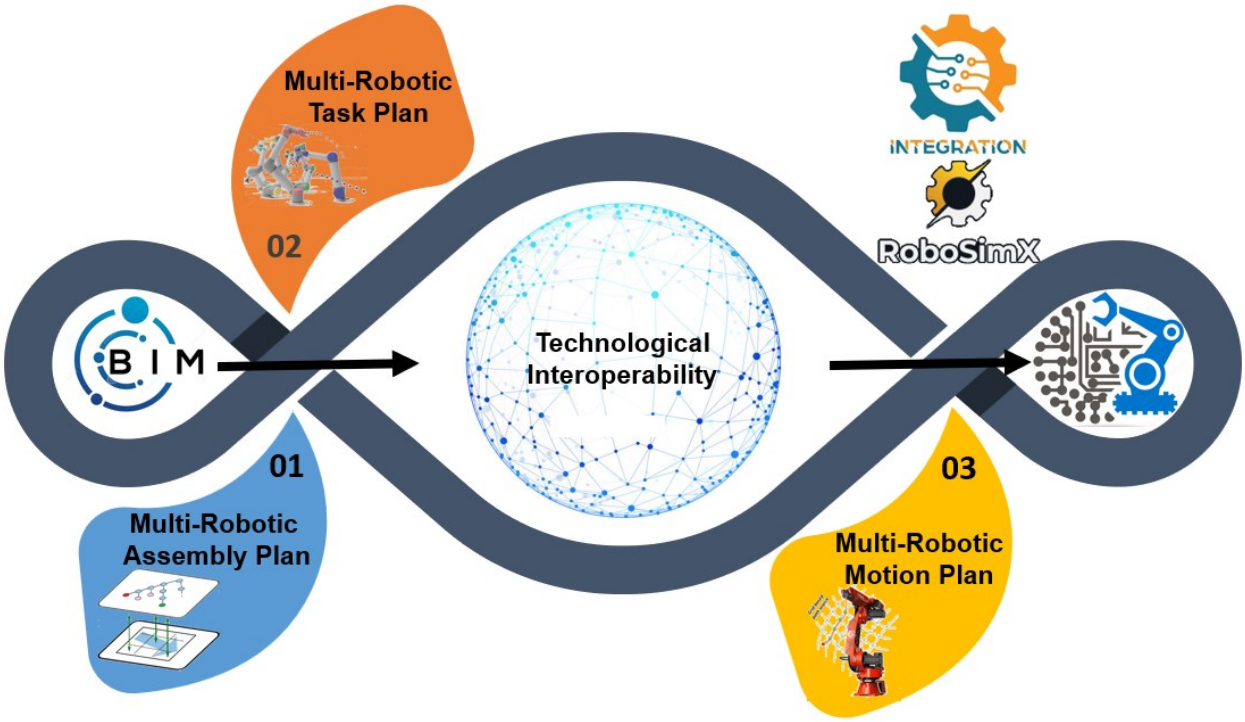


Figure 1. Starting Context of the Research

1.2 Robotic Assembly Task Motion Planner (RATMP)

The integration of Industrial Robotics (IR) arms into prefabrication processes is increasingly popular, driven by decreasing costs and improved capabilities. These robots are equipped with advanced controls and high degrees of freedom (DOF) manipulators, making them ideal for executing repetitive and precise tasks like component assembly autonomously, which accommodates various building designs. However, a significant challenge with IR arms in prefabrication is devising viable assembly plans and ensuring collision-free paths for building components. Assembly sequence planning is critical as it influences tool change frequency, assembly directions, and the design of fixtures, all of which affect process efficiency. Choosing the correct assembly sequence is complex due to the exponentially increasing possibilities with the addition of more parts. In many industrial contexts, assembly planning still often depends on engineer expertise rather than systematic analysis, presenting significant optimization opportunities.

The proposed Robotic Assembly Task Motion Planner (RATMP) addresses these challenges by facilitating both discrete and continuous decision-making processes. Assembly Sequence Planning (ASP) involves discrete decisions, such as selecting objects and ordering actions, while Robotic Motion Planning (RMP) deals with continuous decisions like plotting collision-free trajectories for executing actions. This separation sometimes leads to feasibility issues, where a chosen action may lack a viable motion path for execution (Dantam, 2020).

The RATMP framework integrates assembly, task, and motion planning in a multi-robotic setting, linking high-level assembly planning with mid-level task allocation and low-level motion planning. This comprehensive integration ensures executable steps that transition objects from initial to final states, optimizing planning, allocation, and execution phases in robotic prefabrication.

Erdem et al. (2016) outline three primary integration styles between assembly, task, and motion planning, as illustrated in Figure 2:

(a) **Pre-assembly Motion Planning**: The motion planner evaluates the feasibility of potential assembly actions before starting assembly planning, informing the subsequent planning phases with pre-evaluated data.

(b) **Concurrent Motion and Assembly Planning**: Here, the motion planner functions as a subroutine, invoked by the assembly planner as needed during the planning process, with parallel coordination with the task allocation planner.

(c) **Post-assembly Motion Planning**: The assembly planner first creates a tentative plan, followed by the task planner. The motion planner then assesses this plan's feasibility,

leading to revisions that incorporate new insights or constraints from the motion planning phase, while updating the task allocation plan.

The integration of IR arms into the prefabrication process has become increasingly popular, driven by decreasing costs and enhanced capabilities of these robots. IR arms, equipped with advanced controls and high DOF manipulators, are adept at performing repetitive and precise tasks such as component assembly autonomously, accommodating various building designs. However, a major challenge in using IR arms for prefabrication is creating viable assembly plans and ensuring collision-free paths for the assembly of building components. Planning the assembly sequence is crucial, as it affects tool change frequency, assembly directions, and the design of necessary fixtures, impacting overall process efficiency. Choosing the right assembly sequence from numerous possibilities is complex due to the vast potential combinations, which increase exponentially with the number of parts involved. In many industrial settings, assembly planning often relies on the knowledge of engineers rather than systematic analysis, despite significant room for optimization. This complexity often stems from the challenges in evaluating possible assembly sequences given the constraints inherent in constructing them.

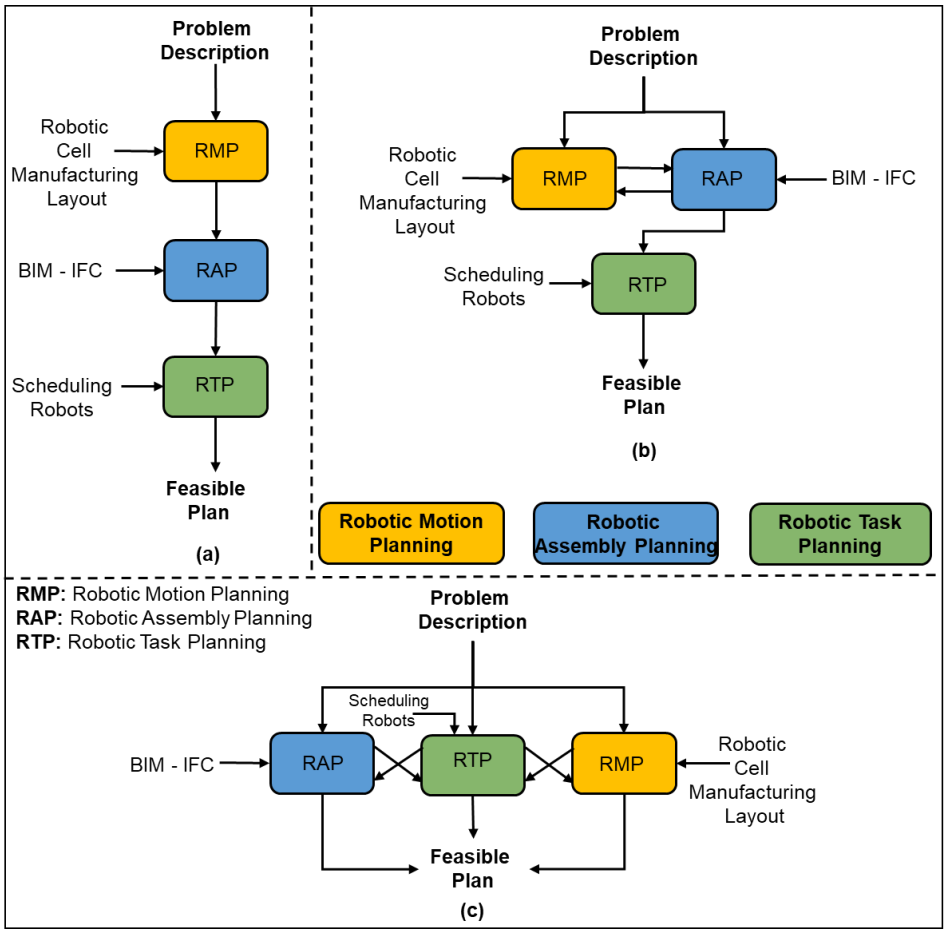


Figure 2. Assembly, Task, Motion Planning Styles

Off-site construction (OSC) diverges from standard manufacturing like automotive production due to its use of diverse components in each project (Figure 3). The RATMP must not only generate feasible assembly sequences but also effectively allocate tasks among robots and optimize their motion plans. This involves planning how components combine into complex units that meet design specifications. Despite research in robotic assembly, methods often fall short in OSC as they are developed independently and don't address its unique challenges (Wan et al., 2017).

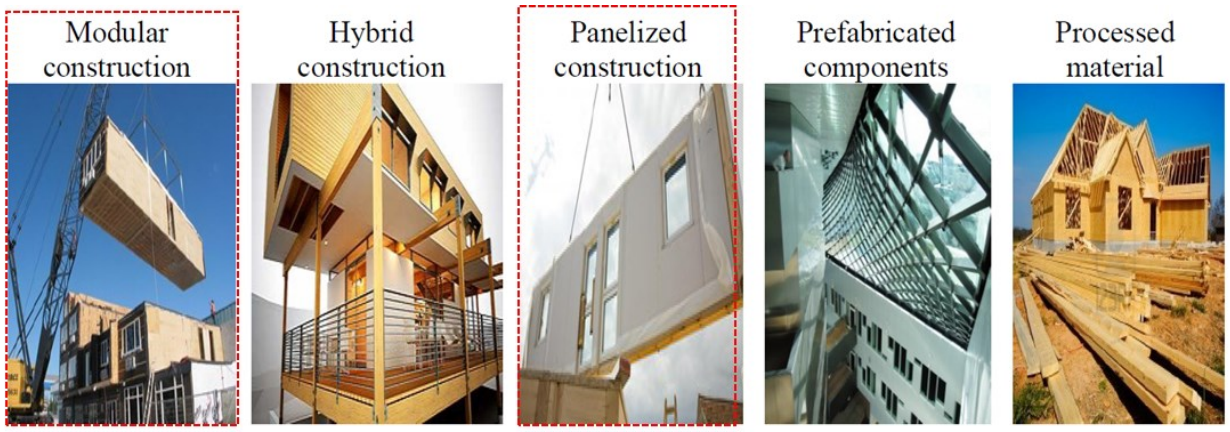


Figure 3. OSC Categories and Types

Unlike standardized manufacturing, OSC deals with variability in product designs, requiring sophisticated algorithms to manage multiple product variants and automate assembly (Iturralde and Bock, 2018). Optimizing assembly processes before production is crucial to enhance manufacturing efficiency, reduce costly revisions, and support timely production (Zhang et al., 2013). Modern systems increasingly rely on digitalization to validate assembly plans through virtual simulations before physical implementation, allowing informed decision-making (de Giorgio et al., 2021; Wang et al., 2020).

Robot Operating System (ROS) is a comprehensive suite of software libraries designed for building robotic systems, known for its modular architecture and integration with platforms like Linux and Windows. Since its inception in 2007, ROS has become widely used in academia and industry, and its advanced version, ROS 2, introduced in 2014 under the Apache 2.0 license, supports a broad spectrum of robotics applications (Quigley et al., 2009). ROS 2's ecosystem includes middleware for component communication, a range of essential algorithms, and developer tools for various operational needs such as visualization, debugging, simulation (refer Figure 4).

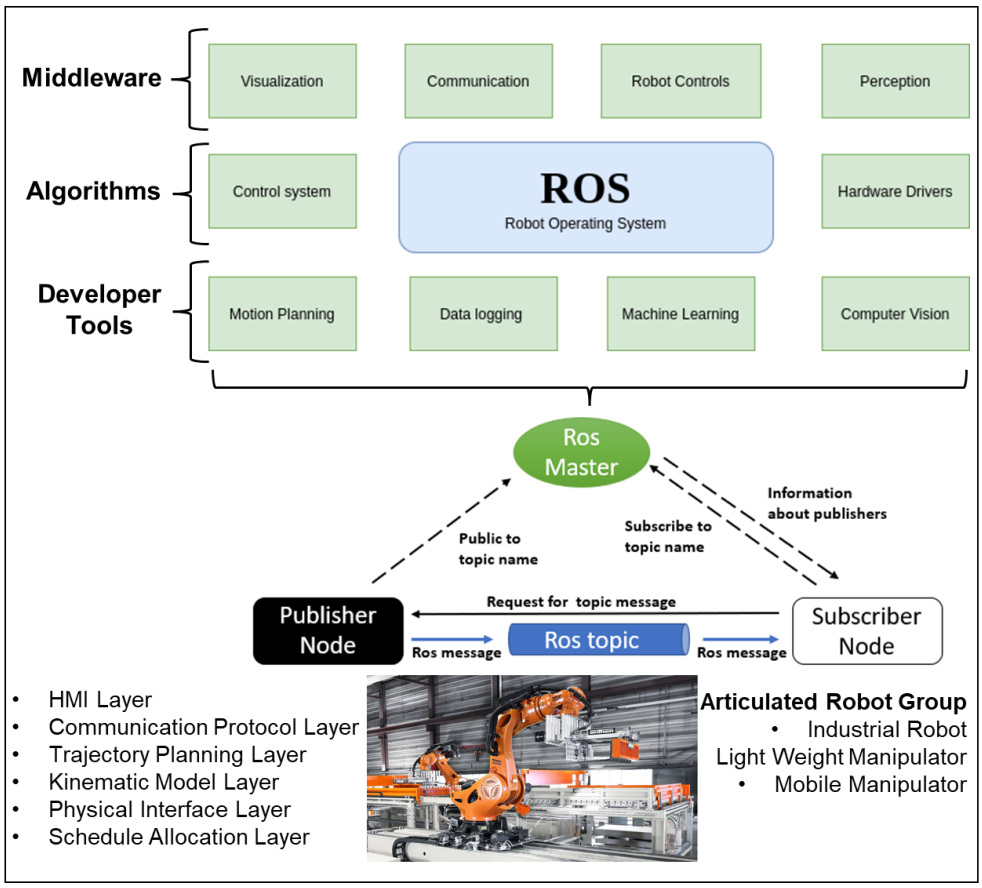


Figure 4. ROS Software Ecosystem

Virtual Commissioning (VC) enhances production engineering efficiency and quality while shortening implementation time. Introduced as "soft commissioning" in 1999, VC allows for the integration of simulation models with real-world entities to test system behaviors before physical setup (Auinger et al., 1999). Research has evolved to encompass the entire lifecycle of systems, involving mechanical design, machine control, and the integration of control systems with actual equipment (Kiefer, 2007; Makris et al., 2012). For successful VC implementation, it requires detailed simulation models, precise layout of production cells, defined material flows, and comprehensive IT infrastructure to bridge theoretical models with practical applications (Lee and Park, 2014) as illustrated in Figure 5.

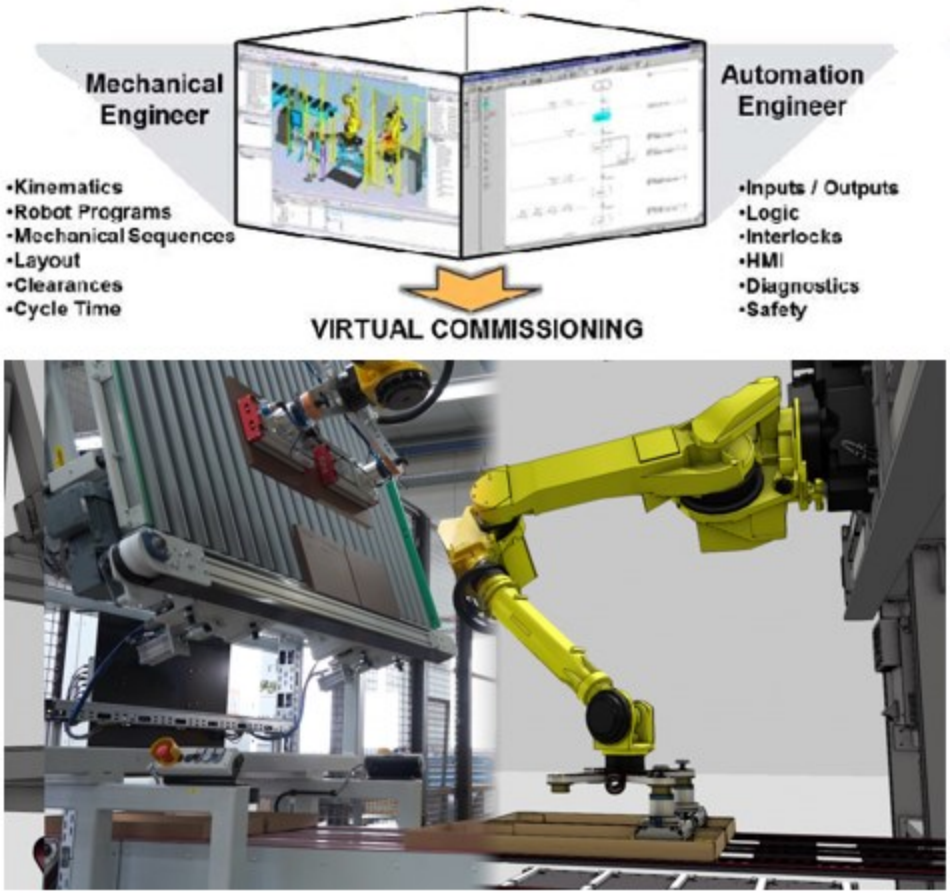


Figure 5. Sample VC Project Setup

1.3 Research Motivation, Challenges, and Proposed Solutions

The motivations driving this research initiative, as shown in Figure 6, are:

(1) *Research and Development Focus*: Aiming to advance OSC manufacturing through the implementation of automated systems in line with the Construction/Industry 4.0 vision.

(2) *Integration Challenges*: While various robotic assembly, task, and motion methods and tools have been developed, their integration in multi-robotic OSC prefabrication processes remains ineffective due to their isolated development.

(3) *Tool Development Needs*: There is a shortage of tools specifically designed for multi-robotic systems in OSC applications, which require advanced, adaptable methods and algorithms for assembling diverse product variants.

(4) *Robotics Innovation*: Current construction robots are mainly designed for repetitive tasks; however, the potential for highly autonomous or intelligent robots in construction is yet to be fully realized.

(5) *Industry Adoption Discrepancies*: Although IR is extensively used in the automotive industry, its adoption in the AEC sector lags, typically limited to unconventional, one-off projects rather than becoming standard practice.

(6) *Technological Adoption Barriers*: The overall adoption of digital technologies in OSC is slow, often impeded by challenges in understanding how to identify, assess, and select appropriate technologies.

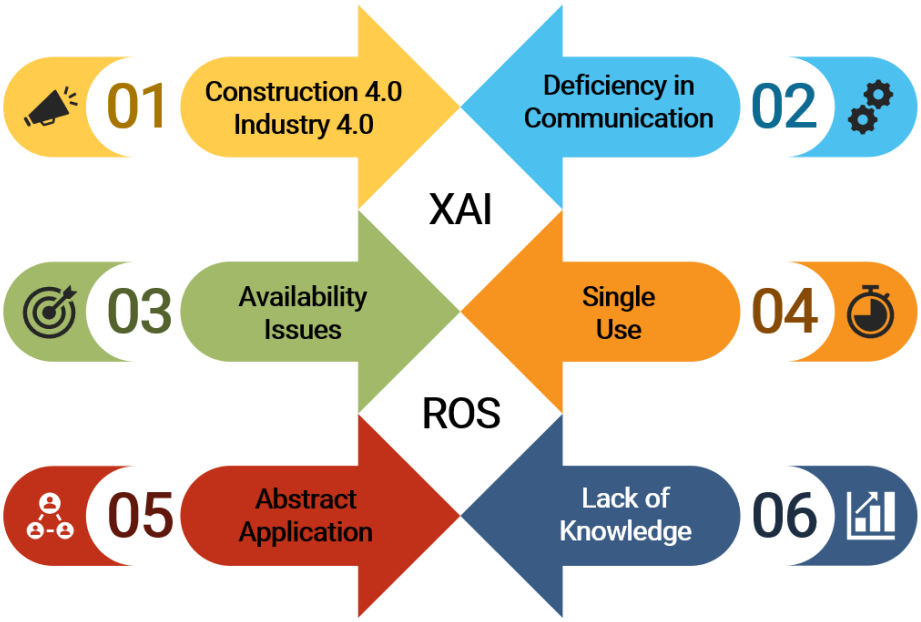


Figure 6. Research Motivation

The challenges of integrating robotics within the construction sector and their proposed solutions are summarized in Figure 7 as follows:

(1) *Challenge:* Component Complexity - The construction industry requires the assembly of thousands of diverse components, which can be daunting for robotic systems due to their complexity.

Solution: Develop a specialized tool designed to efficiently plan and manage the assembly of numerous components.

(2) *Challenge:* Compliance with Building Regulations - Ensuring adherence to building regulations, including structural, connection, or shipping standards, is essential yet challenging for OSC manufacturers using automated systems.

Solution: Implement an automated tool that verifies all designs meet applicable building regulations prior to production.

(3) *Challenge:* Diverse Robotic Motions - Construction tasks often demand complex and varied robotic actions, such as placing different types of studs, assembling frames, and installing elements like windows or doors.

Solution: Create intelligent algorithms that can automatically generate and adapt robotic motions for various construction activities.

(4) *Challenge:* Multi-Robot System Integration - While a single robot can perform basic movements, optimal performance requires integration with additional robots and components like sensors and end effectors.

Solution: Design and evaluate a comprehensive system that ensures seamless interaction among various components through a unified design approach.

(5) *Challenge:* Complex System Interactions - Managing complex interactions between various mechanical and software components in a multi-robot system to analyze scenarios and select feasible plans is intricate.

Solution: Develop an interfacing tool that facilitates communication among different system components using standard industrial communication protocols.

(6) *Challenge:* Diagnostic Clarity - Robotic planning tools often lack clear feedback on their outputs, complicating user understanding of why a plan failed or a trajectory deviated.

Solution: Enhance robotic operation systems to provide detailed explanations of failures and recovery strategies, thereby improving user comprehension and trust.

These solutions aim to tackle significant operational challenges faced by robotics in construction, thereby enhancing efficiency and reliability in automated OSC environments.

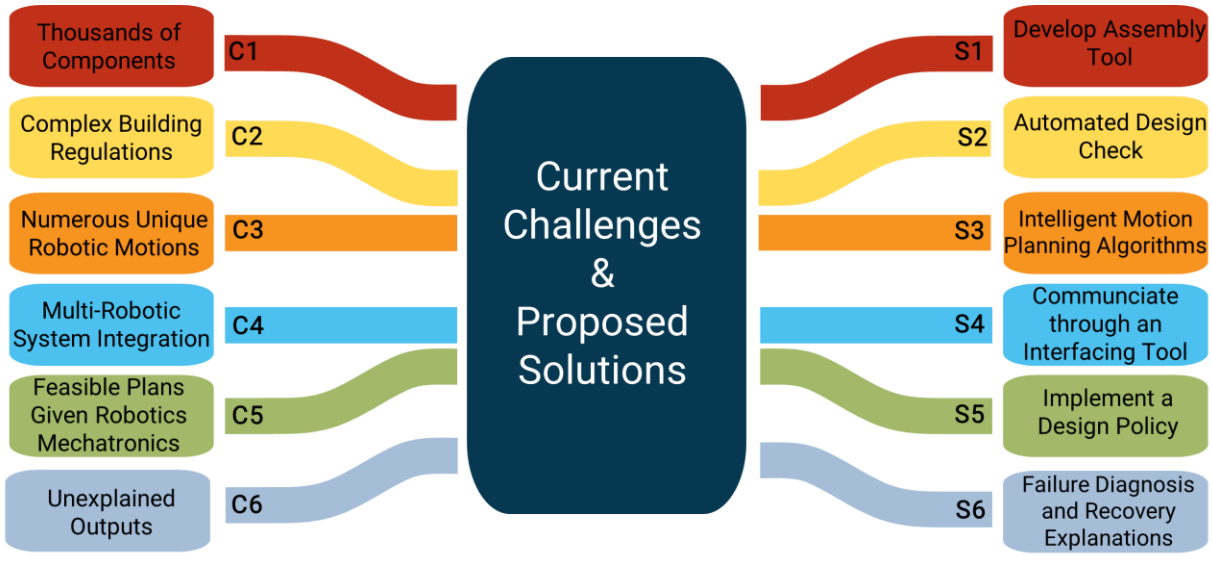


Figure 7. Challenges and Solutions

1.4 Problem Statement, Research Questions and Hypothesis

The research problem was identified through a literature review, focusing on the concurrent advancements in BIM and IR within OSC. This review identified three key areas: Robotic Assembly Planning (RAP), Task Allocation Planning (TAP), and Robotic Motion Planning (RMP), highlighting a significant gap in the technological integration of BIM and IR through the RATMP in OSC.

Despite the potential for increased productivity and safety with OSC and robotic manufacturing, their integration remains poorly explored. This research outlines technological fragmentation between BIM and IR in OSC, identifying clear research gaps and hypotheses (refer to Figure 8). The goal is to enhance technological and digital integration within OSC.

Utilizing the Design Science Research (DSR) methodology, the framework is developed, validated, and evaluated across two industrial OSC sectors through multiple case studies. This study explores three primary research questions linked to hypotheses from the literature:

(1) RQ1: What is the current level of technological integration between BIM and IR in OSC, and how can it be improved?

(2) RQ2: How does RATMP support the integration of BIM and IR in OSC?

(3) RQ3: Can RATMP enhance automated robotic operational intelligence decision-making systems in OSC?

These questions are addressed by testing the following hypotheses:

(1) H1: RATMP is essential for achieving technological integration between BIM and IR.

(2) H2: RATMP facilitates the integration of BIM and IR through virtual commissioning in OSC.

(3) H3: Integrating RATMP in OSC supports an automated robotic operational intelligence decision-making system.

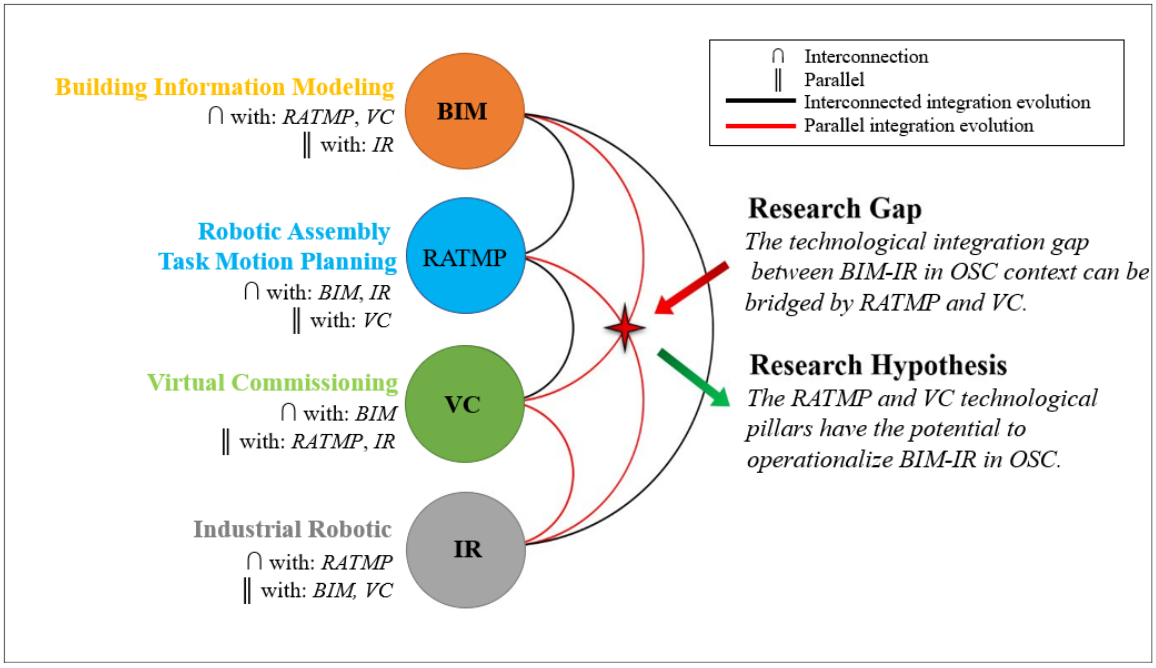


Figure 8. BIM-IR Technological Integration – Research Gap and Hypothesis

1.5 Thesis Organization

The organization and structure of this thesis are illustrated in Figure 9. The primary objective of the RATMP framework is to enhance the technological integration of BIM and IR within OSC environments. This is achieved by integrating assembly planning, task allocation, and motion planning algorithms into a unified tool for multi-robotic systems.

This research contributes significantly to the existing knowledge by creating a systematic workflow that integrates the three essential elements of technological integration—BIM, IR, and RATMP—within OSC systems. The RATMP framework not only advances OSC systems but also

supports the programming, simulation, and operationalization of IR specifically for construction applications. This study introduces innovative methodologies that leverage existing technological tools widely used in the AEC industry.

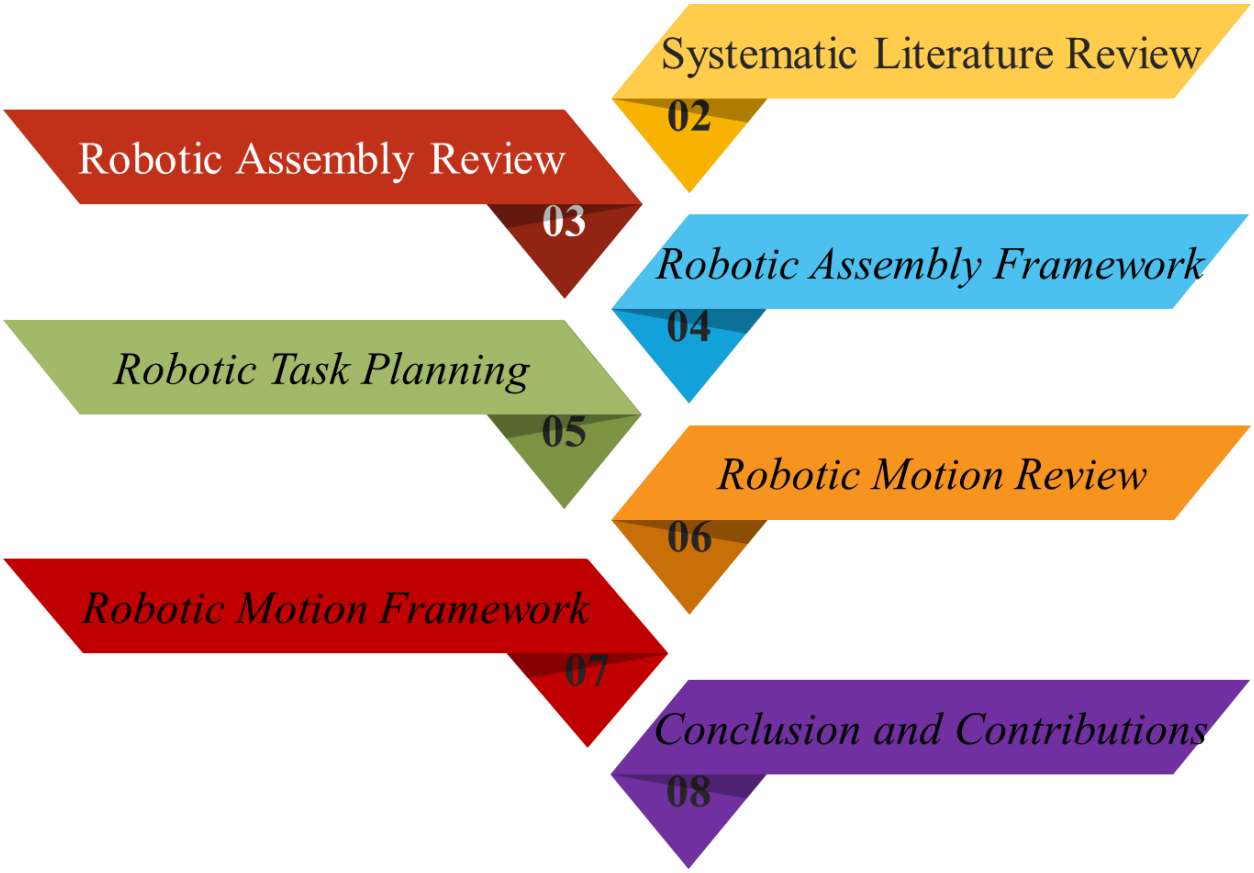


Figure 9. Research Organization Scheme

Chapter 2 : Systematic Literature Review

The AEC industry is experiencing a digital transformation that enhances its performance, productivity, and competitiveness through the adoption of BIM. BIM promotes technological integration, transforming design and management activities, particularly in OSC. Despite its advantages, BIM's integration with digital manufacturing processes in OSC, such as IR, remains underexplored. This study utilizes the Systematic Literature Review (SLR) method to assess the technological integration of BIM and IR tools in OSC, aiming to operationalize IR via BIM. The

findings highlight the need for a reconfigurable framework to bridge the gap between BIM and IR in OSC.

The core philosophy of BIM involves all project stakeholders using a centralized digital model accessible throughout a construction project's lifecycle, enhancing modelling, management, and collaboration (Race, 2019). Concurrently, automation in OSC, often incorporating IR, seeks to apply industrial automation principles to construction (Siciliano et al., 2008; Dachs et al., 2022). The review identifies pivotal research questions regarding the joint utilization of BIM and IR, focusing on technological integration—defined by IEEE in 1990 as the ability of systems to exchange and utilize information, though later expanded to include multiple dimensions (Poirier et al., 2014).

Past research has advanced robotic applications in OSC, mainly in non-standard projects, but has often relied on proprietary CAD platforms limiting broader implementation (Eversmann et al., 2017; Søndergaard et al., 2016). This study advocates for the adoption of open BIM standards like Industry Foundation Classes (IFC) to ensure seamless workflows in automated OSC environments as illustrated in Figure 10 .

As robot technology advances and costs decline with a 28% reduction over ten years according to (Statista, 2017a), the integration of robots in construction presents a promising avenue to enhance productivity and labor efficiency. However, the construction sector has been slow to adopt IR, with 83% of OSC companies not implementing robots as of 2017 (Statista, 2017b). Research indicates that robotic construction, such as the In-situ Fabricator developed by ETH Zürich, can offer superior productivity for complex tasks (Giftthaler et al., 2017; García de Soto et al., 2022). Nevertheless, challenges remain, particularly the need for precise information for robotic

operations, a gap that BIM could fill by providing detailed data for automated fabrication processes.

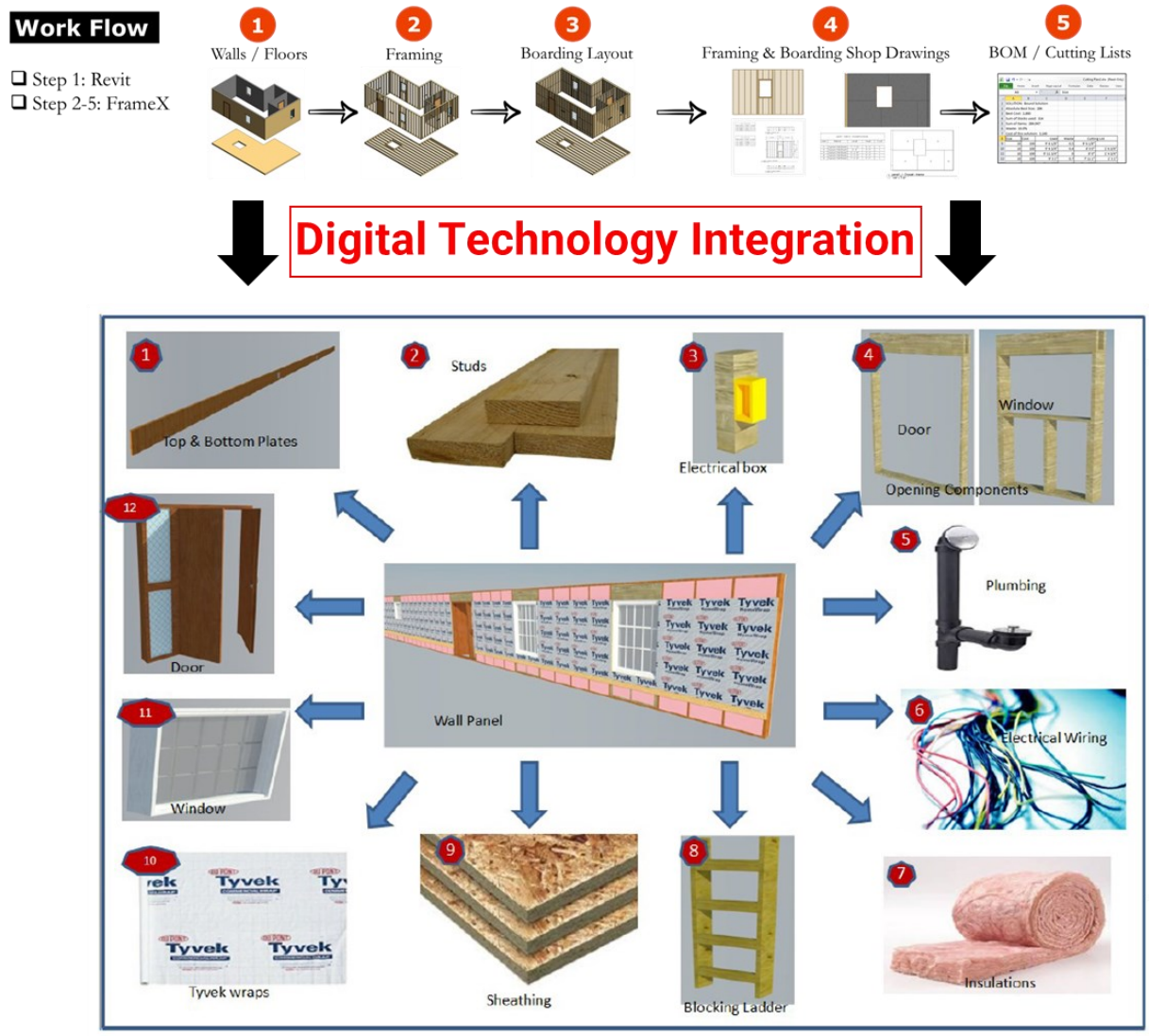


Figure 10. BIM-IFC CAD Application

Overall, this literature review aims to pave the way for further research into the convergence of BIM and IR, addressing the technological and practical challenges to fully leverage digital tools in OSC.

2.1 Manufacturing Automation in Offsite Construction

Automation in OSC requires the integration of products, organizational structures, informational aspects, and machine technology to enhance efficiency and ensure quality, crucial for product customization through automated systems. This integration, which includes modularity in products, effective organizational setups, and computer-aided information chains, has contributed to the success of Japan's large-scale prefabrication industry, known for delivering higher quality and more unique buildings than traditional methods (Bock and Linner, 2015).

One major advantage of OSC is its potential to reduce construction time by 30-50% compared to traditional methods, thanks to the ability to simultaneously prepare sites and manufacture elements or modules in a factory setting (MBI, 2018; Mah and Al-Hussein, 2008). However, OSC often underperforms in achieving time and cost efficiencies and enhancing labor productivity. Addressing these challenges is essential for maximizing the benefits of OSC.

Digital technology is increasingly crucial for enhancing productivity in the construction industry. Surveys in the Chinese OSC sector and among U.S. contractors reveal a strong belief that the lack of advanced technology implementation is a significant barrier to OSC development. Over 90% of Chinese construction enterprises and 87% of U.S. contractors view advanced technology as essential for more precise and efficient OSC (Cheng and Ma, 2020; Hoover et al., 2017).

The levels of automation in construction prefabrication, as shown in Figure 11, form a three-tiered pyramid. At the base, many prefabrication companies still depend heavily on manual processes or semi-automatic machinery, requiring significant skilled labor on-site and off-site. The middle tier includes technologies like integrated databases, robots, automatic machinery, and CAD software, which aid in data analysis and quality control, shifting skilled labor demands more off-site. The peak represents the integration of advanced information systems like BIM and Enterprise Resource

Planning (ERP) systems, where the use of robotics and sophisticated design integration approaches levels of automation seen in the manufacturing sector.

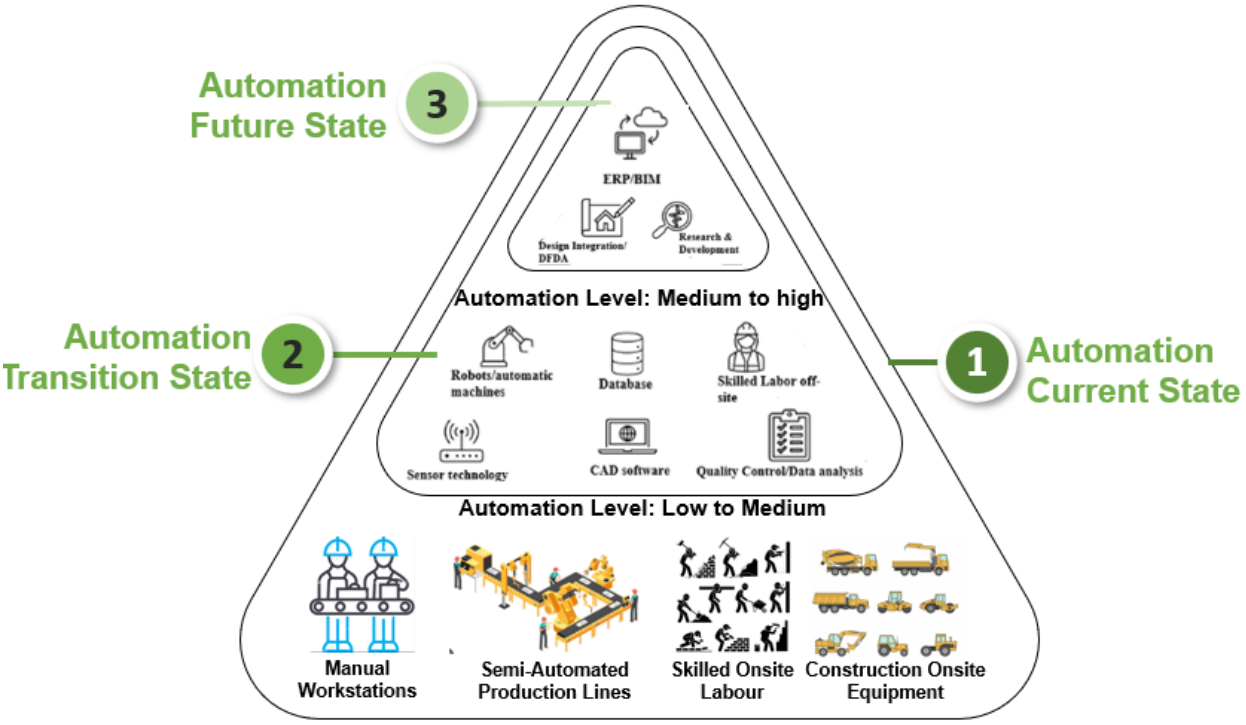


Figure 11. Automation State According to the Literature Review

2.1.1 Integration of Innovative Robotics Technologies in Offsite Construction

The adoption of automation and robotics throughout the entire project lifecycle—from design to deconstruction—can be effectively facilitated through strategies such as Robot Oriented Design (ROD), robotic industrialization, construction robots, ambient robotics, and site automation. ROD streamlines the transformation of raw parts into higher-level assemblies within highly mechanized and automated industrial settings, significantly reducing on-site complexity and enhancing productivity, efficiency, and economic outcomes (Bock and Linner, 2015; 2016).

Robotic Process Automation (RPA) automates routine, rule-based tasks, delivering substantial benefits by increasing efficiency. In contrast, the ROS serves as an evolving open-source environment for complex robotic programming, including Artificial Intelligence (AI) and machine learning capabilities. While RPA focuses on repetitive task execution, ROS enables sophisticated "thinking" and "learning" operations.

ROS, while not an operating system, acts as a comprehensive framework for robot software development, offering hardware abstraction, low-level device control, commonly used functionalities, message-passing, and package management (Zhang et al., 2022). ROS 2 features nodes, topics, services, and parameters for enhanced modularity (Macenski et al., 2022). ROS-Industrial, an extension of ROS, brings these capabilities into manufacturing automation, providing interfaces for industrial manipulators, sensors, and networks. It includes libraries for tasks such as sensor calibration and motion planning, offering a flexible, software-based solution for industrial robot programming, contrasting with traditional external proprietary controllers.

2.1.2 Virtual Commissioning Systems in Industrial Robotics Automation

Virtual Commissioning (VC) is a crucial method for understanding the benefits and challenges of construction robotics by simulating, modeling, and testing a production system's behavior before actual commissioning. This pre-simulation verifies process functionality, validity, and effectiveness, ensuring time and cost savings while minimizing downtime by pre-emptively addressing issues (Pérez et al., 2020).

VC tests manufacturing systems, including mechanical/electrical components and control programs, via simulations conducted before physical implementation. The success of VC relies on detailed manufacturing system models that reduce debugging and correction efforts during real

commissioning (Bokor et al., 2019). It also supports the analysis, testing, visualization, and commissioning of robotic manufacturing processes.

Implementing robotic VC requires expert knowledge across robotics and construction, demanding multidisciplinary collaboration among contractors, developers, and researchers. This collaboration relies on precise information, like object dimensions and reference frames, to ensure reliable robotic task performance (Kim et al., 2021).

VC is vital in robotics research and development for testing control algorithms' efficiency, safety, and robustness. In construction, integrating BIM, CAD, and ROS with simulation tools like Gazebo and Rviz offers new possibilities. For instance, Kim et al. (2021) developed a BIM prototype for task planning in robot-assisted interior wall painting, demonstrating VC's practical application in optimizing construction robotics tasks.

2.2 The Systematic Literature Review (SLR) Methodology

The aim of this study is to synthesize the body of knowledge in coupling BIM and IR using a SLR approach, adapted from Kitchenham (2004), and illustrated in Figure 12. This five-step iterative SLR process encompasses identifying research, conducting bibliographic research, evaluating eligibility, performing bibliometric analysis, synthesizing data, and formulating hypotheses.

Conducted over three cycles utilizing databases like Scopus and Dimensions, which cover extensive engineering research (Mongeon and Paul-Hus, 2016; Singh et al., 2021; Waltman et al., 2010), the initial cycle emphasizes the technological integration between BIM and IR. Findings indicate limited research on integrating these technologies, prompting a shift towards exploring solutions for bridging BIM and IR, ultimately recognizing the RATMP framework as a viable method for enhancement.

The bibliographic research involves filtering relevant publications using specific keywords, followed by an eligibility assessment and a detailed bibliometric analysis with VosViewer software, as illustrated in Figure 13, to explore keyword co-occurrences (Waltman et al., 2010). Although interest in this topic has grown, the literature predominantly remains theoretical, with few real-world case studies demonstrating BIM-IR integration. Moreover, existing studies vary widely in their integration approaches, contributing to inconsistencies in effective strategies for OSC.

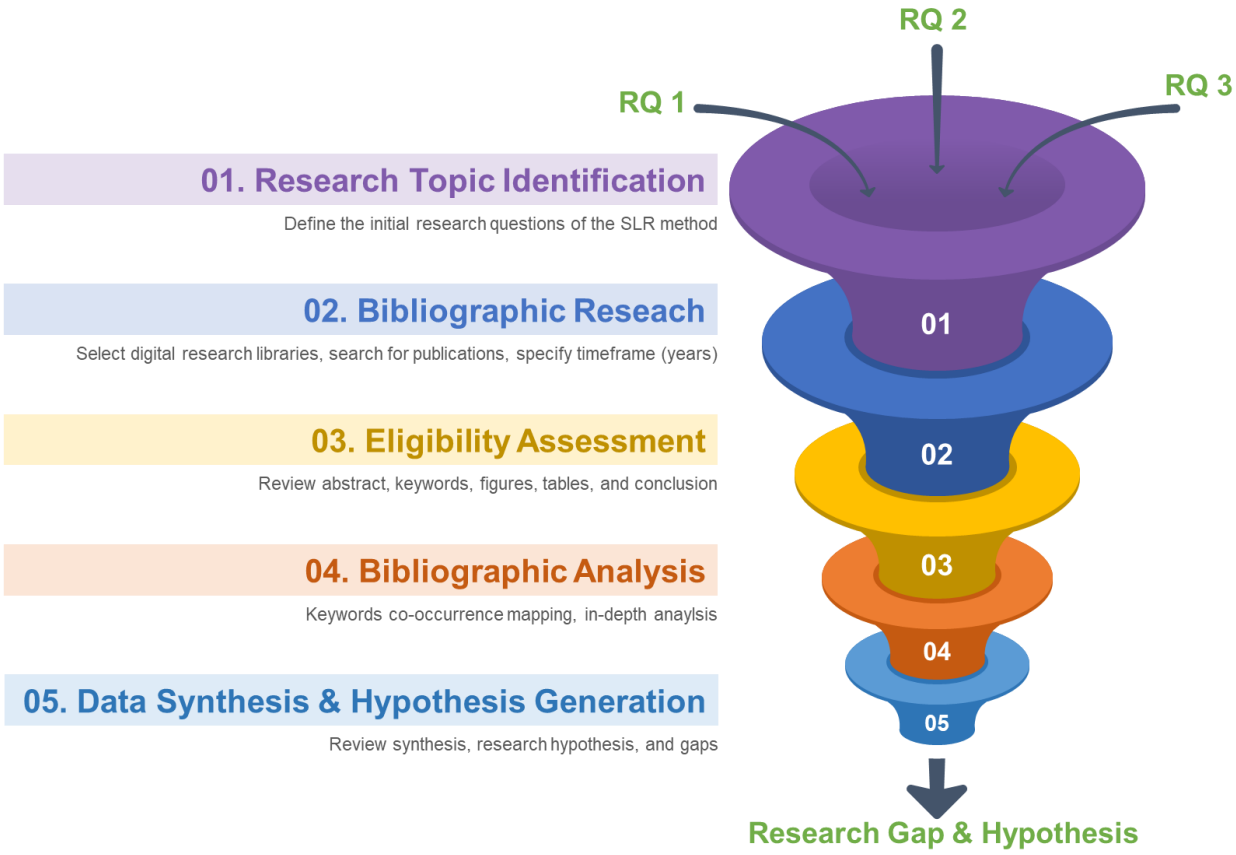


Figure 12. SLR Funneling Methodology

The first SLR cycle's findings led to the second cycle, which investigates integrating a VC system compatible with BIM-IR-RATMP. This confirmed VC's suitability, paving the way for the third cycle, which assesses the need for an automated robotic operational intelligence decision-making

system to validate and enhance BIM-IR data integration. This system employs a data analytics scheme to improve communication and data analysis.

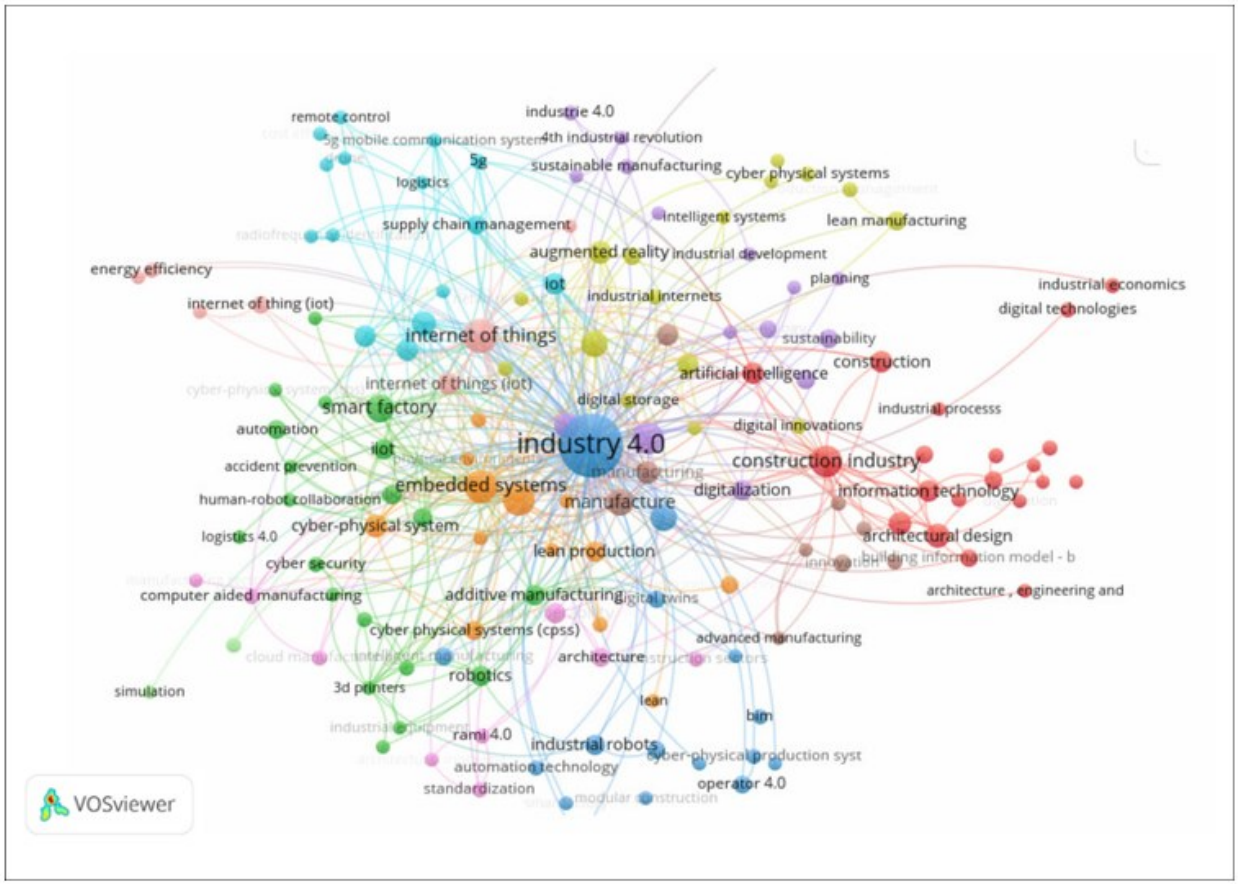


Figure 13. Bibliometric Mapping of BIM-IM Technological Integration in Industry 4.0 (2013-2023)

The research is divided into three sections: The first explores BIM-IR integration and the role of RATMP; the second evaluates VC's potential to connect RATMP with BIM and IR; the third examines the implementation of an automated decision-making system.

BIM and IR integration approaches are categorized according to Janssen's (2015) framework, shown in Figure 14, distinguishing between loosely (a), moderately (b), and tightly (c) coupled approaches. These range from model exchange via file transfer to using Application Programming

Interfaces (APIs) for direct model exchanges, illustrating the varying degrees of achievable integration between BIM and IR systems.

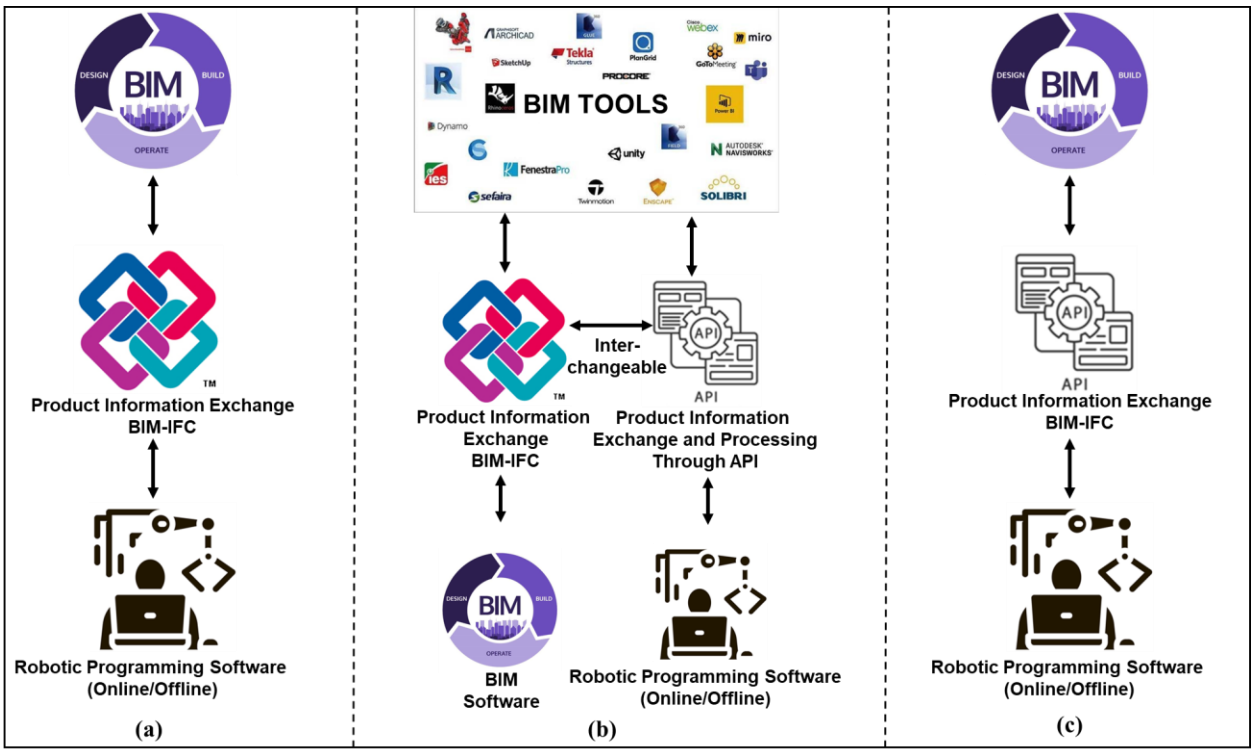


Figure 14. BIM and IR Coupling Approaches

2.3 Preliminary Data-Synthesis and Discussion

The literature review was conducted over three SLR cycles, analyzing 450 documents focused on integrating BIM, IR, RAP, RTP, RMP, and VC within an OSC environment. This review evaluated their joint and parallel technological developments in research, examining the volume of published articles, their deployment, and application in industrial case studies. The following categorizations were illustrated in Figure 8:

- Interconnected Technological Evolution: BIM-RAP / RMP-IR
- Parallel Technological Evolution: BIM / RAP – RTP - RMP / IR
- Central Interconnecting Pillar: VC

This process led to the formulation of three research hypotheses aimed at addressing identified gaps and advancing the objectives of this proposal:

- *Hypothesis 1 (H1)*, emerged from the initial SLR cycle, focusing on the parallel BIM-IR integration. It proposes that RAP-RMP serves as a bridge for technological integration between BIM and IR, indicating the need for further research to validate these connections.
- *Hypothesis 2 (H2)*, developed during the second SLR cycle, suggests that RTP is suited to connect the BIM-RAP and RMP-IR technological evolutions, given the established interconnection between RAP and RMP, positioning RTP as a conduit for BIM-IR integration.
- *Hypothesis 3 (H3)*, conceived in the context of VC's role, posits that VC could autonomously operationalize and intelligently guide IR within OSC. It highlights the strong connection between IR integration in OSC manufacturing and RATMP, supported by BIM, emphasizing the potential of leveraging interconnected technological pillars for achieving the research objectives.

In summary, the study presents a structured approach to understanding the complex interrelations and potential synergies among BIM, IR, RATMP, and VC in OSC, aiming to advance the field through targeted technological integration.

2.4 Design Science Research (DSR) Methodology

This section details the DSR methodology, based on the "science of the artificial" philosophy, which aims to create artifacts for addressing real-world issues (Hevner et al., 2010). The DSR methodology, thoroughly explained by Hevner and Chatterjee (2010), consists of six main steps: (1) Identifying the motivation and problem; (2) Defining solution objectives; (3) Designing and

developing the artifact; (4) Demonstrating the artifact; (5) Evaluating the artifact; and (6) Communicating the results. These steps are illustrated in Figure 15.

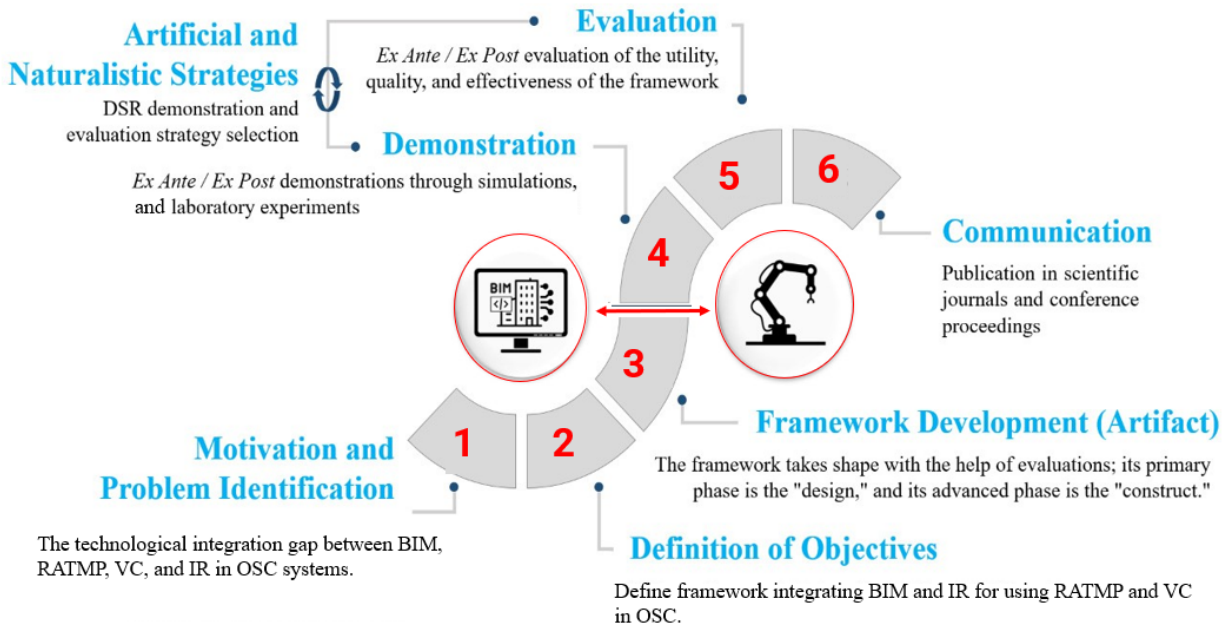


Figure 15. DSR Methodology Process (Adopted and Revised from Hevner and Chatterjee, 2010)

The methodology emphasizes the dual contexts of artifact development: Ex-Ante and Ex-Post. The Ex-Ante context involves preliminary demonstrations and evaluations of the artifact as a "design" in its initial form, acknowledging that research and artifacts are evolutionary and continuously improving (Carpo, 2017). This process is cyclical, oscillating between the Ex-Ante and Ex-Post contexts through what is termed the "design cycle" as shown in Figure 16. In this study, the development of the RATMP framework undergoes two evaluation cycles within the Ex-Ante context. The first cycle includes computer simulations, while the second utilizes laboratory experiments in a real-world Ex-Post context.

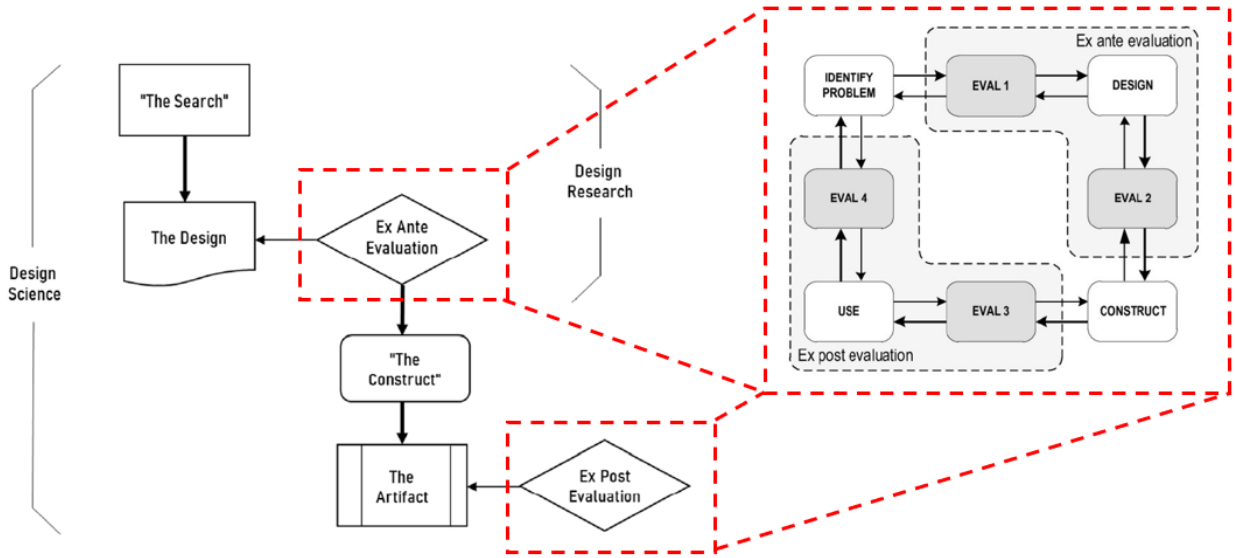
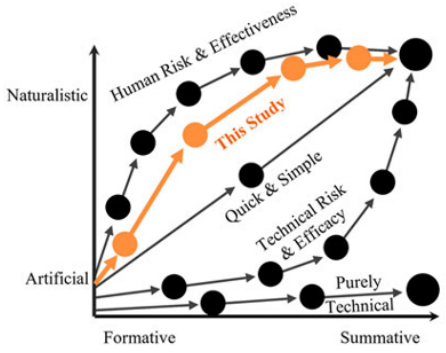
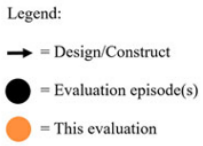


Figure 16. Ex-Ante/Ex Post in DSR Methodology with Evaluation Cycle

The evaluation phase in DSR adheres to Venable et al. (2016) technical risk & efficacy strategy extracted from the Framework for Evaluation as shown in Figure 17. This approach is particularly suited to this project as it emphasizes artificial demonstrations and includes provisions for real-world assessments. It is especially appropriate for projects where evaluations with real users and systems are prohibitively costly, further justifying its use in this research context.





DSR Evaluation Method Selection Framework	Ex Ante	Ex Post
Naturalistic	• Action Research • Focus Group	• Action Research • Case Study • Focus Group • Participant Observation • Ethnography • Phenomenology • Survey (qualitative or quantitative)
Artificial	• Mathematical or Logical Proof • Criteria-Based Evaluation • Lab Experiment • Computer Simulation	• Mathematical or Logical Proof • Lab Experiment • Role Playing Simulation • Computer Simulation • Field Experiment

Figure 17. FEDS Evaluation Method Selection Framework

2.5 Main Research Methodology

The literature review identified significant ambiguity in the technological integration between BIM and IR. This research aims to clarify this integration and enhance the design process by developing the RATMP approach, incorporating a reconfigurable robotic operating system to streamline construction tasks in OSC.

The primary research objective is to design a comprehensive RATMP framework to automate framing operations in OSC, aiming to improve productivity and performance by:

- Standardizing tasks,
- Accelerating framing operations,
- Enhancing accuracy and minimizing errors,
- Enabling continuous operation,
- Reducing safety-related disruptions.

The system employs versatile autonomous robotic manufacturing cells controlled via a VC interface. This setup includes a flexible RATMP tool adaptable to various framing tasks (e.g., nailing, cutting, drilling) and materials (e.g., wood, steel), using a mathematical model that integrates AI algorithms and BIM models to meet design criteria and user requirements.

Additionally, the VC system simulates, tests, and validates manufacturing activities, optimizes robot motion to ensure safe and precise operations, and prevents collisions and errors. Supported by a VC tool that integrates data from various sources, the framework promotes an automated operational intelligence decision-making system for effective robot guidance. The overall methodology is presented in Figure 18.

The research is structured around four key objectives to achieve its goals:

- **Objective A** - *Design*: Integrate an interface for exporting BIM/CAD models with a smart robotic assembly sequence planner.
- **Objective B** - *Allocate*: Develop a two-stage scheduling strategy to create a balanced schedule for multiple robots and ensure an efficient, collision-free plan.
- **Objective C** - *Plan*: Model the properties of the industrial robotic manufacturing cell and integrate it with a robotic motion planner.
- **Objective D** - *Simulate and Analyze*: Create an interface to simulate and execute outputs from the assembly, task, and motion planning, using an animation-based robotic programming method and perform feasibility evaluations within a VC environment.

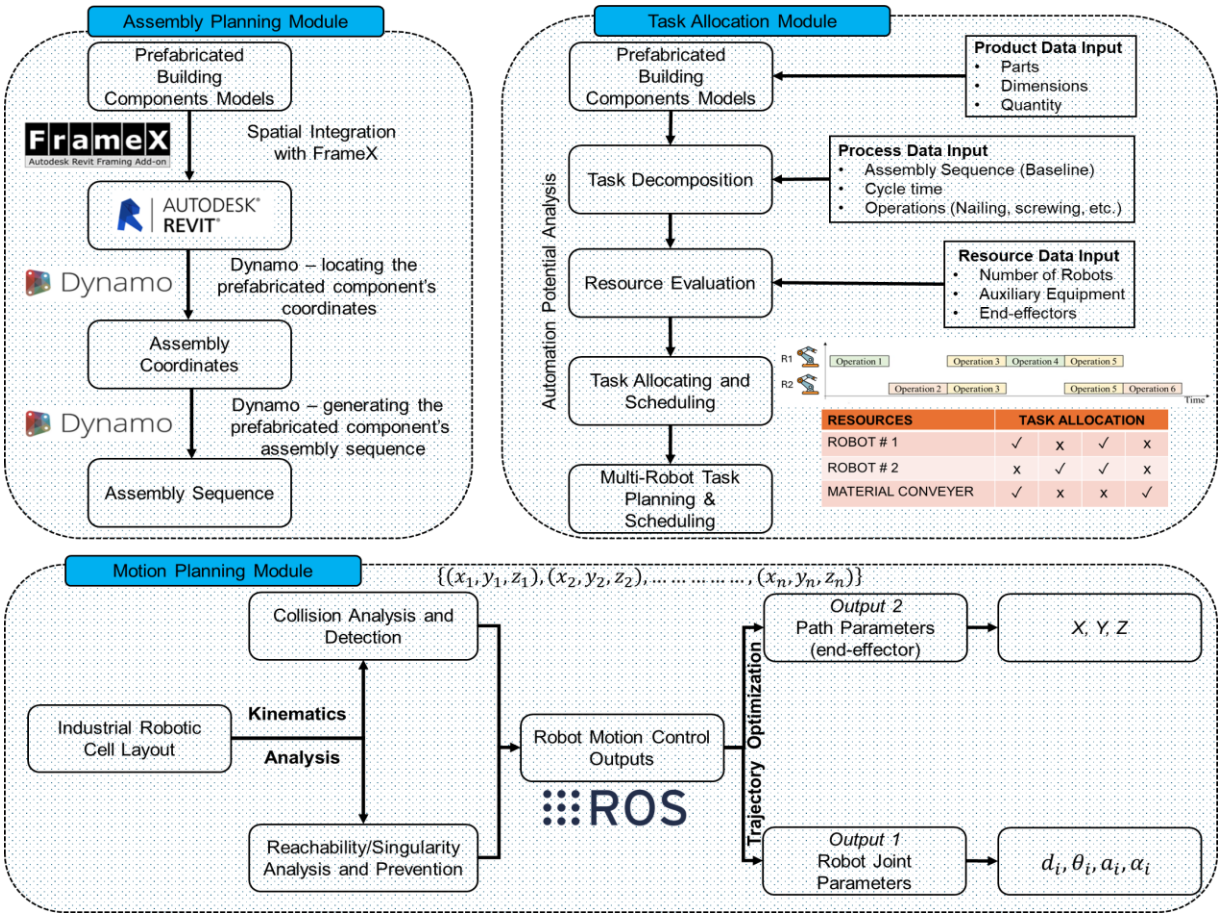


Figure 18. RATMP Framework Enabled by BIM-IR Approach for Manufacturing in OSC

2.6 RATMP Framework

This research explores the application of seven DOF IR arms, vital for tailoring manufacturing processes to specific construction needs. These IR arms are adaptable, supporting various end effectors like extruders and grippers, and can interface with different technological components such as sensors and conveyors. However, frequent and unpredictable design changes in construction make traditional manual programming inadequate (Brell-Cokcan and Braumann, 2013), and standard CAM processes do not meet the unique programming requirements of construction workflows.

Robotic arm programming defines toolpaths via the tool center point (TCP), which dictates the trajectory and manipulations of the attached tool, adjusting through joint rotations via inverse kinematics (Renaud, 2000). These instructions are typically visualized in a simulator to ensure correct operation before execution, a practice in offline programming to avoid malfunctions (Devadass et al., 2019). Design modifications necessitate manual reprogramming, reducing flexibility in the dynamic construction industry.

To address these limitations, this research proposes an autonomous RATMP approach for robotic programming, starting with informed models of the robotic arm, tool, and workpiece (as shown in Figure 19), processed through an algorithm-aided transformation to eliminate constant switching between modeling and programming software. This integrated process enhances the flexibility of IR systems by allowing direct modifications in design and programming within integrated User Interfaces (UIs) and automatically generates robot-specific commands. The resulting robotic program is managed within the BIM environment, enabling collaborative production management through the RATMP tool and ensuring design and manufacturing programming are instantly responsive to changes.

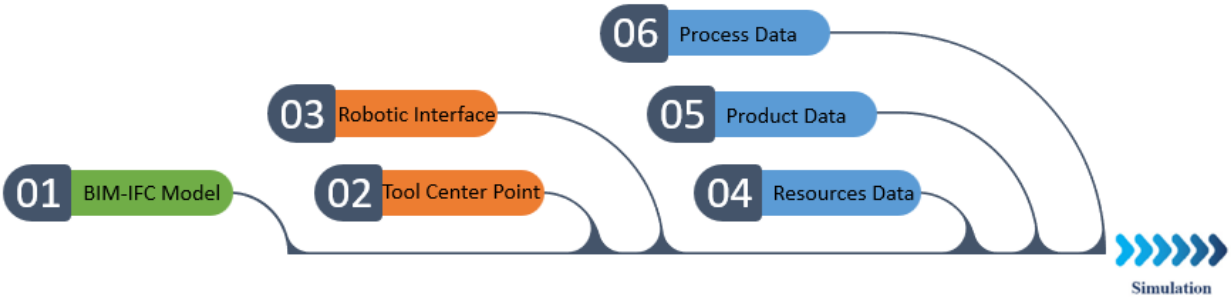


Figure 19. Planners Inputs

The research proposes a framework structured into four main sections targeting four research objectives, illustrated in Figure 20, integrating various stages of the construction and manufacturing process.

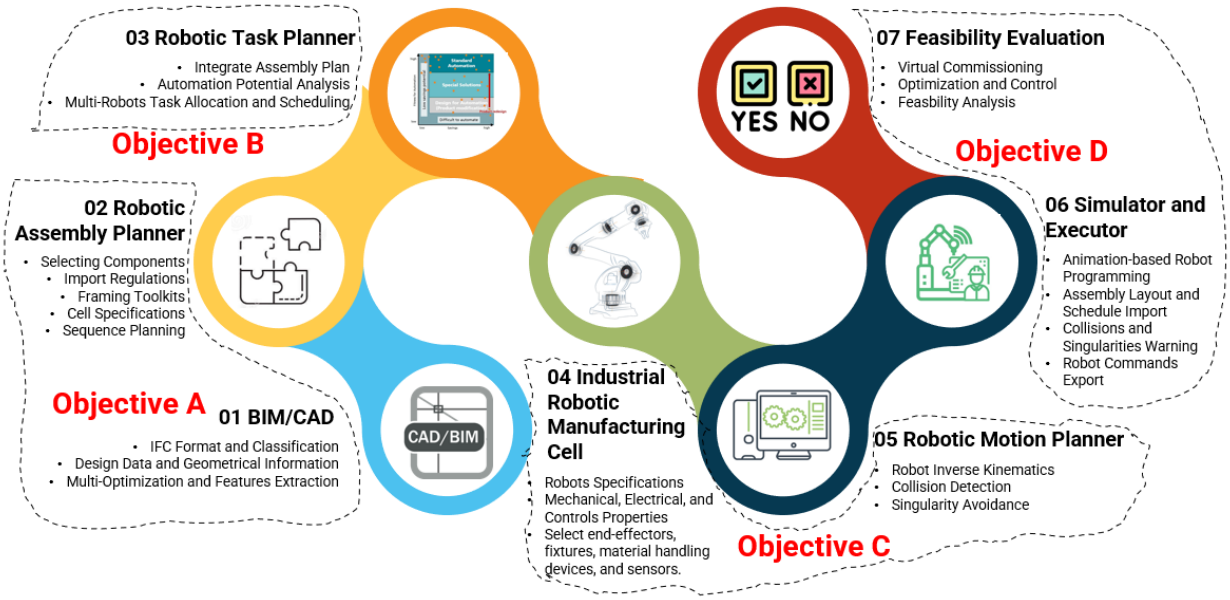


Figure 20. Research Framework

2.7 Modelling a Robotic Work Cell

The advancement of industrial robotics has been significantly shaped by technical, economic, and social factors, becoming a critical element of economic growth in industrialized nations. In the business world, companies face challenges such as shortening product lifecycles, reducing

production costs, maintaining high quality, and ensuring timely delivery while competing globally. This requires rapid and flexible responses, often through the adoption of capital-intensive technologies like robots and manipulators to enable continuous operation.

The increasing use of robotics in industry is driven by their ability to perform monotonous or precision tasks beyond human capability, with some achieving accuracy to the thousandths of a millimeter. The widespread availability and evolution of industrial robots have led to their adoption across various sectors, resulting in systems that offer greater flexibility, automation, and ease of reconfiguration than traditional manufacturing setups.

This evolution presents challenges for robotics engineers in designing and modeling manufacturing systems that integrate seamlessly and operate smoothly. The complexity of these systems necessitates models that cover construction and operational aspects from the design phase. A production plant includes numerous subsystems influencing productivity, efficiency, and process synchronization, highlighting the need for innovative solutions in designing, modeling, and simulating robotic production work cells. Such designs must support co-processing and self-organization, emphasizing enhanced autonomy, particularly in reconfiguration capabilities and context awareness.

Multiagent approaches offer promising solutions for designing robotic manufacturing work cells, integrating distributed knowledge resources. Within technical means designing, methodologies vary widely in process structure and the principles for setting design requirements (Gwiazda, 2013). An integrative approach to structural form design is shown in Figure 21(a), applicable to complex technical design fields like robotized production work cells.

In adapting the design and simulation of robotized production work cells, the framework shown in Figure 21(a) evolves into a specialized scheme illustrated in Figure 21(b), highlighting the integration of various subsystems that form the structure of a robotized production work cell. Specifically, the robotized production system comprises three interconnected subsystems:

- The main components subsystem, including industrial robots, auxiliary devices, end-effectors, etc.
- The kinematic and dynamic dependencies subsystem, covering configuration ranges, component velocities, and motion types.
- The spatial organization subsystem, detailing the layout and arrangement of elements within the work cell.

The design process emphasizes maintaining system integrity. This comprehensive methodology, encompassing various design branches, can be modelled as a multi-agent optimization task, where each design aspect (geometrical, material, dynamic features) is managed by an independent yet collaborative agent focused on constructing the final robotized work cell design. This agent-based system architecture with broad objectives is refined as the system is decomposed into smaller, semi-autonomous agent-based subsystems, enhancing process efficiency and autonomy.

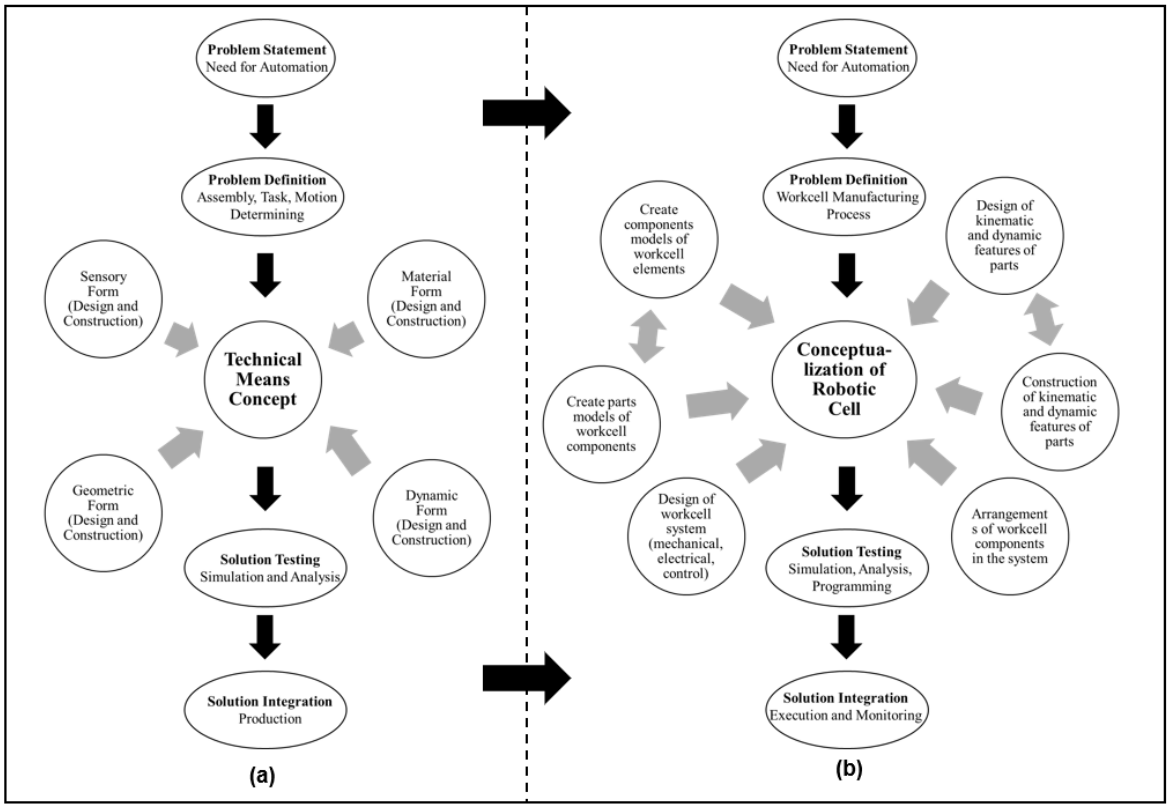


Figure 21. (a) Flowchart of Technical Means Concept (b) Robotic Workcell Design Process

2.8 Simulating a Robotic Work Cell

Simulation is a technique for conducting numerical experiments using dynamic models that represent either existing or envisioned systems, primarily aimed at providing insights into system behavior over time (Monica, 2015). Applied to developing a robotized workcell, simulation adheres to mechatronic design principles, enhancing understanding of the system's performance by integrating mechanical components with actuators, sensors, and digital information processing technologies (Wittler and Moritz, 1998). This approach considers geometric configurations, physical-topological properties, and the mathematical relationships between components. The mechatronic design process involves several simulation types: kinematic, dynamic, and those assessing mechatronic functions (Lückel and Wallaschek, 1997), each contributing to a comprehensive understanding of system capabilities.

The competition between continuously evolving offline and online software significantly enhances robotic system performance. Simulation has become vital in modeling robotized workcells, with many robot manufacturers offering virtual platforms that facilitate robot programming and comprehensive 3D simulations mirroring actual controller cycles. The process of robot modeling and simulation includes several key stages (Cheng, 2000; Aguiar and Silva, 2008; Grajo et al., 1994):

- Creation of robot and workcell component models,
- Configuration of components within the workcell,
- Specification of machinery and device kinematics,
- Establishment of robot movement paths, and
- Evaluation and validation of the developed model.

Initially, it's essential to define the characteristics and interactions of system components. The first three stages focus on building a digital workcell by selecting models, positioning elements, and detailing kinematics. The fourth stage sets robot trajectories, including collision detection and necessary adjustments. These outcomes inform an internal preprocessor that visualizes activities within the modeled workcell. Adjustments are made based on simulation performance, and upon satisfactory results, a postprocessor creates the robot's operating program. This program is then uploaded to the robot for testing and calibration, with the cycle repeating if results are not as expected.

Chapter 3 : Innovative Approaches in Assembly Sequence Planning: A Review of Soft Computing and AI Methodologies

Abstract: Assembly Sequence Planning (ASP) is an NP-hard challenge that demands an optimal sequence for product assembly, a task that's increasingly complex for sophisticated products. Traditional methods diminish when large combinations are involved, necessitating the adoption of advanced computational strategies for efficient optimization. This critical problem has taken the attention and interest of experts across computer science, engineering, and mathematics, propelling a surge in soft computing research aimed at refining ASP solutions. This comprehensive review synthesizes the collection of soft computing literature dedicated to ASP, serving as a crucial foundation for future scholars in the field. It analyzes various modelling approaches, optimization algorithms, and objectives, drawing insightful pathways for high-impact studies. It underscores the necessity to model flexible components within ASP frameworks and to integrate sustainable and ergonomic considerations into manufacturing processes. The exploration of innovative optimization algorithms that deliver optimal solutions within a feasible computational timeframe is also highlighted. In parallel, the rapid advancements of artificial intelligence (AI) techniques have revolutionized the solution of complex engineering issues. ASP stands out as a critical combinatorial puzzle that industrial engineers are keen to decode, aiming to decrease manufacturing expenses by minimizing assembly duration and resource expenditure. The expansive search space and numerous assembly predicates have fueled the application of AI, leading to an intensive review of various AI methodologies for ASP. This review has shed light on existing limitations and furnished a forward-looking perspective for researchers to employ diverse AI strategies. The goal is to master ASP by satisfying an array of assembly predicates, thus reaching the vertex of manufacturing efficiency.

3.1 Introduction

In industrial manufacturing and construction, maintaining accurate assembly is crucial for product uniformity and quality, as critical for economic competitiveness (Youhui et al., 2012). Component assembly, constituting up to 50% of total production costs (Kalpakjian and Schmid, 2006; Li et al., 2015), is optimized through ASP. ASP, a complex non-deterministic polynomial-time (NP)-hard problem (Wang, 2010), focuses on creating efficient assembly sequences within defined constraints to reduce product cycle time and facilitate smooth transitions from prototype to production (Rashid et al., 2011).

ASP involves determining the optimal sequence from potentially factorial ('n!') combinations, making it both difficult and time-intensive due to specific assembly constraints (Bahubalendruni and Biswal, 2016; Hsu, 2002). Proper modeling of these constraints is vital for the viability of assembly sequences, which aims to reduce production costs and enhance lead time (Kumar, 2015). Assembly predicate testing utilizes data such as liaison information, geometric and mechanical feasibility to ensure stability and determine optimal assembly sequences.

Advancements in CAD have simplified automatic assembly sequence generation, integrating collision detection and geometrical feasibility (Wilson and Latombe, 1994; Mok et al., 2001; Pan et al., 2006; Li et al., 2012; Alfadhlani et al., 2011; Giri and Kanthababu, 2015; Yu and Wang, 2013). Despite progress, achieving a globally optimal assembly sequence is challenging due to potential premature convergence of AI techniques (Ghandi and Masehian, 2015; Guo et al., 2015; Ibrahim et al., 2015). To address this, Hybrid Algorithms (HAs) combine classical algorithms to optimize assembly sequence outcomes.

This research segments into three phases: product development, production planning, and manufacturing, focusing on ASP and Assembly Line Balancing (ALB) as illustrated in Figure 22.

ASP determines the optimal assembly sequence based on a multi-criteria fitness function, while ALB ensures equitable task distribution across workstations. Optimizing these processes is crucial for reducing manufacturing costs and enhancing product quality (Lu, 2016).

The study explores the relationship between AI techniques and ASP resolution, analyzing various AI methods (e.g., Genetic Algorithm (GA), Ant Colony Optimization (ACO), Simulated Annealing (SA), Particle Swarm Optimization (PSO), Artificial Neural Network (ANN), Artificial Immune System (AIS)) for ASP optimization. It offers a comprehensive review of literature on ASP, exploring its description, modeling, constraints, and optimization objectives, and evaluates the performance of different AI techniques in solving ASP challenges. The research concludes with a summary of findings and performance evaluations of the algorithms, providing insights into their effectiveness in ASP optimization.

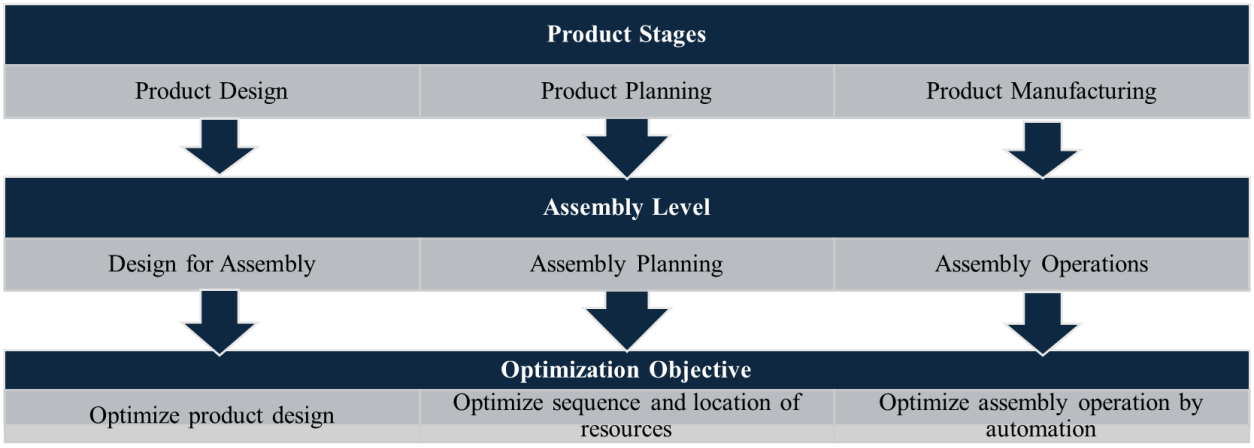


Figure 22. Assembly-Levels in Different Product Stages

3.2 Literature Review

3.2.1 ASP Overview

ASP is a crucial component in assembly planning that influences layout design, resource allocation, efficiency, and cost-effectiveness in product development. Automating ASP to generate

and optimize assembly sequences is crucial for enhancing product competitiveness and profitability (Marian et al., 2006). ASP involves calculating the motion sequences necessary for assembling components into a final product, considering variations in geometry, precedence, accessibility, and other constraints (Ghandi and Masehian, 2015; Choi et al., 2008).

Traditionally, assembly sequences are determined by the expertise of designers and engineers, but this approach often fails to identify the optimal sequence, especially as the number of components and potential sequences increase exponentially, posing challenges in error-free sequence formulation (Li et al., 2013). ASP is an NP-hard problem, where solving it with exact methods in polynomial time is impractical for large problem sizes (Wang et al., 2014). The complexity of ASP, particularly for products with many parts, has spurred research into computer-aided assembly planning.

Assembly can be linear or parallel, with parallel assembly involving the simultaneous joining of sub-assemblies. ASP significantly impacts manufacturing time and costs, as illustrated in Figure 23, which detail the classification of AP.

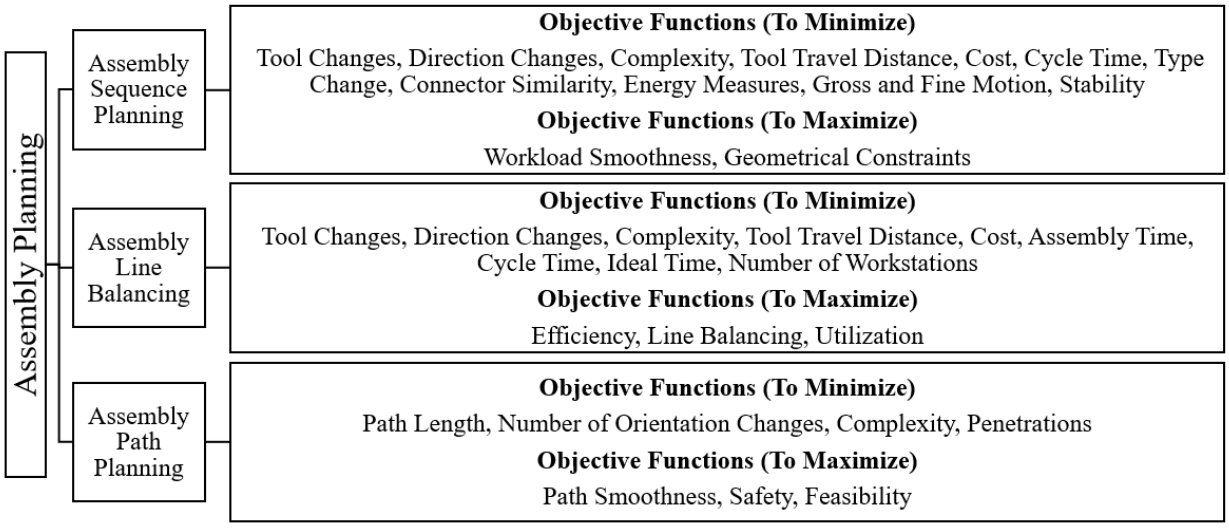


Figure 23. Assembly-levels Classification and Objective Functions

3.2.2 ASP Problem Modelling

AP is crucial in industrial manufacturing and construction, affecting product layout design, resource allocation, efficiency, and cost-effectiveness. Effective assembly planning, particularly ASP, is vital for reducing manufacturing costs and enhancing product competitiveness (Chryssolouris, 2006). ASP involves optimizing the sequence in which product components are assembled, considering variations in component geometry, precedence, accessibility, and other constraints (Bourjault, 1984; Choi et al., 2008).

Traditionally, assembly sequences are determined by the expertise of designers and engineers. However, with the increase in assembly components, potential sequences grow exponentially, necessitating a systematic approach to ASP, which is classified as an NP-hard problem due to its complexity and the vast number of possible sequences (Wang et al., 2014; Ghandi, 2015).

Research has developed several ASP modeling approaches, categorized into part-based, task-based, and connector-based modeling (Abdullah et al., 2019). Part-based modelling, the most common approach, as shown in Figure 24, focuses on the assembly's fundamental unit—the individual part—and involves creating precedence graphs based on a fixed or flexible base part (Suszyn´ski et al., 2014; Liu et al., 2013). Task-based modelling represents ASP through assembly tasks, setting forth assumptions about the combinations and movements involved in assembly tasks (Rashid et al., 2011). Connector-based modelling organizes assembly components by fixtures into sets, requiring a multi-step optimization process that includes determining the sequence of fixtures and sequencing the components within each fixture (Tseng and Tang, 2006; Su et al., 2014).

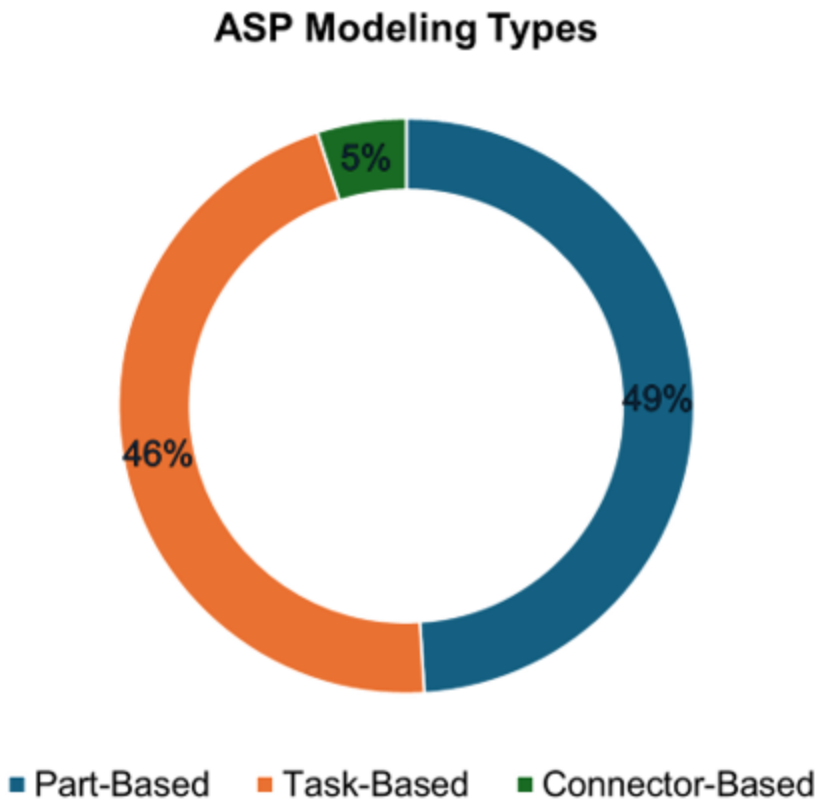


Figure 24. Modelling Approaches for the ASP Problem

The complexity of ASP has prompted the use of computer-aided tools and AI techniques to derive optimal assembly sequences efficiently. Traditional methods like knowledge-based approaches and cut-set methods relied on liaison data and precedence diagrams (De Fazio and Whitney, 1987; Baldwin et al., 1991) as presented in Figure 25 (T1 for tool changes and directional indicators like -X, +X, +Y, -Y). Modern AI techniques offer the potential to minimize computational demands and optimize assembly sequence generation. These methods are incorporated into an illustrative example, as shown in Figure 26, which includes part categorization, sequence generation, and station layout optimization.

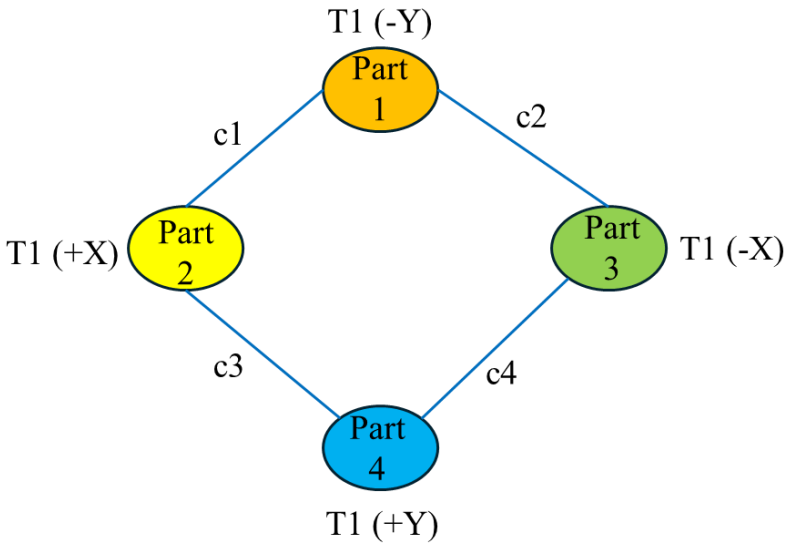


Figure 25. Precedence Diagram

This research presents a detailed analysis of ASP optimization through AI methods, covering its description, constraints, and optimization objectives. It explores various AI techniques used in solving ASP challenges, including their advantages and limitations. The study concludes with a review of performance evaluations of these algorithms, highlighting their effectiveness in enhancing assembly sequence planning.

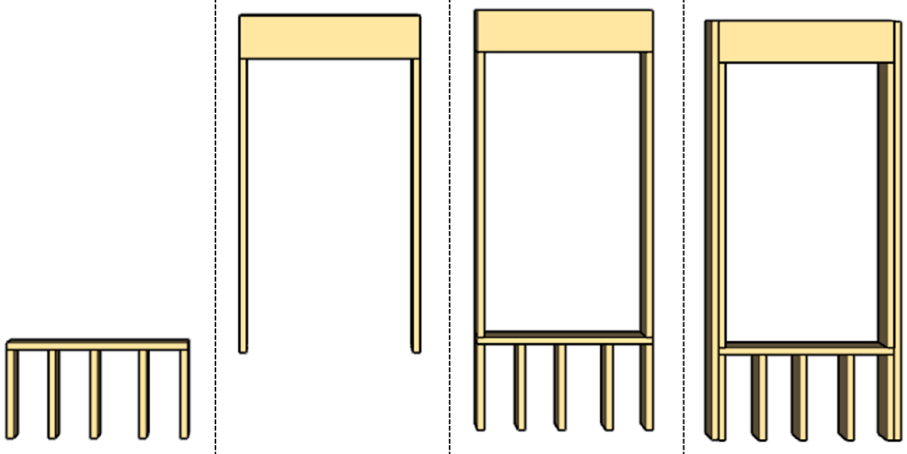
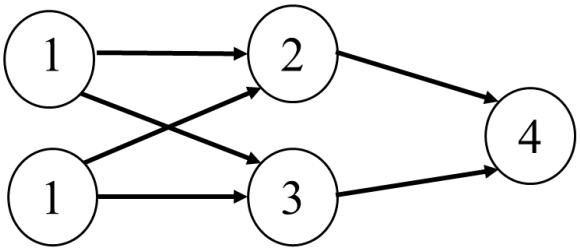
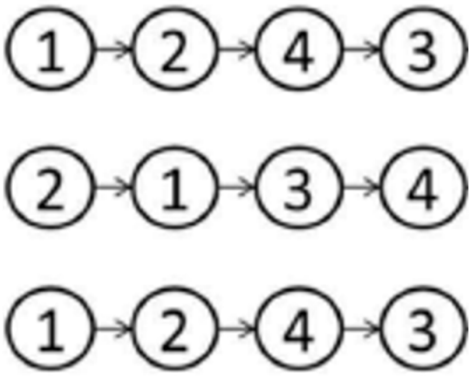
Window Sub-Assembly	Sequence 1	Sequence 2	Sequence 3	Sequence 4
Framing Tasks Description	Cut and nail the windowsill plate to cripples	Cut and nail the window header to jack studs	Combine and nail the window top part to lower part	Attach and nail the window interior part to king studs
Framing Tasks Sketch				
Assembly Precedence Graph				
Three Potential Sequences Derived from the Precedence Graph				

Figure 26. Example of Assembly Sequence for Window Sub-Assembly

3.2.3 ASP Constraints

In ASP, there are two main types of constraints: absolute and optimization constraints. Absolute constraints must be met for an assembly sequence to be feasible; if violated, they make a sequence infeasible. These include precedence and geometrical constraints, which cover the following areas (Rashid et al., 2011):

- Liaison Data: Information about contact points between parts.
- Stability Data: Details on part stability during assembly.
- Geometric Feasibility Data: Directions parts can be assembled without interference.
- Mechanical Feasibility Data: Whether parts can be joined in the presence of other components.

Geometric constraints, critical in ASP, define spatial relationships among components and ensure all valid assembly sequences satisfy these spatial conditions (Chen and Liu, 2001). They include dimensions such as distances, angles, and relative positions, validated through coordinate computations. Geometric constraints are further categorized into Constraint Direction (CD), Constraint Assembly State (CAS), and Minimal Constraint Assembly State (MCAS), which assess assembly feasibility through intersection matrices and Boolean operations for interference conditions (Su and Lai, 2010; Sinanoglu, 2005).

Precedence constraints dictate the order of assembly operations, derived from standard mechanical assembly principles. These constraints confirm the validity of assembly sequences by considering the geometric relationships of parts, stability of operations, and feasibility for robotic handling. A feasible assembly sequence must adhere to these precedence constraints, which are often extracted through a reasoning process that establishes each part's precedence relationship based on a user-defined sequence (Yuan, 2002). To effectively represent these relationships, a Precedence Matrix

(PM) is used, detailing the precedence relations between parts within the assembly (Zhang et al., 2010).

3.2.4 ASP Optimization Approach

The primary goal of ASP is to identify the most efficient assembly sequence for a product, particularly when multiple parts result in numerous potential sequences. Selecting the best sequence is influenced by factors such as tool and directional changes and can be time-consuming due to the complexity and multitude of options (Mok et al., 2001; Pan et al., 2005). The choice of sequence is further affected by product type, machinery availability, and market demand, with the aim to minimize assembly time and cost. Different feasible sequences may offer trade-offs between these factors, leading to research into multi-objective optimization techniques, which have evolved from single-objective methods to hybrid approaches for optimal sequencing (Figure 27). Optimization constraints play a crucial role in this process, as their violation can degrade the quality of sequences. These constraints are essential in developing fitness equations to assess sequence quality (Figure 28).

Integrating ASP with product design helps reduce assembly costs by facilitating early design modifications, minimizing the need for significant later investments. However, sequences from the concept phase may not be optimal, increasing time and cost. Design for Manufacturing and Assembly (DFMA) principles aim to minimize these costs by necessitating optimal assembly sequences for timely market release (Boothroyd et al., 2002; Kuo et al., 2001).

Initially, research focused on establishing at least one feasible sequence for part assembly, but AI algorithms have since enabled faster achievement of optimal sequences. Traditional methods involved manually extracting assembly constraints, a tedious process especially for complex products (Bullinger and Ammer, 1984). Automated extraction of liaison data has improved,

enhancing sequence quality and applicability (Mathew and Rao, 2010; Bahubalendruni et al., 2016).

Meta-heuristics are commonly used to optimize ASP, providing adaptable solutions to various problems with minimal adjustments (Kuo et al., 2001). These algorithms reduce computation times but do not guarantee optimal outcomes, yet they deliver acceptable solutions for large-scale problems (Prenting and Battaglin, 1964).

ASP research has employed various meta-heuristic algorithms like GA, ACO, and PSO, among others. Figure 29 tracks ASP algorithm research from 2000 to 2022, showing ACO and GA as most prevalent. Less commonly used algorithms like Gravitational Search Algorithm (GSA) and Frog Leaping Algorithm (FLA) tend to have poorer search capabilities and a higher propensity for local optima (Smith et al., 2001).

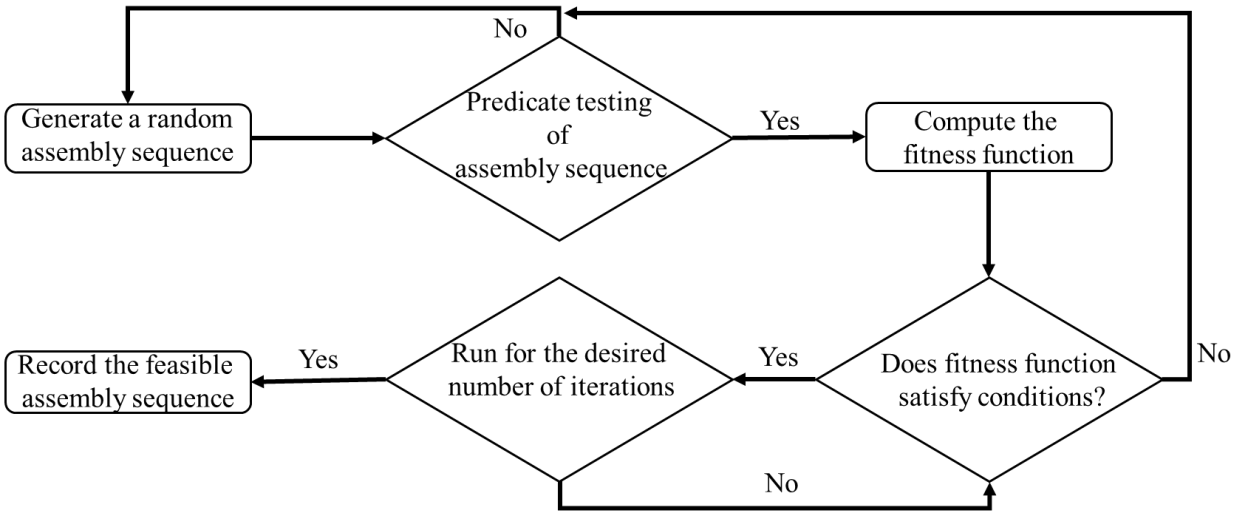


Figure 27. Basic Optimization Algorithm Flowchart

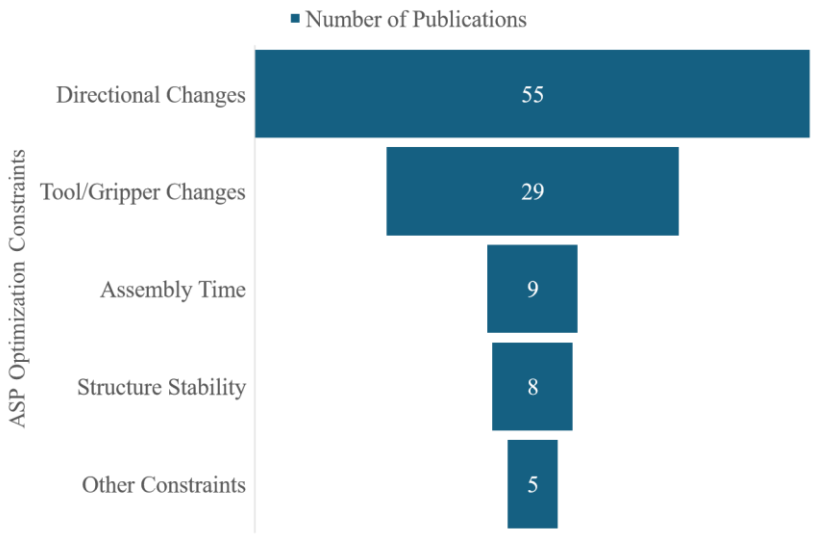


Figure 28. ASP Optimization Constraints Extracted from the Cited Literature

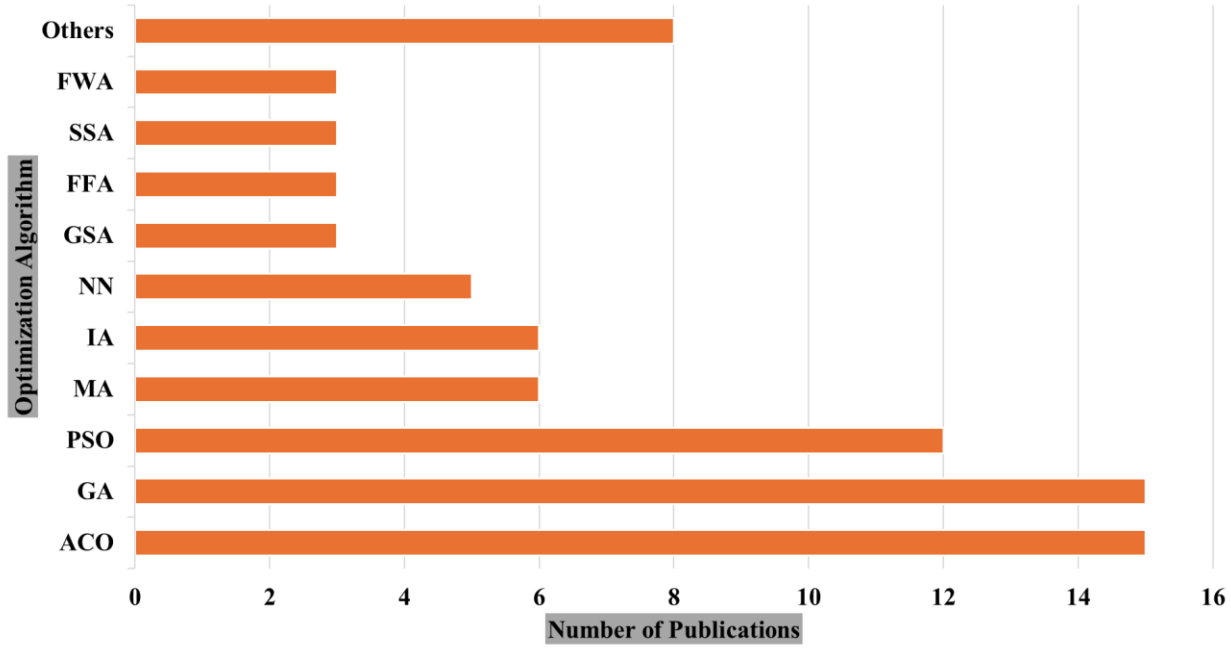


Figure 29. Frequency of ASP Algorithms in Cited Literature

3.2.5 ASP Optimization Algorithms

Genetic Algorithm (GA): GA, part of evolutionary algorithms, is designed to find optimal or near-optimal solutions for search problems, inspired by natural selection and genetics. Introduced by Johan Holland in 1975, GA operates on the principle of "Survival of the Fittest." The process starts

by generating a random population of potential solutions, applying crossover and mutation operators, and evaluating fitness, eliminating weaker solutions over iterations until an optimal solution emerges.

GA has been adapted to improve its effectiveness in solving the ASP problem. Innovations in GA for ASP include real number coding by Senin et al. (2000), enhancements to the Partially Matched Crossover (PMX) by Lazzerini and Marcelloni (2000) for robotic assembly optimization, and the incorporation of AND/OR graphs and new task re-ordering operators by Valle et al. (2003). Later adaptations include automatic precedence data extraction by Marian et al. (2003) and connector-based approaches by Tseng et al. (2004). Hui et al. (2008) and Choi et al. (2009) explored different crossover operators for optimal disassembly and multi-criteria ASP.

A comprehensive review of GA's application in ASP (Arunkumar et al., 2016; Kumar, 2015; Mahmoodabadi and Nemati, 2016) indicates that while GA is effective for many applications, its standard form struggles with complex combinatorial problems like ASP. Despite these challenges, GA remains a favored choice for its simplicity, global perspective, and adaptability, particularly effective in large-scale problems where constraints are integrated into the fitness function. However, its effectiveness reduces as the size of the assembly problem increases, often yielding near-optimal solutions for smaller-scale projects (Yasin, 2010; Mathew and Rao, 2014).

Ant Colony Optimization (ACO): ACO is an algorithm designed for optimization, drawing inspiration from the searching traits of ants. It is commonly used to address ASP problems. ACO simulates the way ants discover the shortest path to food by following pheromone trails—chemical markers that become more pronounced with repeated use This biological concept has been translated into a mathematical model to form the basis of ACO, leveraging pheromone trails to guide the search for optimal solutions.

ACO has evolved significantly since its initial application to ASP. Early implementations, like those by Sharma et al. (2008), utilized straightforward ACO techniques based on disassembly rules but suffered from slow convergence rates. Later advancements by Shi et al. (2010) and Wang et al. (2016) introduced dynamic adjustments to pheromone levels to improve convergence speeds and avoid local optima. Lu and Zhuo (2016) successfully combined ACO with ALB to enhance sequence optimization.

Further refinements were made by Yu and Wang (2013), who incorporated the Max-Min ant colony system to boost performance by setting limits on pheromone trails and refining initial values. Despite these improvements, ACO still faces challenges, particularly with longer convergence times and the risk of suboptimal solutions due to pheromone evaporation, which can lead to premature convergence and local optima, as noted by Ping (2016) and Sivakumar and Elakia (2016).

Artificial Neural Networks (ANNs): ANNs emulates the human nervous system, acting as a signal processing system composed of interconnected nodes and links. ANNs are capable of parallel distributed processing, making them effective for mathematical optimization problems with notable speed due to their parallel processing capabilities. However, ANNs are not typically applied directly in ASP; instead, they are used primarily for data storage and sorting within assembly processes.

Early integration of ANNs in ASP was spearheaded by Chen and Pao (1993), who combined ANNs with a rule-based method to store assembly data from CAD environments and generate rules for deriving optimal assembly sequences. Hong and Cho (1993, 1995) used neural networks to process data from expert systems for sequence optimization. Sinanoglu and Rıza Borklu (2005) utilized ANNs to apply binary vector representations of assembly data to find optimal solutions. Chen et

al. (2008) advanced this approach by proposing a three-stage method integrating neural networks. A primary challenge with ANNs in ASP is the complexity of training the networks to ensure they produce high-quality outputs, adding to the intricacy of their application in assembly sequence optimization.

Simulated Annealing (SA): Simulated Annealing (SA) is an optimization algorithm inspired by the metallurgical annealing process, in which metals are heated and then gradually cooled to minimize defects and achieve a low-energy state, according to the Boltzmann equation. Introduced by Metropolis in 1953 to model the annealing process, SA simulates small random atom displacements that alter energy states: configurations move to a lower energy state if the energy change is negative, and to a higher state if positive, allowing the algorithm to tackle both maximization and minimization objectives.

Chen et al. (2008) used a basic SA approach to derive optimal assembly sequences by evaluating multiple fitness variables. Hong and Cho enhanced this by introducing a multi-echelon method that applies different ranking levels at various stages of the algorithm to optimize robotic assembly sequence planning, integrating ALB to address practical assembly scenarios (Hong and Cho, 1997, 1999).

The SA algorithm stands out for its simplicity and guaranteed convergence to a solution, making it particularly valuable for complex ASP problems involving numerous parts. However, its main drawback is its slow convergence rate, which can lead to lengthy computational times to achieve a solution.

Particle Swarm Optimization (PSO): PSO is a stochastic optimization method developed from the social actions of animal swarms, where multiple agents (particles) within a swarm seek the optimal

solution in a search space. These particles adjust their trajectories based on their personal best achievements and the best results found by their neighbors, propelled by random accelerations towards these targets. Despite its effectiveness, traditional PSO struggles with local optima in complex, high-dimensional problems like ASP.

To enhance PSO for ASP, researchers have developed adaptations such as redefining particle dynamics to avoid local optima (Yu et al., 2010) and introducing chaos to increase randomness and improve solution diversity (Wang et al., 2010). Tseng et al. (2011) adapted PSO for environmentally-friendly ASP, employing a unique coding system within a comprehensive assembly and disassembly framework.

While PSO is known for its simplicity, rapid convergence, and low memory needs, it faces challenges with parameter tuning and can suffer from premature convergence and stagnation (Lv and Lu, 2010; Liu et al., 2008). Innovative approaches like Chaotic Particle Swarm Optimization (CPSO) and discrete PSO (DPSO), where particles represent specific assembly sequences, have been introduced to mitigate these issues (Ibrahim et al., 2014).

Overall, PSO's global search capability and ease of implementation make it attractive for ASP, but it requires careful customization to overcome its limitations, especially in large-scale assembly scenarios where computational demands are significant.

Artificial Immune Systems (AISs): AISs are computational models in artificial intelligence inspired by the biological immune system's learning and memory functions. These systems mimic the immune response, where immune cells produce antibodies to combat infections, applying these principles to solve computational challenges.

In ASP, AIS algorithms have been employed to develop optimal assembly sequences. Cao and Xia (2007) introduced an Immune Optimization Approach (IOA) utilizing immune regulatory processes, clonal selection, inoculation, and immune metabolism to find optimal solutions. Similarly, Chang et al. (2009) used a clonal selection-based AIS algorithm for ASP, while Biswal et al. (2013) enhanced their immune-based ASP approach with clonal selection and affinity maturation techniques to refine assembly sequences. Despite their effectiveness, these AIS methods face challenges with increased computational demands, particularly when integrating physical connectors as key components in the assembly sequences.

Memetic Algorithm (MA): MA is an advanced evolutionary computational method that blends cultural units of information (memes) with simple agents to solve complex problems. It integrates evolutionary or population-based methods with local improvement strategies, ideal for optimization challenges in large search spaces.

In ASP, MAs have been adapted in several ways:

(1) Tseng et al. (2007) employed a binary tree algorithm to generate initial feasible sequence populations, optimizing these with a GA using PMX and guided mutation.

(2) Zeng et al. (2011) developed a multi-agent evolutionary algorithm that group evolutionary operators with competition and crossover with mutation where multiple agents collaborate towards a common goal.

(3) Gao et al. (2010) used MA with a "two-tuple" gene concept for chromosome construction to facilitate crossover and mutation processes.

Overall, MAs effectively combine local optimization and population-based methods, making them highly effective for navigating the vast and complex search spaces often encountered in ASP.

New Emerging Algorithms in Computational Intelligence: Researchers have explored both traditional and innovative algorithms to optimize ASP. Notable among the newer approaches are Breakout Local Search Algorithm (BLSA), FLA, GSA, and Imperialist Competitive Algorithm (ICA), each introducing unique strategies:

(1) **Breakout Local Search Algorithm (BLSA)** by Ghandi and Masehian (2015): This method iteratively refines solutions by adjusting variable values to improve fitness, effectively addressing complex engineering challenges.

(2) **Frog Leaping Algorithm (FLA)** by Guo et al. (2015): Inspired by frogs' food-searching behavior, FLA evolves solutions within subgroups (memeplexes) and uses global information exchange to leap towards optimal solutions.

(3) **Gravitational Search Algorithm (GSA)** by Ibrahim et al. (2015): Utilizing Newtonian physics, GSA models searcher agents as masses that attract each other based on gravitational forces, where heavier masses indicate superior solutions.

(4) **Imperialist Competitive Algorithm (ICA)** by Zhou et al. (2013): ICA simulates human social evolution to solve NP-hard problems like ASP, employing a mathematical and computational approach.

These algorithms, known for their precision, involve complex implementations and typically require specialized knowledge. While effective, they remain under continuous development and are part of a broader set of innovative approaches, including MA and AIS, which collectively advance the field of ASP.

Hybrid Algorithms: HA combine multiple optimization algorithms to harness their best features, enhancing solution quality in complex problems like ASP. These algorithms integrate the strengths of each component algorithm to optimize performance, as illustrated by several implementations:

(1) Chen et al. (2002): Introduced a hybrid GA using heuristic methods to generate initial feasible solutions for multi-objective ASP, employing various crossover techniques for optimization.

(2) Shan et al. (2009): Developed a hybridized SA and GA approach, using SA to refresh GA's initial population to prevent premature convergence and enhance convergence rates.

(3) Ning and Gu (2007): Merged ACO with GA, where ACO creates feasible sequences that GA optimizes using crossover and mutation.

(4) Shuang et al. (2008): Combined PSO with ACO to improve ACO's convergence rate, balancing exploration and exploitation.

(5) Tseng et al. (2008): Integrated ASP with assembly line balancing using a local search algorithm and GA, employing PMX crossover and an elite strategy.

(6) Zhou et al. (2011): Introduced a hybrid of Bacterial Chemotaxis (BC) and GA, enhancing solution accuracy through diverse bacterial movement behaviors and genetic operations.

(7) Li et al. (2013): Employed an Evolutionary Direction Operator (EDO) within their HA to generate superior population individuals, further optimized through PSO.

(8) Zhang et al. (2014): Created a hybrid immune and PSO-based algorithm, combining immunization concepts with PSO for enhanced fitness evaluation.

While powerful, HAs face challenges in efficiently integrating different algorithms and often require more execution time due to their complexity and the extensive search processes they entail. Nevertheless, HAs remain valuable for addressing the complexities of ASP, particularly in handling intricate assembly configurations.

Knowledge-Based System: AI methods for ASP often rely on detailed assembly attributes like precedence constraints, graphs, and matrices, but their application can be complex. Researchers have combined heuristic rules with these attributes to optimize sequence generation:

(1) Huang and Lee (1991): Used a "three-tuple" approach to extract essential assembly knowledge, enabling the deduction of optimal sequences.

(2) Dong et al. (2007): Developed a connection semantic-based assembly tree to represent each part connection with a connect-type, aiding in the visualization of part interconnections.

(3) Hsu et al. (2011): Introduced a three-stage optimization method that uses a combination of neural networks, the Taguchi method, and Response Surface Methodology to analyze inter-part relationships for sequence optimization.

(4) Zha et al. (1998): Utilized reasoning-type questions to generate feasible sequences, represented through Petri net modeling, facilitating the extraction of inter-part knowledge.

(5) Kashkoush and ElMaraghy (2015): Applied a knowledge-based mixed integer programming approach using data from similar product assemblies, using a master sequence as a template for new products.

(6) Belhadj et al. (2016): Focused on detecting sub-assemblies to identify optimal assembly sequences automatically, though without specifying how to sequence the entire product optimally.

These methods deliver high-quality solutions but can become complex with products containing many parts. Recent efforts aim to simplify ASP by identifying sub-assemblies to reduce assembly levels and save time, although this approach increases search space complexity, especially for intricate products.

3.2.6 ASP Optimization Objectives

In ASP research, various optimization objectives are set to refine the process. According to the analysis (referenced in Figure 30), the most common objective is to minimize the time taken to change assembly directions. Additionally, frequent objectives include minimizing the time for assembly tool changes and reducing the variety of assembly operations. Conversely, the least emphasized objectives in the cited studies are maximizing assembly production levels, minimizing overall cycle time, and reducing the number of workstations required. These objectives reflect a focus on enhancing efficiency and simplifying the assembly process in ASP research.

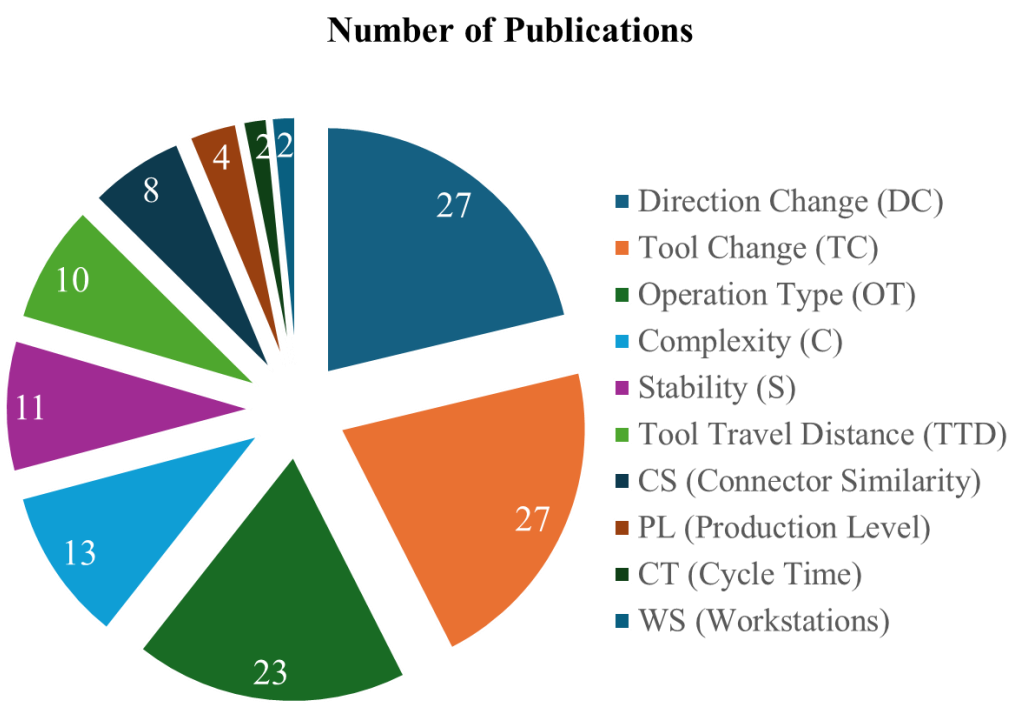


Figure 30. Frequency of ASP Optimization Objectives in Cited Research

(1) Minimize Assembly Direction Change (ADC)

In ASP, minimizing ADC is an essential optimization goal. This is crucial because assembly orientation, typically in six main directions (±x, ±y, ±z), significantly impacts efficiency and cost. Lu and Yang (2016) categorize assembly direction changes into two types: 180-degree changes (D_{1k}) and 90-degree changes (D_{2k}), with each change incrementing the respective count. Wang and Liu (2010) note that similar assembly tools, directions, and types are grouped to enhance efficiency and reduce costs. Meng (2016) highlights that fewer changes in assembly direction are preferable as they consume more time. Understanding and calculating these directional changes, often through an assembly interference matrix, is vital for optimizing the assembly process.

(2) Minimize Assembly Tool Change (ATC)

Minimizing ATC is a significant objective in ASP. Tools are categorized into four levels (T1 to T4) based on the difficulty of assembly tasks (Lu and Yang, 2016). The basic principle is to use the same tools and operation types for as many parts as possible within the same assembly direction. Frequent tool changes can increase the assembly sequence time, especially in complex product assemblies where specialized tools are used (Li et al., 2013). Assembly tools include both manual and robotic types, with decisions on tool choice being influenced by factors such as fastening times, part design features, accessibility, and operation methods (Mishra and Deb, 2016). Reducing tool changes is generally beneficial for the assembly process's efficiency.

(3) Minimize Assembly Operation Type (AOT)

In ASP various operation types like nailing, screwing, and picking are essential (Lu et al., 2006). Changes in these operation types can significantly affect assembly time and costs, often necessitating tool changes. In large-scale automatic assembly, most operations are robot-performed, but frequent reorientation and tool-grasping by robots can be challenging. Such changes in operation types, tools, and orientations can increase assembly energy and costs, making ASP a complex research area (Lv and Lu, 2009).

(4) Assembly Sequence Generation (ASG)

In optimal ASG, the primary goal is to minimize assembly time and cost while improving solution quality. This is achieved through an objective or fitness function that combines various factors, such as assembly directional changes, tool or gripper changeover, assembly fixation, cycle time, and effort usage. An example of such a fitness function could include components representing directional changes, tool changes, and operation type or equivalent assembly energy, as demonstrated in a specific equation format. This multi-faceted approach ensures a comprehensive

evaluation of the assembly process, balancing efficiency with quality. For example, in Equations 1 and 2, a fitness function for a given number of parts *n* consists of number of directional changes (DC), number of tool changeover (TC), and parts assembly effort (AE).

$$f = \sum_{i=1}^{n} DC_i + TC_i + AE_i \quad (1)$$

$$Fitness\ Function\ (FF) = \sum_{i=1}^{n} DC_i \quad (2)$$

3.3 Results and Discussions

In the area of ASP, researchers have focused on optimizing assembly efficiency using soft computing approaches over the past 17 years. A significant trend in ASP research, as illustrated in Figure 31, is the growing number of publications, particularly peaking in 2016, indicating sustained interest in the field. Key optimization algorithms like ACO and GA have gained attention due to their effectiveness in discrete combinatorial problems. However, challenges persist in rapidly achieving optimal assembly sequences, especially with complex products having numerous parts.

Recent research has highlighted the need for more comprehensive methodologies in developing feasible solution spaces for ASP problems. Integrating flexible parts in ASP, accounting for part deformations during assembly, is a growing area of interest but requires further development. Current research tends to focus on maximizing assembly efficiency, often overlooking crucial aspects like resource utilization, ergonomic factors, and sustainability in the planning stage. Recent introductions of new meta-heuristic algorithms like Grey Wolf Optimizer (GWO) and Ant Lion Optimizer (ALO) offer the potential for addressing discrete combinatorial challenges in ASP.

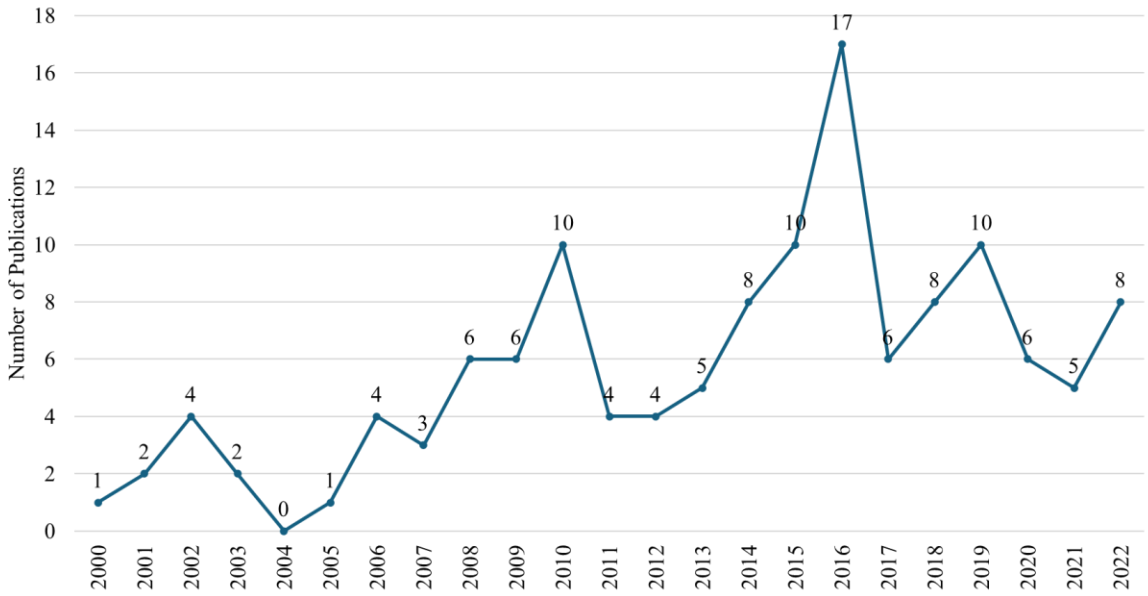


Figure 31. Number of ASP Publications Over Years (2000-2022)

Table 1. Summary of Advantages and Disadvantages of ASP Algorithms

ASP Algorithm	***Advantages***	***Disadvantages***
GA	Optimal for discrete optimization due to its comprehensive global search.	Utilizing binary strings and crossover-mutation are less effective for complex combinatorial challenges.
ACO	Second best choice for its user-friendliness and superior solution quality, ensuring consistent progression toward the solution.	Increased solution convergence time, making it less suitable for issues with time constraints.
NN	Ideal for parallel computation, it minimizes the risk of local optima due to simultaneous processing.	Constructing and training a neural network can be cumbersome, leading to complexities in its application.
SA	Straightforward to implement, adaptable for intricate problems.	Its main drawback lies in the risk of entrapment in local solutions and inefficient

		optimization for complex assemblies.
PSO	Execution involves fewer equations, simplifying the process.	Its partial optimization can be difficult for complex solutions.
AIS	Guarantees a global optimum by leveraging a parallel computation approach similar to NN.	Complex implementation makes it less practical for detailed assemblies.
MA	Prevents early convergence and is effective in securing the best solution.	May struggle with optimization, leading to local optima.
BLS	Natural for discrete optimization, offering simplicity.	Encounters difficulties with complex solution spaces, getting stuck in local optima.
FLA	Easier in comparison to other evolutionary algorithms, with fewer parameters to manage.	More time-consuming and challenging to implement for complex assemblies.
GSA	Simple to use, requiring minimal parameters and guided by a natural law principle.	Can be inefficient in complex solution spaces and local optimum challenges.

3.4 Conclusion

This research offers a comprehensive review of 24 years of research in ASP, analyzing 82 papers from various journals. It categorizes ASP modelling approaches into part-based, task-based, and connector-based. The review identifies the ACO, GA, and PSO as the most frequently used optimization algorithms. A common objective in ASP research is minimizing assembly direction change time, tool change time, and operation type changes. Future research directions are

highlighted, focusing on flexible part modelling, integrating ergonomics into ASP, and exploring new algorithms for sustainable manufacturing, emphasizing energy consumption in assembly planning. Also, future research should explore parallel assembly, integration of Design for Assembly (DFA) concepts, and comprehensive precedence criteria consideration to enhance ASP efficiency and quality. The study concludes that ASP research offers extensive opportunities for future contributions as it is considered a critical area in manufacturing, directly impacting time, cost, and product assembly quality.

Key findings include:

- GA is effective for assemblies with fewer parts.
- SA, ACO, and PSO are suited for medium-sized assemblies.
- HAs are best for large assemblies with many parts.

However, these AI algorithms face limitations in implementing them such as high computational time and local search space inadequacies, often leading to near-optimal solutions.

Chapter 4 : Integrated BIM-Robotic Assembly Sequence Planning Tool: Bridging the Gap from CAD to CAM in Offsite Construction Manufacturing

Abstract: The integration of artificial intelligence (AI) has significantly advanced various engineering fields, mainly in offsite construction manufacturing robotics for assembly sequence planning (ASP). ASP, a complex combinatorial challenge, seeks to minimize manufacturing costs by optimizing assembly time and energy consumption. This process requires accurate inputs from building information models (BIM), product details, and materials specifications. Classified as an NP-hard problem, ASP necessitates finding the optimal assembly order within a large search space and multiple criteria. AI techniques are increasingly favored over traditional methods for ASP due to their efficacy in handling complex products and vast search spaces. A comprehensive literature review reveals various AI methodologies for optimal ASP, highlighting the limitations of current approaches, particularly the reliance on heuristic algorithms focused on singular goals and the insufficient use of accumulated artificial experience and knowledge in real-world scenarios. To address these shortcomings, a new method leverages logic representation and rationalization, extracting and deducing information from IFC-structured BIM models using First and Second Order Logic (FOL and SOL). This approach develops algorithms that apply logical rules for data extraction and property deduction, implemented in Autodesk Revit's API - Dynamo. It assesses assembly sequence based on key indicators such as datum, neighbor, complexity, accuracy, and symmetry. The method has proven effective, with testing on unseen BIM models showing high precision, recall, and F1 measures. Its iterative nature allows continuous improvement and robustness through additional logic rules. Empirical results underscore the time efficiency and practical potential of this logic-enabled algorithm.

4.1 Introduction

The AEC industry, traditionally reliant on labor, faces a significant workforce shortage, compounded by an aging workforce and a high-risk work environment. This shortage has led to decreased productivity, project delays, and cost overruns, prompting a shift towards technological solutions like offsite construction and robotics automation to enhance efficiency and safety (Castro, 2009; Craveiro et al., 2019; Associated General Contractors, 2019; Bousquin, 2020; McGraw-Hill Construction, 2016; Pan et al., 2018).

Robotic automation in construction is growing, with predictions of substantial increases in robot usage by 2025. Robots excel in repetitive tasks within controlled environments like offsite construction facilities, potentially alleviating labor shortages and improving productivity, quality, and safety (Bock, 2015; Goodman, 2019).

OSC offers advantages such as controlled environment operations, concurrent task execution, and higher construction quality. However, the adoption of automation technologies faces challenges like high design and planning demands, high implementation costs, industry resistance, and workflow incompatibilities (Lawson et al., 2010; McGraw-Hill Construction, 2011; Alwisy et al., 2019; Davila Delgado, 2019; Buchli et al., 2018).

BIM has emerged as a crucial technology, facilitating the design, planning, construction, and operation phases. Despite its advantages, BIM's integration into digital workflows is limited, particularly in modeling complex structures and automating processes within offsite construction settings (Sacks et al., 2018; Kim et al., 2021). BIM's limitations necessitate manual data transfer and integration, imposing substantial costs and highlighting the need for enhancements to accommodate the complexities of offsite construction (Gallaher et al., 2004).

Recent studies have focused on developing algorithms to extract and derive facts from IFC-based BIM models using rule-based methods, aiming to automate the generation of robotic assembly sequences for offsite construction. This research fills a gap in the literature by investigating the transfer and utilization of information between digital building designs and robotic platforms, potentially advancing autonomous robotic assembly (Terada and Murata, 2008; King et al., 2014; Willmann et al., 2016; Ding et al., 2020; Tao et al., 2020; Zhang et al., 2022).

4.2 Literature Review

4.2.1 Automation in Offsite Construction

OSC in North America, known under various terms such as prefabricated, precast, and modular construction, involves the manufacturing and assembly of building components in a controlled environment before transport to construction sites (Goodier and Gibb, 2007; Lawson et al., 2014). Despite its growing popularity, the shift to OSC has not brought significant innovation in manufacturing techniques, often replicating on-site processes offsite with minimal automation (Altaf et al., 2018). OSC is categorized into panelized and modular construction, utilizing materials like concrete, steel, and wood, with specific types preferred for different building scales and purposes (Lopez and Froese, 2016).

OSC offers productivity gains, safer work conditions, cost and time efficiencies, standardization, and quality improvements. However, it faces challenges in automation due to the need for manufacturing flexibility and its traditionally labor-intensive nature, which impacts productivity and safety (Brissi and Debs, 2019; Kamath and Sharma, 2019). Large factory spaces and significant labor are required, with standardized production lines sometimes limiting design flexibility. The manufacturing sector under OSC faces pressures to increase production flexibility, diversify product offerings, and optimize space (Salvador et al., 2007; Meyer and Jacob, 2008).

Emerging automated systems and digital technologies are seen as crucial for OSC's transformation, particularly under Construction 4.0, aiming to integrate design, manufacturing, and assembly phases more effectively, as shown in Figure 32. However, OSC's adoption of these technologies trails other industries, hindered by low technology adoption and inadequate R&D investment (Leviäkangas et al., 2017; Liu et al., 2021; Orlowski, 2020; Aapaoja and Haapasalo, 2014). For instance, in the Canadian wood framing industry, the assembly stage is noted as a significant bottleneck, demanding substantial labor and impacting production efficiency (Van Ooteghem and Xu, 2012; Rodríguez et al., 2020). Despite a shift towards a customer-centered approach, assembly planning still requires considerable manual labor, impeding efficiency in the OSC process.

IR are increasingly significant in wood and steel construction fabrication due to their adaptability and capacity to produce complex building components, marking a shift from traditional machine functions to treating buildings and components as customizable products (Menges et al., 2015; Schindler, 2010). This evolution necessitates an integrated digital workflow that ensures seamless transfer of design, manufacturing, and robotic data.

In wood construction, automation technologies such as robotic arms and computer numerical control (CNC) machinery are critical. Robotic arms facilitate automated assembly and material handling, exemplified by "The Sequential Roof" project (Willmann et al., 2016), while CNC machines enable precise wood component shaping (Eversmann et al., 2017). Both rely heavily on accurate digital data from sources like BIM, which provides detailed digital representations essential for effective operation.

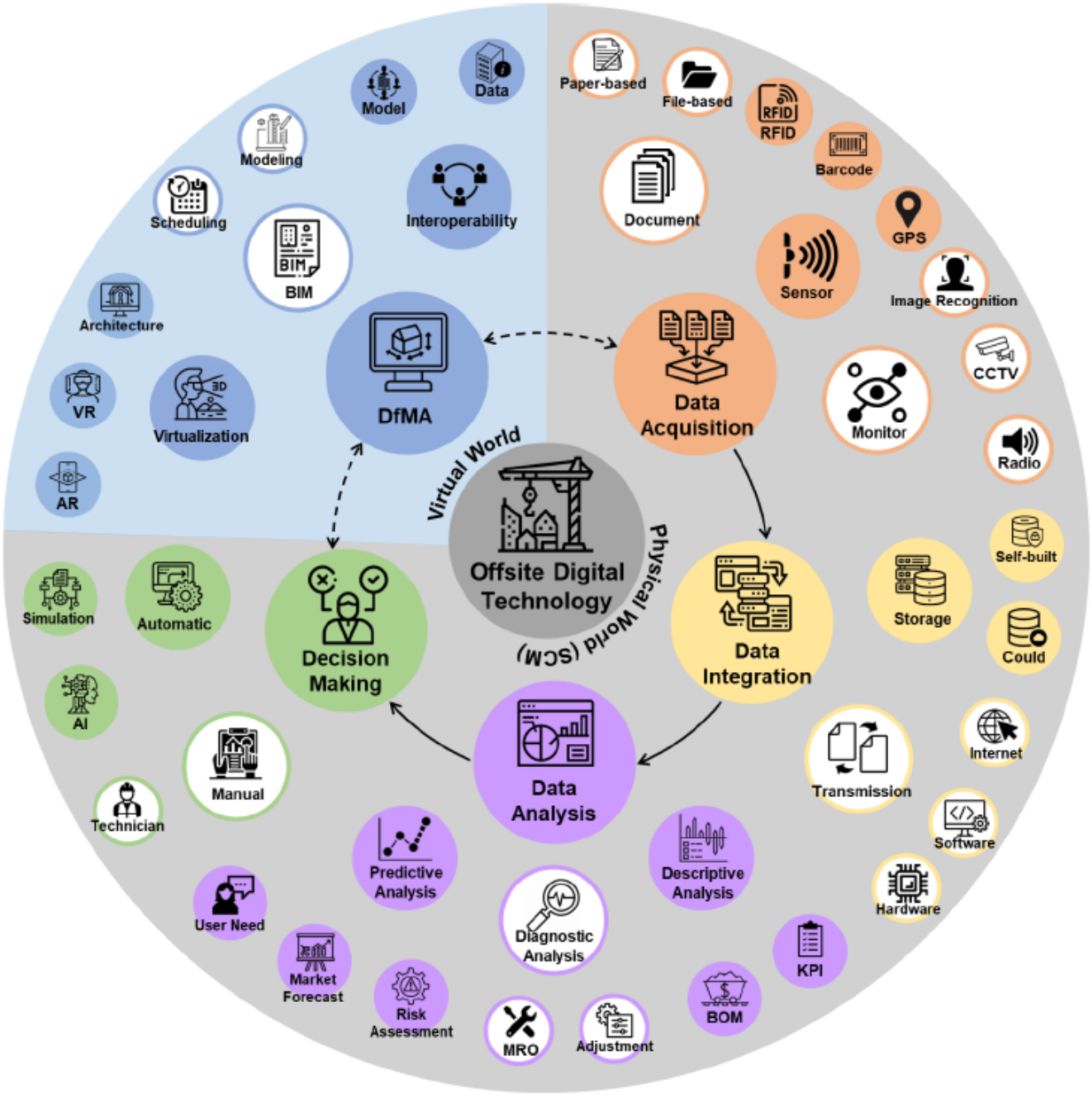


Figure 32. Digital Technologies for OSC

OSC is adopting a semi-industrialized approach where prefabricated components are treated like manufactured products, integrating manufacturing technologies into building prefabrication, especially for residential structures (Ramaji and Memari, 2016). This integration offers efficiencies over traditional onsite construction but faces limitations in analyzing and interpreting building designs directly, particularly during the design development phase.

A key challenge is the inefficient data transfer between BIM systems and automation technologies, which inhibits seamless integration (Sacks et al., 2018). This disconnect requires the development of more sophisticated systems and protocols to bridge the gap between digital design information and automated manufacturing processes, aiming to streamline and enhance the efficiency of workflows from design to construction.

4.2.2 Role of BIM in Offsite Construction Manufacturing

Scholarly efforts have focused on enhancing OSC through BIM-based methodologies, primarily improving design, planning, and optimization. Liu et al. (2018) created a rule-based algorithm using an automated BIM strategy to reduce waste in light-frame boarding design, implemented as a Revit add-on. This method reduces errors and manual modeling time with contractor expertise. Alwisy et al. (2019) automated the design and drafting process of wood light frame wall panels using Visual Basic within AutoCAD, transforming 2D CAD layouts into BIM models and shop drawings, aiming to reduce design costs and improve accuracy and productivity. Abushwereb et al. (2019) introduced FrameX, an Autodesk Revit add-on to facilitate the automated analysis, modelling, design, and drafting of light-frame wood structures, enhancing efficiency and accuracy in early project stages.

Expanding beyond design, Isaac et al. (2016) utilized a graph-based approach guided by BIM designed models to refine OSC designs, aligning them with client expectations and reducing delays and costs in modular housing prefabrication. Jensen et al. (2012) integrated manufacturing CAD software like SolidWorks with TactonWorks Studio into the workflow to automate module configuration selection, reducing stakeholder conflicts and inconsistencies in prefabrication. Mekawy and Petzold (2017) developed the Box Module Generator using Autodesk Dynamo for Revit, exploring design alternatives for Box Prefabricates.

Gbadamosi et al. (2019) introduced a method to refine design choices using lean principles and design for manufacturing strategies, integrating tools like Revit, Dynamo Studio, and Excel to minimize waste and improve assembly ease in exterior insulation finish systems. Wang et al. (2019) developed an automated framework utilizing BIM-based SQL database for bill of materials including quantity take-off for building components, reducing manual labor needs. Root et al. (2019) examined offsite techniques for producing structural exterior wall insulated and finished using Revit and Strucsoft's plug-in.

Despite these advances, most research relies on proprietary BIM platforms, particularly in planning and design phases, limiting BIM's full potential. Moreover, the scarcity of research focused on the construction phase presents challenges for implementing automation in OSC.

4.2.3 Industry Foundation Class (IFC)

Current strategies to enhance BIM models data exchange are focused on standardizing data schemas and defining the semantics of AEC objects, as highlighted by Poirier et al. (2014). Two important standards in this area are the CIMsteel Integration Standards (CIS/2) which been adopted by the American Institute of Steel Construction and the IFC for broader building, fabrication, and construction data. Both standards are developed to utilize the Standard for Exchange of Product (STEP) file format structure in accordance with ISO 10303 (Isikdag et al., 2007). The IFC, a vendor-neutral standard maintained by buildingSMART, is designed to enhance synergy within the AEC industry. It uses the EXPRESS data definition language and is organized into four conceptual layers—resources, core, interoperability, and domain—encompassing a wide array of data such as dimensions, properties, and relationships within a BIM given model (Arayici et al., 2018). This schema encompasses diverse building components such as beams, columns, and roofs, each represented through various geometric techniques like Swept Solid, Boundary Representation

(B-rep), or Body Clipping. Additionally, it provides multiple cross-sectional profile options for each element. For example, a beam could have a rectangular or arbitrary closed profile if it was represented using the Swept Solid method, showcasing the schema's flexibility to adapt to different BIM authoring tools' 3D modeling techniques. Figure 33 and Figure 34 further exemplify the complexity and capabilities of the IFC data model by detailing IFC instances, including entities, relationships, and a tracing pattern for the information of a column element.

BIM and IFC standards play a crucial role in improving data exchange and collaboration within the AEC industry by standardizing how building components and their relationships are represented. This standardization facilitates more efficient and accurate data sharing among stakeholders, potentially enhancing project outcomes and construction efficiency. However, challenges such as proprietary concerns and achieving true synergy remain significant obstacles to fully leveraging BIM benefits (Steel et al., 2012; Ren et al., 2018).

Despite its advantages, the IFC format faces challenges that affect its standardization. The flexibility of IFC allows for multiple representation methods for single elements, which can introduce subjectivity and inconsistencies. Additionally, user-defined property sets add unpredictability, and there are calls for improved descriptive capabilities and more robust element representations. These issues highlight the difficulty in achieving universally accepted standardization within the IFC schema.

Nevertheless, IFC is still recognized as a pivotal solution for integration issues in the AEC industry. Its openness and neutrality make it a key focus in BIM research and practice. As of 2020, the growing number of BIM platforms certified for IFC compliance (94 in total) reflects the industry's commitment to enhancing interoperability and standardization through IFC

(buildingSMART, 2020; Wu and Zhang, 2019). This demonstrates ongoing efforts to address these standardization challenges and enhance overall project efficiency in the AEC industry.

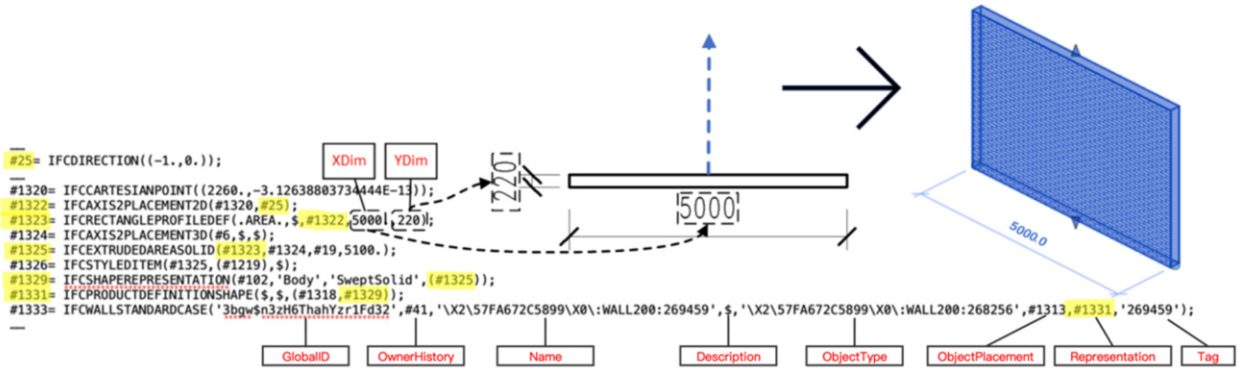


Figure 33. Sample of an IFC Instance

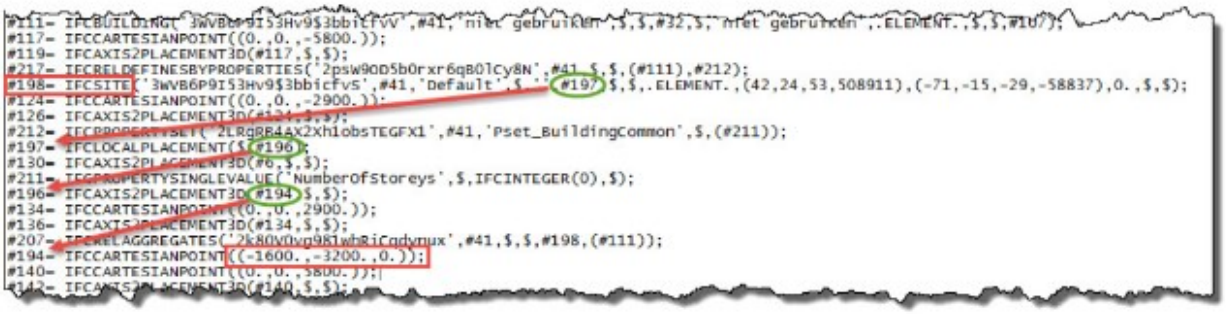


Figure 34. Tracing Pattern of Dimensions Information for the IfcColumn

4.2.4 Representation and Application of Rule-Based Method Using Logic Theory

Rule-based methods in machine learning and data mining are employed to identify patterns in data through IF-THEN rules, focusing on uncovering regularities within data sets (Fürnkranz and Kliegr, 2015). These methods are categorized into association rule discovery and predictive rule learning, with the latter aiming to assemble a comprehensive set of rules that cover all possible instances for accurate predictions.

The transparency and interpretability of rule-based methods make them particularly valuable in fields where the clarity of decision-making processes is essential. These methods structure decisions into clear, logical steps, allowing users to trace how conclusions are reached, thereby

building trust in the system's outputs. Additionally, rule-based systems offer adaptability, enabling them to be tailored to meet the specific complexity and requirements of various tasks, from simple classifications to complex decision-making in dynamic environments, optimizing performance across different scenarios.

4.2.4.1 A first-order theory

Logic is fundamental in computer science, particularly influencing the design and functioning of computer systems (Sterling and Shapiro, 1994). It manifests prominently in logic programming, where sets of rules articulate relationships among entities using formal logic, especially predicate logic (Johansson, 2011). This approach supports the formation and derivation of logical conclusions from articulated knowledge. A prevalent method in logic programming is First-Order Logic (FOL) which is known for handling quantifiers and variables effectively (Uszkoreit, 1986). FOL allows for complex expressions and has broad applications, including database queries and developing artificial intelligence algorithms.

4.2.4.2 First order logic and second order logic

Rule-based first-order logic is effectively used to represent IFC-based BIM data by converting structured BIM data into logical statements or predicates. This method allows for a dynamic interpretation of the data, articulating complex relationships and dependencies between model components clearly. It enhances understanding of the model by demonstrating how elements interact and improves automation and optimization in building design and construction processes. The generated predicates provide a strong basis for querying, rationalization, and decision-making within the BIM framework, thereby improving efficiency and accuracy in building projects.

FOL predicates are intertwined logically through the use of quantifiers and logical connectors. FOL utilizes both universal and existential quantification to construct statements regarding variables (Zhang et al., 2016). Moreover, logical connectors including conjunction (^), disjunction (∨), biconditional (↔), and implication (→) are employed to forge logical links between predicates (Zhang et al., 2016). Building on the foundations of FOL, Second-Order Logic (SOL) enhances expressive capabilities, allowing for the quantification of subsets of individuals or relationships within a domain (Väänänen, 2019). SOL plays a pivotal role in discerning all occurrences of building components that possess specific attributes within the framework of building design logic facts. Within the context of logic programming, SOL is queried by predicates like findall (Term, Goal, List), as explained by Sterling and Shapiro (1994).

4.2.4.3 Fundamental components of logic programming

Logic clauses in FOL are often expressed through Horn clauses (HCs). Horn clauses are named after the logician Alfred Horn, who first highlighted their importance in 1951. A Horn clause is a logical statement that links a single predicate to a set of conditions necessary for its application. These clauses are utilized in areas such as logical programming, formal specification, and model theory. The clause is interpreted such that the connecting predicate is applicable to X and Y if X and Z are friends, and Z and Y are friends. The predicate on the left is relevant to the variables X and Y only if all the predicates on the right are also relevant to X, Y, and Z. An HC is a disjunctive clause with at most one positive literal (Makowsky, 1987).

Denoted as $H \leftarrow (G_1 \wedge G_2 \wedge G_3 \cdots \wedge G_n)$, where H is the head and G_1, G_2, G_3, G_n are goals, it implies H is valid if all goals are met. HCs are simple yet expressive enough for general and efficient program representation (Pereira and Shieber, 2002). They are categorized into three types: facts, rules, and queries (Zhou, 1997).

In the fundamental architecture of logic programs, a HC can function as a fact, a query, or a rule. are statements describing the properties of objects or their relationships (Uszkoreit, 1986), comprising predicates that include a predicate name and its arguments. The term "arity" refers to the number of arguments a predicate has, with the most basic form, an atom, possessing just one argument. Queries are composed solely of goals and are employed for retrieving information, concluding about the relationships between objects. Rules feature both a head (H) and goals, and they are deemed true if all the goals are satisfied; otherwise, they are false.

4.2.4.4 Principles and mechanisms of logic rationalization

Using rule-based representation combined with first and second order logic theories significantly improves the analysis of architectural design data. This approach uses rule-based logic to autonomously rationalize IFC data through assembly rules, relying on unification functions and involving principles of deduction such as identity, generalization, and instantiation. The identity principle allows for querying logical facts to identify consequences; generalization applies logical outcomes to various cases with existential quantification; and instantiation derives specific facts from universally quantified ones. Supported by the unification algorithm, this automated deductive process enhances pattern recognition and variable binding, facilitating rationalization based on established logic rules (Sterling and Shapiro, 1994; Luger and Stubblefield, 2009).

4.2.4.5 Overview of the prolog programming language

Prolog, a declarative programming language, is distinguished by its basis in logic programming, making it especially suited for fields like AI, expert systems, and natural language processing (Ritchie, 2002; Covington et al., 1995). Unlike procedural and object-oriented languages that require explicit instructions, Prolog uses facts, rules, and queries to define relationships and employs its inference engine to deduce solutions. Operating under a closed-world assumption,

Prolog presumes statements not provable by available information as false, requiring comprehensive and logically structured data for accurate outcomes. Its problem-solving approach, which involves pattern matching and recursive querying, enables sophisticated data manipulation and decision-making, ideal for complex applications demanding logical inference and pattern recognition.

4.2.5.6 Utilizing rule-based representation in AEC automated rationalization systems

Rule-based representation is widely used in automated rationalization, allowing computing systems to draw inferences automatically, and is crucial across various fields including computer science, mathematics, and verification of software and hardware. It is integral to disciplines like sensing, natural language processing, and robotics, playing a key role in AI development by enabling intelligent systems to perform autonomously (Nehra, 2015). In the AEC sector, the use of AI technologies such as NN, GA, ML, and DL has increased, as shown by the rise in scholarly publications since the early 2000s (Darko et al., 2020). Despite the trend towards machine learning and statistical methods, rule-based systems are noted for their precision, comprehensiveness, and ability to form logical conclusions. However, they are relatively underutilized. The goal is to further explore integrating rule-based representation and rationalization to automate inference processes, particularly in the construction fabrication industry.

4.2.6.7 Analyzing differences between rule-based representation and ifcOWL ontology

Semantic modeling and rule-based methods are two distinct approaches used to enhance logical deductions and automated rationalization. Semantic modeling, particularly ontology using Web Ontology Language (OWL), organizes knowledge into concept hierarchies, relationships, and axioms, primarily aimed at capturing domain-specific semantics (OWL Working Group, 2012; El-Gohary and El-Diraby, 2010). Ontologies rely on rule-based descriptions like the Semantic Web

Rule Language (SWRL) for automation because they do not naturally support direct if-then logic (Zhang and El-Gohary, 2017).

Rule-based methods, on the other hand, have been integral to automated theorem provers and have successfully addressed complex scientific challenges, including verifying computer program accuracy and generating mathematical proofs (Dahn, 1998; Kotelnikov, 2018). Despite their effectiveness, rule-based methods are underutilized in AEC compared to statistical ML and semantic modeling.

To bridge IFC standard data with rule-based logic, the ifcOWL ontology was created, simplifying data exchange and enhancing interoperability (Pauwels and Terkaj, 2016; Gonzalez et al., 2021). However, utilizing the ifcOWL ontology faces several challenges:

1. **Complexity and Size**: Its large and intricate structure can be cumbersome to manage, especially in systems with limited computational resources (Pauwels et al., 2017).
2. **Performance Issues**: The detailed nature of ifcOWL can slow down loading and rationalization processes, leading to delays (Pauwels et al., 2017).
3. **Need for Additional Rule Representations**: ifcOWL often requires supplementary rule-based languages like SWRL, which may lack the expressivity and computational efficiency of more robust systems like FOL and SOL.
4. **Decidability Issues**: SWRL struggles with decidability issues crucial for determining the truth values of propositions within the ontology.
5. **Limited Expressivity**: The expressivity of SWRL does not match that of more complex logical frameworks, potentially limiting the modeling and inference of complex relationships within the ontology.

Despite these challenges, the ifcOWL ontology and rule-based methods continue to provide valuable tools for enhancing data rationalization and interoperability in the AEC industry.

4.2.5 Application of IR in Wood and Steel OSC Manufacturing

Despite growing interest in automation within offsite construction prefabrication, the sector faces challenges hindering the adoption of new technologies, including insufficient expertise, inadequate training, low standardization, high risk factors, significant initial investments, and prolonged lead times (Kyjanek et al., 2019). A particular issue is the low standardization of products, stemming from factors like poor collaboration, high variability, and inconsistent scheduling (Aapaoja & Haapasalo, 2014). Additionally, automated fabrication companies in the U.S. have reported minimal savings and significantly higher costs compared to traditional construction, mainly due to underutilization of advanced technologies (Wang et al., 2020).

The integration gap between robotic systems and BIM or comprehensive management systems leads to disconnects between design and production phases (Chea et al., 2020). Moreover, the deployment of standard robotic units in manufacturing poses safety risks, and necessary safety mechanisms can increase financial costs (Apolinarska et al., 2021). Furthermore, current automated or semi-automated mechanisms for wood framing require substantial space within construction facilities (Martinez et al., 2020). The machinery, mainly designed for repetitive tasks, requires costly and time-consuming reprogramming for new tasks (Bennulf et al., 2018). While BIM can store detailed information, it doesn't inherently provide manufacturing instructions, necessitating effective workflows from BIM to numerical control machinery (Santana-Sosa and Riola-Parada, 2018).

Advancements in robotics within wood construction have evolved significantly, transitioning from single-function robots to more integrated robotic systems in recent years. For example, the Robotic

Wood Construction (RTC) system developed by Willmann et al. (2016) at ETH Zurich, integrates a digital workflow from design to fabrication, utilizing innovative design processes, materials systems, and robotic fabrication to erect non-standard wood structures. Additionally, efforts like those by Kontovourkis (2017) showcase the potential of robotics in optimizing material use and advancing the fabrication of wood structures.

However, the reliance on proprietary CAD systems in these projects limits the standardization of workflows, contrasting with initiatives aiming to adopt open BIM standards like IFC, which promote a more unified and universally applicable strategy within the field.

4.2.6 Building Components Preliminary Assembly Plan

Prefabricated building elements had predefined spatial coordinates for robotic assembly yet lacked automated methods to determine the optimal assembly sequence. Consequently, human operators needed to manually program the assembly coordinates for each component. As product diversity and complexity increase, there is a growing need for the robotic assembly process of wood and steel elements to operate autonomously, without manual intervention. The main challenge is planning a logical sequence for assembling components that allows the robotic arm's end-effector to accurately position each element based on its designated coordinates. Addressing the challenge of quantitatively assessing and rationalizing the assembly sequence is essential to enhancing the efficiency of robotic assembly systems for prefabricated structures.

4.3 Research Methodology

The main goal is to develop a data-driven approach to improve the extraction and analysis of building design data from BIM, facilitating automation in OSC fabrication for the wood and steel industries. This approach uses rule-based representation and rationalization to automate BIM data processing. It aims to enhance performance through iterative refinement of the algorithm and

expansion of assembly logic rules. The methodology consists of six main stages: (1) Generation of Rule-based Facts, (2) Development of Assembly Rules, (3) Execution of Rule-based Rationale, (4) Assessment Metrics, (5) Stepwise Refinement of Rule-based Facts, and (6) Extension of Assembly Rules as described in Figure 35. Each stage is detailed with practical examples to demonstrate the functionality of the proposed method.

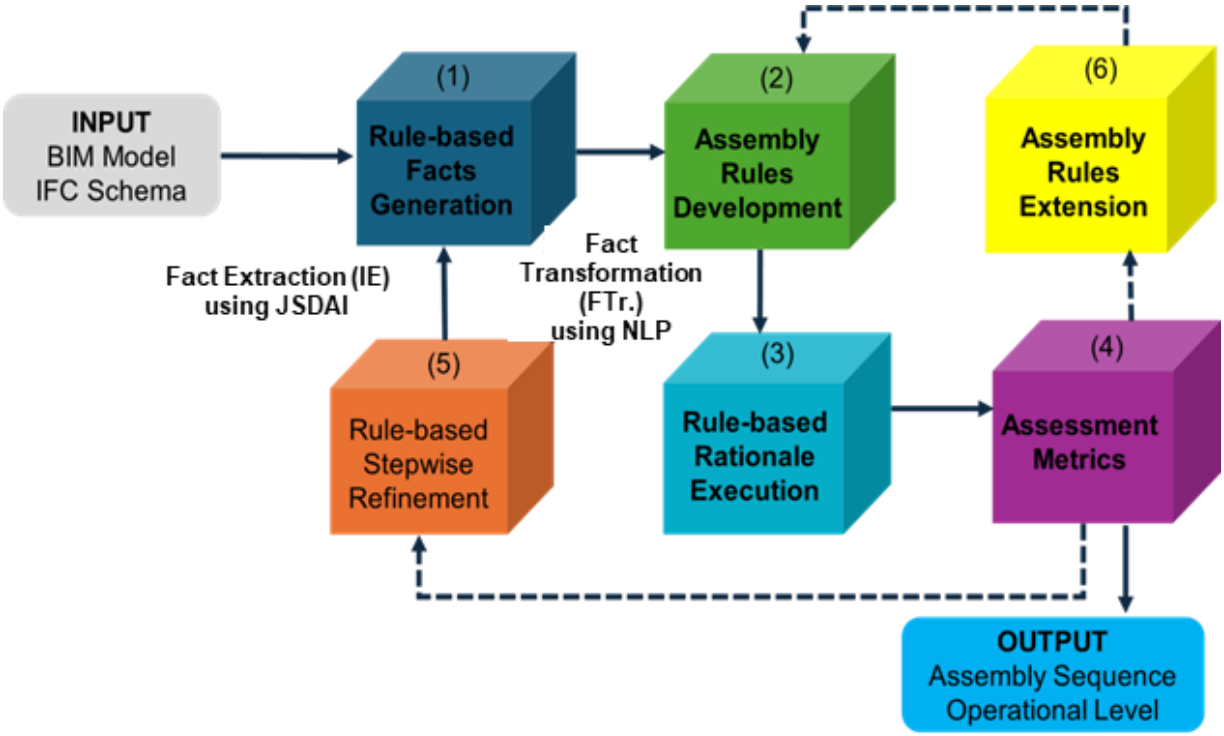


Figure 35. Overview of RAP Research Methodology

Step 1: Generation of Rule-based Facts

The process of creating logic facts is divided into three distinct sub-steps: Firstly, sub-step (1) involves the export of data from BIM to the IFC format. Following this, sub-step (2) entails the conversion of IFC data into rule-based facts. Lastly, sub-step (3) focuses on the stepwise refinement of these generated rule-based facts.

Sub-Step 1: Data extraction from BIM to IFC

The initial step in generating rule-based facts involves exporting BIM model instances into the IFC format, using tools like Autodesk Revit that are certified by BuildingSMART for IFC compatibility (BuildingSMART, 2023). This export process allows customization of various parameters such as the selection of a specific IFC schema version and model view definitions. For the implementation of the suggested methodology, the IFC 2x3 standard along with the Model View Definition (MVD) Coordination View (CV2.0) has been chosen.

Sub-Step 2: Transforming IFC data into rule-based facts

During the conversion phase, IFC data is transformed into rule-based facts through specialized algorithms for fact extraction (FE) and fact transformation (FTr), tailored to handle building design information encoded in the IFC format. This methodology processes BIMs in the ".ifc" file format, which adheres to the IFC schema, an internationally recognized data modeling standard under ISO 16739 (BuildingSMART 2014). The IFC format, known for its software neutrality, structures BIMs using the "STEP physical file" format specified in ISO10303-21, where each line represents a unique entity that could be either conceptual or relational.

During the FE phase, the Java Standard Data Access Interface (JSDAI) retrieves entities and their attributes from IFC files. The methodology utilizes late binding, compatible with various IFC schema versions (e.g., IFC 2x3, IFC2x3-TC1, IFC4), to ensure adaptability to evolving standards and facilitate processing across different IFC schema releases.

The FE process utilizes the EXPRESS data schema to interpret and structure BIM data, with detailed steps shown in Figure 36. The process begins by initializing variables and loading the applicable IFC schema version, then iteratively processes each line of an IFC file. A subroutine, SI, analyzes each line; if an entity is an aggregate type, SI recursively processes each sub-entity.

For non-aggregate entities, SI retrieves and records all attribute names and values as defined in the IFC schema.

After processing, SI stores a tuple for each entity containing the entity's name, line number, attribute names, and attribute values. An example of this tuple creation is shown in Figure 38, where an entity at line “#134271” results in a tuple that illustrates the method of extracting facts from IFC files.

In the FTr phase, semantic Natural Language Processing (NLP) techniques convert these tuples into structured logic facts, involving two main tasks: (1) semantic lookup of entity and attribute names, and (2) conversion of these into conceptual and relational facts. During the semantic lookup, names and values are matched against the IFC schema to find corresponding identifiers or enumeration values, reformulating these into underscore-separated terms for consistency. For example, the term “OWNERHISTORY” is matched and transformed into “owner_history” as shown in Figure 38. This transformation aligns entity and attribute names with the semantic, rule-based representation of the data, facilitating rule generation that meets regulatory standards.

In the transformation of entities and attributes using rule-based NLP, three main rules are applied:

1. **Entities Conversion**: Entities are transformed into concept facts by using the entity's name as the predicate name and combining the entity's name with its line number to create the predicate's argument. For instance, a column entity is transformed into "column(column134271)," where "column" is the predicate name, and "column134271" (a blend of the entity name and line number) serves as the predicate argument.
2. **Attributes Conversion**: Attributes are turned into relational facts by prefixing the attribute name with "has_" to establish the predicate name. The associated entity constant acts as

the first argument, and the attribute's value as the second, provided it doesn't reference another entity. For example, the attribute "global_id" for a beam entity is transformed into "has_global_id(column134271, 1bUGceuUvBshy6udF4uL24)."

3. **Reference to Other Entities**: If an attribute's value references another entity, this referred entity's constant is used as the second argument. An example is the transformation of the "owner_history" attribute of a beam entity into "has_owner_history(column134271, owner_history18)," where "owner_history18" is the second argument.

These transformations' structure data into a semantic, rule-based format that simplifies the interpretation and manipulation of BIM data.

Sub-Step 3: Refinement process for rule-based facts

The final sub-step in the process refines rule-based facts converted from the IFC format to ensure they meet the syntax requirements of the Prolog programming language and are suitable for rationalization. This refinement involves two main adjustments:

1. **Translation of Unit Symbols**: Unit measurements commonly use symbols from BIM authoring software, such as dimensions noted as 84" × 36". During refinement, these symbols are converted into full unit names for clarity and syntax compliance in Prolog, changing 84"_x_36" to 84_inches_x_36_inches.
2. **Encapsulation of Strings in Quotes**: Prolog requires specific formatting where strings that start with a number must be enclosed in single quotes to avoid syntax errors. For example, the predicate has_globalid(door1864,3famcbrsy2ysbbeuhqdlyz) would be refined to has_globalid(door1864,'3famcbrsy2ysbbeuhqdlyz') to ensure proper interpretation as strings.

An algorithm utilizing the regular expression syntax in Python (3.12) was developed to execute this refinement process. This algorithm is responsible for refining the rule-based factual information, as demonstrated in Figure 37.

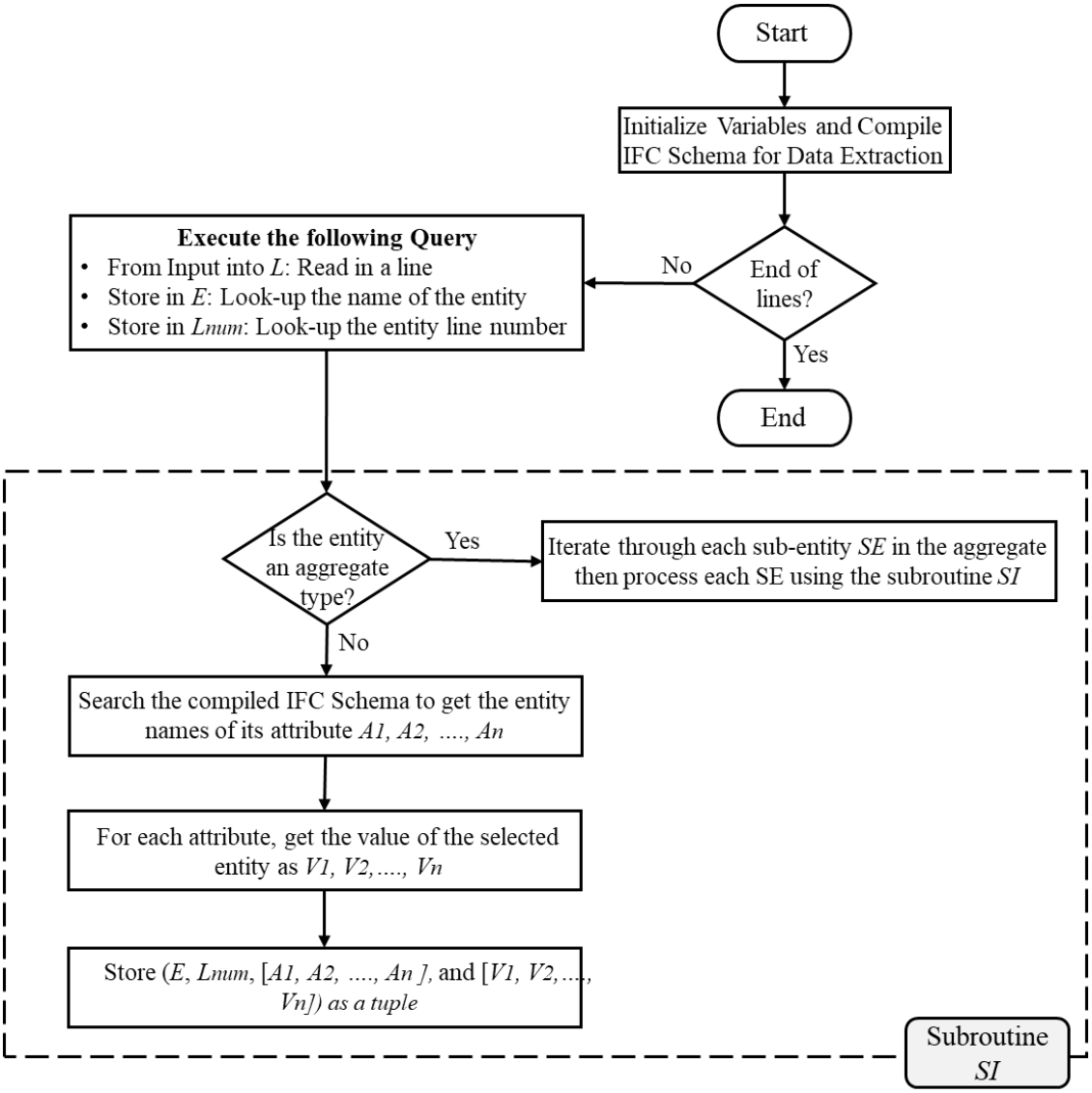


Figure 36. Proposed FE Algorithm

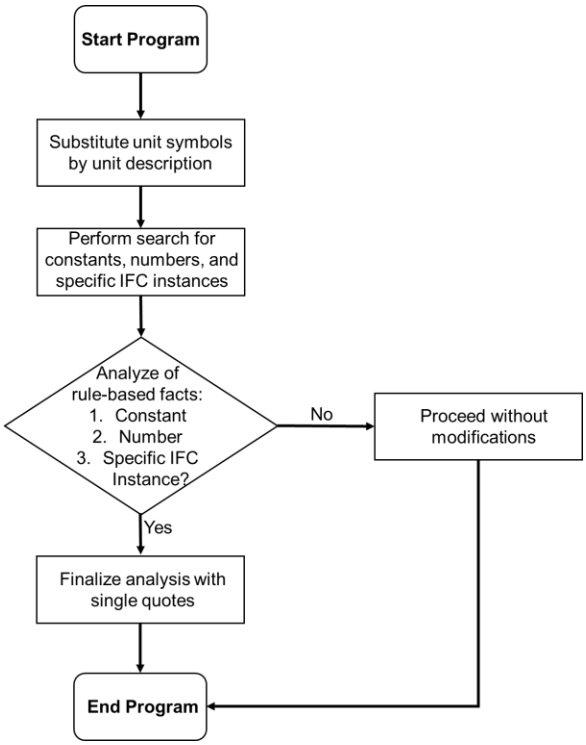


Figure 37. Refinement Algorithm Flow Chart

Level 1 IFC-Based BIM Model Lines	#18=IFCOWNERHISTORY(#17,#2,$,.NOCHANGE.,$,$,$,1710175198) #134265=IFCMAPPEDITEM(#134263,#3786); #134266=IFCSHAPEREPRESENTATION(#110,'Body','MappedRepresentation',(#134265)); #134267=IFCPRODUCTDEFINITIONSHAPE($,$,(#134266)); #134268=IFCCARTESIANPOINT((0.,0.,-1.2161458333333335)); #134269=IFCAXIS2PLACEMENT3D(#134268,$,$); #134270=IFCLOCALPLACEMENT(#152,#134269); #134271=IFCCOLUMN('1bUGceuUvBshy6udF4uL24',#18,'BIMSF-Dimension Lumber-Column:2x6:1042008',$,'BIMSF-Dimension Lumber-Column:2x6',#134270,#134267,'1042008');
Level 2 IE Output (Tuples)	(OWNERHISTORY, 18, [owninguser,owningapplication,state,changeaction,lastmodifieddate,lastmodifyinguser,lastmodifyingapplication,creationdate], [#17,#2,none,NOCHANGE,none,none,none,1710175198]); (LOCALPLACEMENT,134270,[placementrelativeto,relativeplacement].[#152,#134269]); (PRODUCTDEFINITIONSHAPE,134267,[name,description,representations],[none,none,(#134266)]); (COLUMN,134271, [globalid,ownerhistory,name,description,objecttype,objectplacement,representation,tag], ['1bUGceuUvBshy6udF4uL24',#18,'BIMSF-Dimension Lumber-Column:2x6:1042008',$,'BIMSF-Dimension Lumber-Column:2x6',#134270,#134267,'1042008']);
Level 3 Tuples after Semantic Look-up	(owner_history, 18, [owning_user,owning_application,state,change_action,last_modified_date,last_modifying_user,last_modifying_application,creation_date], [#17,#2,none,NOCHANGE,none,none,none,1710175198]); (local_placement,134270,[placement_relative_to,relative_placement].[#152,#134269]); (product_definition_shape,134267,[name,description,representations],[none,none,(#134266)]); (column,134271, [global_id,owner_history,name,description,object_type,object_placement,representation,tag], ['1bUGceuUvBshy6udF4uL24',#18,'BIMSF-Dimension Lumber-Column:2x6:1042008',$,'BIMSF-Dimension Lumber-Column:2x6',#134270,#134267,'1042008']);
Level 4 Rule-Based Facts as Output of ITr	owner_history(owner_history18) local_placement(local_placement134270) Product_definition_shape(product_definition_shape134267) column (column134271) . has_global_id (column134271, 1bugceuuvbshy6udf4ul24) . has_owner_history (column134271, ownerhistory18) . has_name (column134271, dimension-lumber-column: 2x6 : 1042008) . has_object_type (column134271, 2x6) . has_object_placement (column134271, localplacement134269) . has_representation (column134271, productdefinitionshape134267) . has_tag(column134271, 1042008).

Figure 38. Sample of IFC Conversion into Rule-Based Facts

3.1 Development Process for Assembly Rules

In developing assembly rules for fabricating building components, a knowledge engineering approach is applied, tailored to user-specific needs, incorporating two heuristic rules: specific situations and situation comparison. The first rule refines best practices by using highlighted instances, employing predicates for common building elements (e.g., walls, floors, columns, beams) to set up extraction and derivation rules. While the second heuristic involves comparing predicates across different building components to classify them accurately.

Assembly rules are organized into four categories based on their functions (Figure 39):

- **Rule A (Assembly Information Extraction)**: Extracts assembly details such as bill of quantities, product dimensions, and material types from building design components.
- **Rule B (Assembly Information Derivation)**: Derives physical properties (area, volume, weight) from building components using outputs from Rule A and external material libraries containing properties of wood and steel.
- **Rule C (Assembly Information Processing)**: Filters out non-wood/steel materials and unrelated object types.
- **Rule D (Assembly Information Fixation)**: Details the fixation materials and configurations, such as nails and screws.

These rules facilitate the extraction, filtering, and inference of data, enhancing insights from building design logic facts.

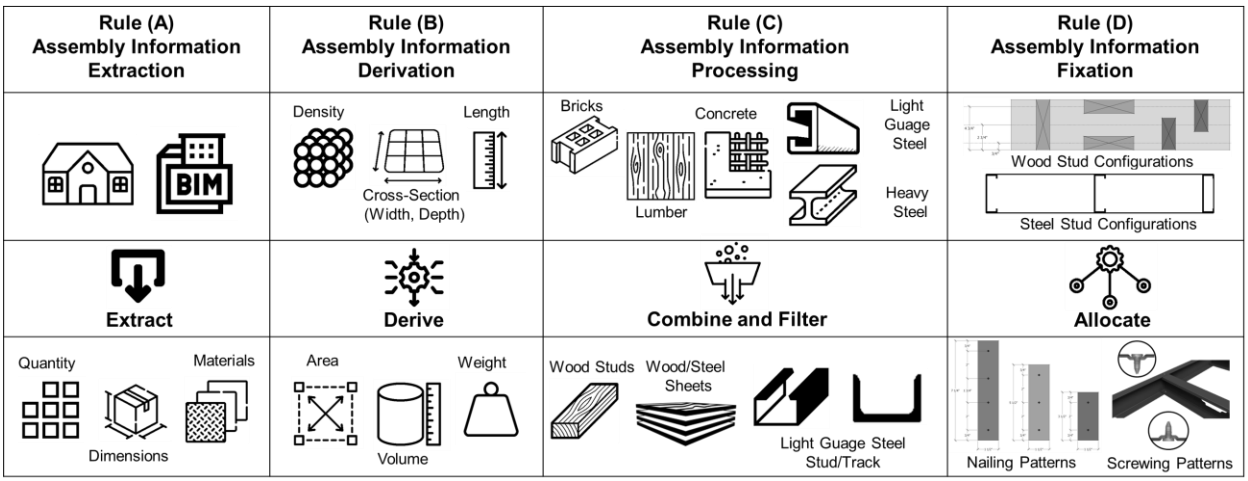


Figure 39.Types of Assembly Rules and Application Examples

The development of assembly rules for building components is organized into four main steps to ensure broad applicability and effective inference of data:

1. **Identifying Relevant Rules**: This step focuses on selecting assembly rules that effectively address specific constraints and sub-goals within a unification framework, such as deriving material information from column instances, as shown in Figure 40.
2. **Replacing Terms**: Constants and numbers within the rules are replaced with variables to generalize the rules for application across different instances.
3. **Connecting Rules**: Assembly rules are linked through logical conjunctions to form a coherent and interconnected rule set.
4. **Employing SOL**: Utilizes all-solution predicates to formulate the assembly rule and extend it to all building components attributes, facilitating comprehensive data extraction and derivation.

This structured approach allows for the detailed extraction of material properties and calculations of volume and weight for components like columns, based on their geometric representations and profile definitions, as shown in Figure 41. Additionally, specific assembly rules are tailored for

columns with different geometric representations (e.g., B-rep) to enhance their utility across various scenarios.

Sub-Step 1	relassociatesmaterial(relassociatesmaterial1849). material(material1849). has_relatingmaterial(relassociatesmaterial1849,material1833). column(column). has_relatedobjects(relassociatesmaterial1849,column134271). has_name(material1833,'BIMSF-Dimension Lumber').
Sub-Step 2	relassociatesmaterial(Relassociatesmaterial). material(Material). has_relatingmaterial(Relassociatesmaterial,Material). column(Column). has_relatedobjects(Relassociatesmaterial,Column). has_name(Material,Materialname).
Sub-Steps 3-4	rule3_column_material: L=[H | T], findall ((Column,Materialname), (relassociatesmaterial(Relassociatesmaterial),material(Material), has_relatingmaterial(Relassociatesmaterial,Material), column(Column), has_relatedobjects(Relassociatesmaterial,Column), has_name((Material,Materialname), L), foreach ((Column,Materialname) in L, Asserta (has_material_list(Column,Materialname))).

Figure 40. Assembly Rule Development for Extracting the Material Information

(a) **Material Module**	unit(unit01). density(density35). has_property('BIMSF-Dimension Lumber',density35). has_value(density35,35). has_unit(density35,unit01). has_name(unit01,pound_per_cubic_feet).
(b) **Volume Derivation**	rule5_column_volume: L=[H | T], findall ((Column,Volume), (column(Column), has_dimension(Column,Depth,X,Y) Volume equal Depth*X*Y), L), foreach ((Column,Volume) in L, assert (has_volume(Column,Volume))); assert (has_volume(Column, 0)).
(c) **Weight Derivation**	rule6_column_weight: L=[H | T], findall ((Column,Weight), (relassociatesmaterial (Relassociatesmaterial),material(Material), has_relatingmaterial (Relassociatesmaterial,Material), column(Column), has_relatedobjects (Relassociatesmaterial,Column), has_name ((Material,Value), density (Density) has_property (Value,Density), has_value (Density,Value1) has_volume(Column,Volume) Weight equal Volume*Value1), L), foreach ((Column,Weight) in L, format ("Weight of ~w is ~w ~n", [Column,Weight])).

Figure 41. Rule B: Assembly Rule Example for Columns: a) Material Specifications, and Rule to Derive b) Volume and c) Weight

3.2 Execution Process of Rule-based Rationale

The implementation of a rationale engine for building design involves several key components and processes, centered around rule-based facts, assembly rules, and supporting modules to enhance the rationalization process. These modules include a unit conversion module for converting any length unit into feet, and a material properties module that contains data on various material types and their densities.

The process initiates by loading the rule-based facts into the rationale engine, which then processes these facts alongside assembly rules using supporting modules. During this phase, the engine uses unification and substitution clauses to validate and apply the assembly rules to the given goals.

The output from the rationale engine, which indicates the success of the assembly rules, is crucial for assessing the effectiveness of the entire system.

The primary inputs for this process come from architectural design information extracted from BIM instance models formatted to IFC standards. This includes a detailed enumeration of structural components, such as the sequencing from bottom track through studs to top track, and includes their dimensional attributes and mass properties. Each structural element's spatial positioning, defined by coordinates and angular disposition relative to the BIM design model's coordinate framework, is methodically extracted from the IFC data.

This rule-based methodology ensures accurate generation of inputs necessary for subsequent simulation activities. The extracted data and information, processed through rule-based representation and inferential rationale, are crucial for both the simulation framework and the robotic control system. Detailed data inputs and methodologies for extraction and analysis of IFC model data are documented in Table 2 and Table 3.

Table 2. Assembly Rules Input Data for the Rationale Engine

Input data	**Description**
Density of element	Each element's density was allocated while matching elements from the BIM/IFC input to the rationale engine.
Sequence of construction	The construction sequence was established based on a flexible sequencing requirement, which entails any feasible sequence without optimization considerations. The elements are

	then introduced into the rationale engine according to this predetermined order.
Dimension	The element dimensions were correlated with the predetermined order within the rationale engine.
Position and orientation	This element represents both the location within the BIM environment and the orientation of the cross-section.

Table 3. Rule-Based Facts Input Data for the Rationale Engine

Input data	**Description**
Name	The name of the element provides a unique identifier and the type of elements (e.g., beam33, column35).
Local Placement	The local placement location is the centroid of one of the extreme end's cross-sections of the element relative to the global coordinate system of the BIM design model.
Main Axis	The main axis in IFC refers to the orientation of the longitudinal axis of an element relative to its local cross-sectional coordinate system. For regular elements, this axis corresponds to the direction of extrusion in a solid sweeping 3D representation, specifically aligning with the z-axis of the element's coordinate system.

Dimension	Dimensions include the width, depth, and height of a stud element.

4.4 Evaluation of Results and Error Analysis

For the purpose of experimentation, the designed FE and FTr algorithms were deployed using the JAVA Standard Edition Development Kit, specifically version jdk1.7.0_40. Access to IFC-formatted BIMs was facilitated through the utilization of JSDAI version 4.3. JSDAI serves as an interface toolkit for STEP (ISO 10303) defined as the Standard for the Exchange of Product model data. JSDAI is also tailored to support the development and Java implementation of EXPRESS data models, as specified in ISO 10303-11.

The model's performance was evaluated using a confusion matrix, a tool that organizes actual versus predicted classifications in a structured tabular format. In this matrix, each column correlates the true classifications with the predictions made by the developed model for the samples under review. The results obtained from this confusion matrix analysis are comprehensively presented in Table 4. This method not only highlights the accuracy of the model but also sheds light on any misclassifications, providing a clear visual representation of where the model performs well and where it may need improvement. Detailed analysis is crucial for refining the model and enhancing its predictive capabilities.

Table 4. Confusion Matrix

Confusion Matrix	**Predicted Category**	
True category	0	1
0	TN	FP

1	FN	TP

Table 5 represents the correlation between true negative (*TN*), true positive (*TP*), false positive (*FP*), and false negative (*FN*). The effectiveness of the extraction algorithms is assessed through five key metrics in terms of Sensitivity (*Sn.*), Specificity (*Sp.*), Precision (*Pr.*), Accuracy (*Acc.*), and F1-measure (*F1m.*). Sensitivity (*Sn.*) or true positive rate is the probability of a positive test result of an extracted rule-based fact (RBF), conditioned on the RBF truly being positive (Equation 1). While Specificity (*Sp.*) or true negative rate is the probability of a negative RBF result, conditioned on the individual RBF being negative (Equation 2). Precision (*Pr.*) is a metric that evaluates the accuracy of the extraction results. It is calculated as the proportion of correctly extracted RBF instances out of the total extracted RBF instances (Equation 3). In contrast, Accuracy (*Acc.*) measures the completeness of the extraction process, defined as a ratio composed of correctly extracted RBF instances to the existing RBF instances (Equation 4). Since relying solely on either precision or recall does not provide a comprehensive evaluation, the F1-measure (*F1m.*) is employed as an additional performance metric. The F1-measure represents a balanced assessment of the overall performance (Equation 5). The expressions for all the performance indicators are provided in Table 4. The other evaluation indicators investigated during the study are the error rate and the loss value of the algorithm for both the training and testing datasets.

Table 5. Performance Indicator Values Obtained for the Proposed Rationale Engine Algorithm

Eq. #	**Performance Description**	**Performance indicators**	**Equation Expression**
1	Sensitivity	*Sn.*	$Sn. = \frac{TP}{TP+FN}$
2	Specificity	*Sp.*	$Sp. = \frac{TP+TN}{TP+TN+FP+FN}$

3	Precision	*Pr.*	$Pr. = \frac{Correctly\ extracted\ RBF\ (TP)}{Total\ RBF\ extracted\ (TP + FP)}$
4	Accuracy	*Acc.*	$Acc. = \frac{Correctly\ extracted\ RBF\ (TP + TN)}{Total\ existing\ RBF\ (TP + TN + FP + FN)}$
5	F1-measure	*F1m.*	$F1m. = \frac{2\ \times Pr. \times Sn.}{Pr. + Sn.} = \frac{2 \times TP}{2 \times TP + FP + FN}$

4.4.1 Stepwise Refinement and Improvement of Rule-based Facts

In this phase, the focus is on addressing additional rule-based predicates that were not adequately processed during the rule-based facts stepwise refinement and are currently unsuitable for effective rule-based rationalization. These unrefined predicates can lead to unexpected outcomes in the related assembly rules. As a result, there is a necessity to extend the stepwise refinement algorithm to modify these predicates, thereby rendering them compatible for rule-based rationalization.

A typical example of this issue is observed with the predicate has_segments#(Term1, Term2). The inclusion of numbering in this predicate (such as has_segments1, has_segments2, ..., up to has_segments#) poses a challenge for the rule-based rationalization process. This is because the rule-based rationale engine treats each numbered predicate as a distinct and independent entity, which hinders its ability to unify all instances related to that specific predicate. Consequently, to ensure effective rule-based rationalization, the refinement algorithm needs to be adapted to address and restructure such predicates.

4.4.2 Expansion Process for Assembly Rules

Initially, the proposed methodology includes a set of assembly rules derived from the developed BIM models. For instance, this method can segregate columns and beams modelled in a swept solid representation, as featured in the model, but it lacks the capability to analyze them with other geometric representations not included in the model, such as B-rep. To improve its performance,

additional assembly rules are progressively added to the existing rule set to tackle instances where the current rules fall short.

As these new assembly rules are incorporated, the method becomes increasingly proficient at accurately analyzing geometric representations and material information that were previously misidentified. This step-by-step enhancement of the assembly rules significantly elevates the effectiveness and overall performance of the proposed methodology. This process is achieved through the iterative and cumulative expansion of the assembly rule set, continually refining the system's ability to handle a wider variety of scenarios and complexities.

4.5 RAP Framework Design

This study investigates the utilization of robotic manipulators to assemble prefabricated wood and steel wall panels from pre-made components. The control of these robotic manipulators involves guiding the end-effector's tip along a predetermined sequence of coordinates. The primary challenge here is to generate a logical assembly sequence for building components and precisely position them in the required spatial coordinates. To address this challenge, our research proposes expanding the digital blueprint of prefabricated components in BIM into instructions for robotic control through the application of RoboSimX.

RoboSimX involves the development of a Robotic Assembly Planner (RAP) module, a Robotic Task Planner (RTP) module, and, subsequently, a Robotic Motion Planner (RMP) module. In this section, the RAP module is discussed and developed. An overview of the RAP prototype's architecture is provided in Figure 42. As discussed by Ding et al. (2020), BIM projects contain substantial spatial information that can be harnessed for construction activities. For instance, BIM projects comprise loadable families, which are visual representations of prefabricated building components (Kontovourkis et al., 2020). These families are spatially integrated to create a unified

BIM project using FrameX (an in-house developed Autodesk Revit API as shown in Figure 43), resulting in georeferenced properties crucial for generating assembly coordinates and assembly sequences for multi-panels.

BIM design tools like Autodesk Revit host extensive databases for project management. Programming interfaces such as Dynamo are employed to enable the extraction and retrieval of data, enhancing usability for end-users (Chong et al., 2020). Within the framework of this research, the ROS is used to design and create a robotic prototype specifically for assembling prefabricated elements in wall panels. Dynamo plays a crucial role by supplying ROS with the necessary data inputs for its functions. ROS then carries out the motion planning algorithm, which processes the assembly sequence data to calculate the robots' kinematic parameters required for assembling wall panels, including joint and path configurations. Additionally, a data transmission interface has been established to facilitate the exchange of data between the modules.

In this research, a developed Dynamo-based assembly planning algorithm embedded in the assembly planning module identifies the assembly coordinates of prefabricated elements in the wall panel Revit model and generates a feasible assembly sequence based on these coordinates as shown in Figure 44. This assembly sequence data is essential for robotic motion planning. In a later stage of the research, the task allocation and motion planning modules will be discussed.

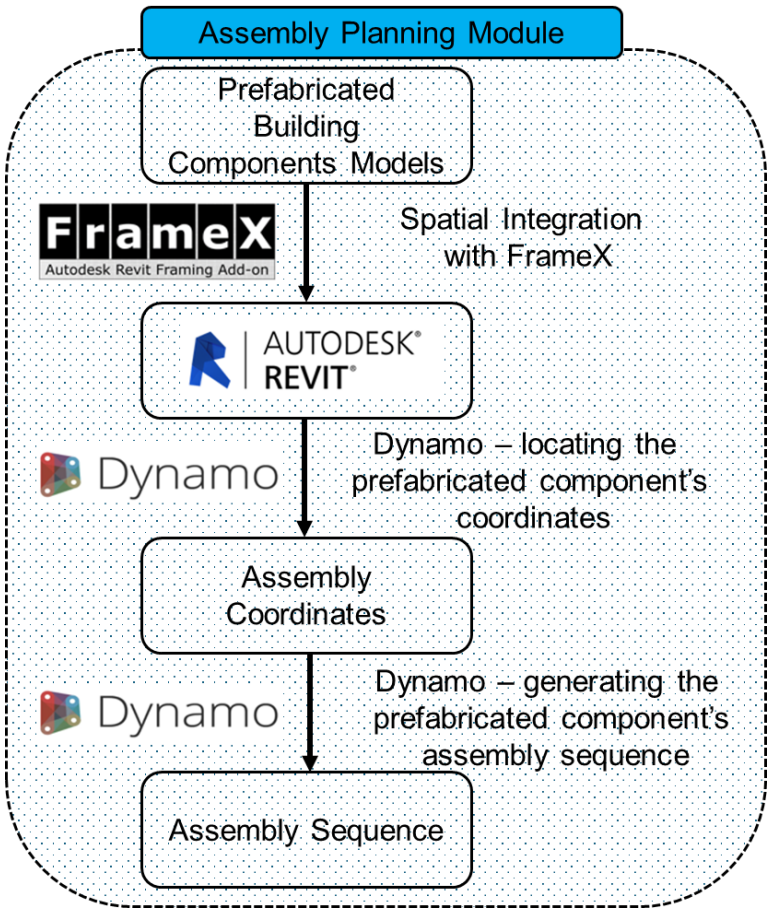


Figure 42. Assembly Planning Module Prototype Architecture

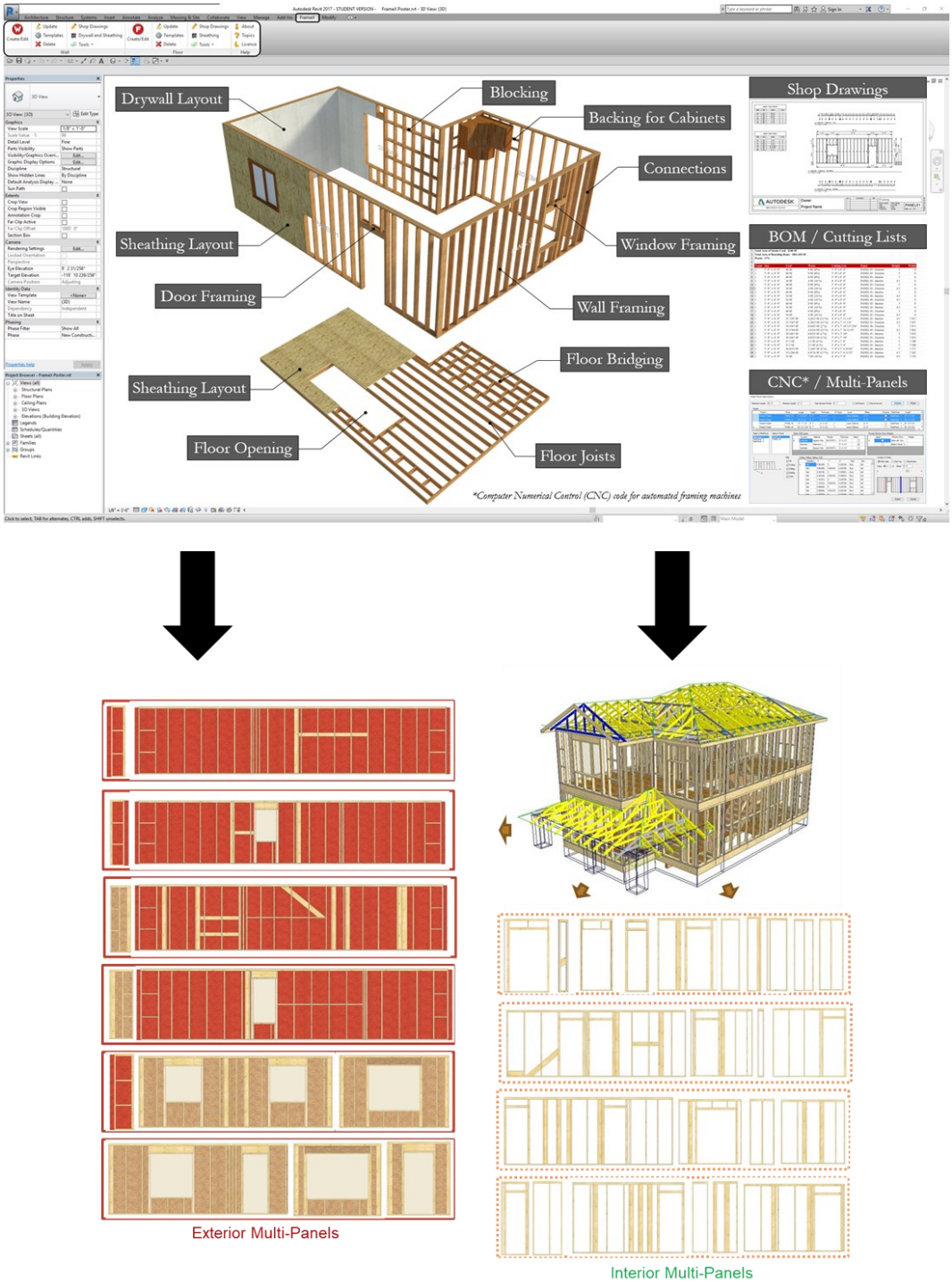


Figure 43. BIM Project Creation Using Frame X

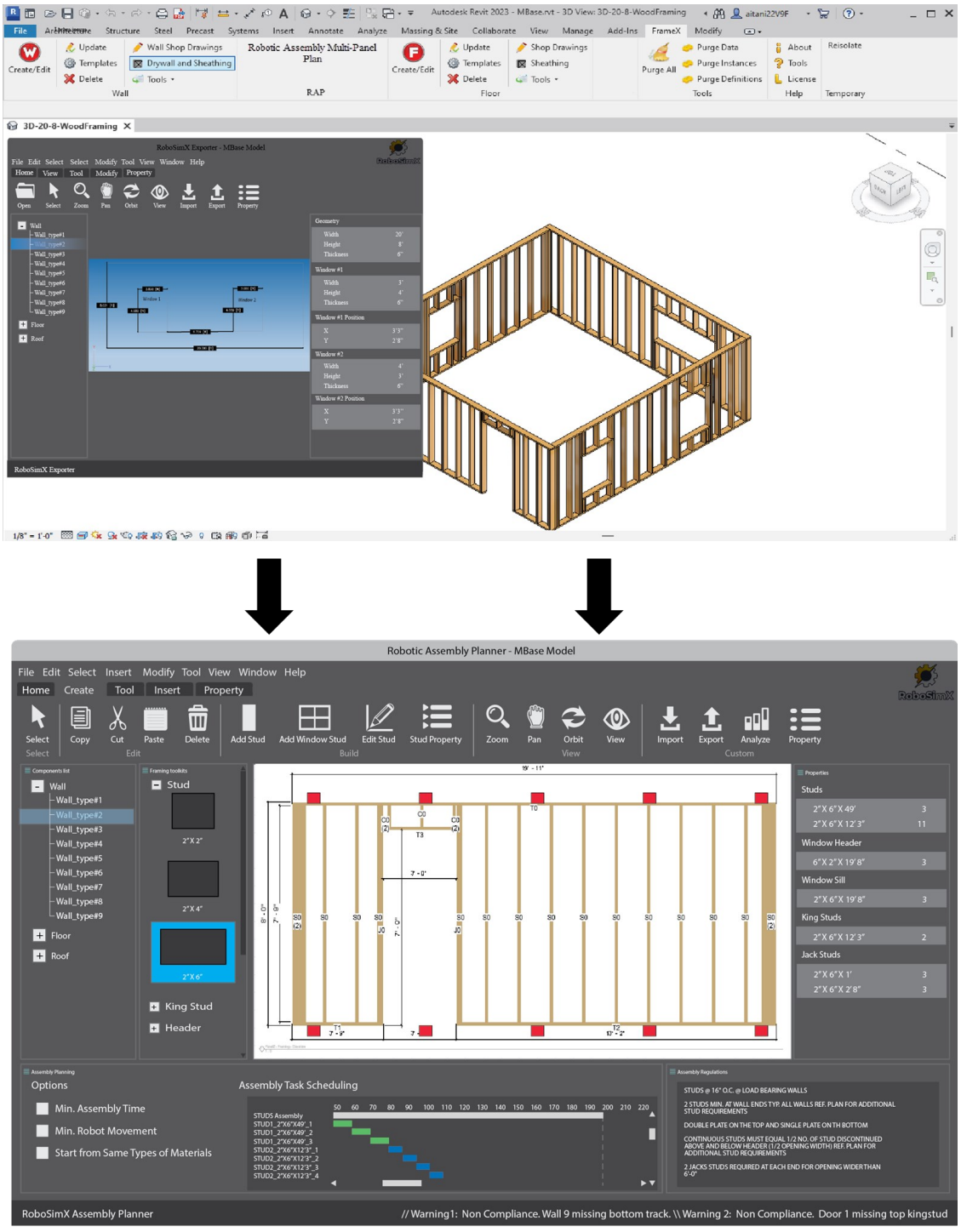


Figure 44. Robotic Model Exporter and Assembly Planner using RoboSimX User Interface

4.5.1 Assembly Planning Algorithm: Determining Assembly Coordinates and Sequence

In the study, the implementation focus of the assembly planning paradigm involves determining the spatial coordinates for installing architectural elements, followed by formulating a viable

sequence for their assembly, predicated on these coordinates. The research utilized the Autodesk Revit API—Dynamo—to engineer the assembly planning mechanism, designated as the ASP algorithm. Dynamo serves as a graphical programming platform augmenting the parametric analysis features of Revit, as noted by Fernández-Rodríguez et al. (2018). Dynamo's analytical functionality is facilitated via functional nodes, each comprising input and output interfaces, sequentially linked to construct an integrated logic. Python scripting is leveraged by users to devise nodes tailored for distinct functionalities (Chong et al., 2020). The structural elements of the ASP algorithm are illustrated in the below figures and in Algorithm 1. The algorithm is segmented into four distinct parts, each explained in more depth in the subsequent paragraphs. The developed algorithm categorizes building components and sorts them based on their geometric properties within a BIM context. It includes operations for sorting structural elements such as framing, columns, floors, and walls according to their spatial coordinates. The sorting function sortbyXYZ() organizes these elements based on their X (length), Y (height), and Z (depth) coordinates. The script differentiates between wall elements (frame and sheet) within the loop, appending them to separate result lists after sorting. The final output is a tuple containing two lists: result_frame for frame elements and result_sheet for wall elements.

Part 1 of the ASP algorithm is designed to enable Dynamo to recognize all architectural elements from the Revit model, as shown in Figure 45. This process is facilitated by integrating the nodes 'Categories ()' and 'All Elements of Category ()'. 'Categories ()' node functions to list the title of structural elements extracted from the model, such as beams, columns, floors, and roofs. Revit maintains a cache that monitors the architectural elements by assigning a distinct identifier (ID) to each element. The node 'All Elements of Category ()' can access the identifier IDs of all elements

within the specified categories. The output of this node is then fed into the final node, which compiles the identified elements IDs.

In Part 2, the algorithm utilizes the nodes 'Element.Geometry ()' and 'Solid.Centroid ()' to ascertain the spatial positioning of the identified elements within the coordinate system of the Revit model. This node input is the ascertained element IDs, which contain their corresponding geometrical data from Revit model. Subsequently, 'Solid.Centroid ()' node analyzes the edges and vertices of each geometrical element, calculates the centroid point coordinates, and highlights its position illustrated as black dots relative to the frame of reference, as shown in Figure 49.

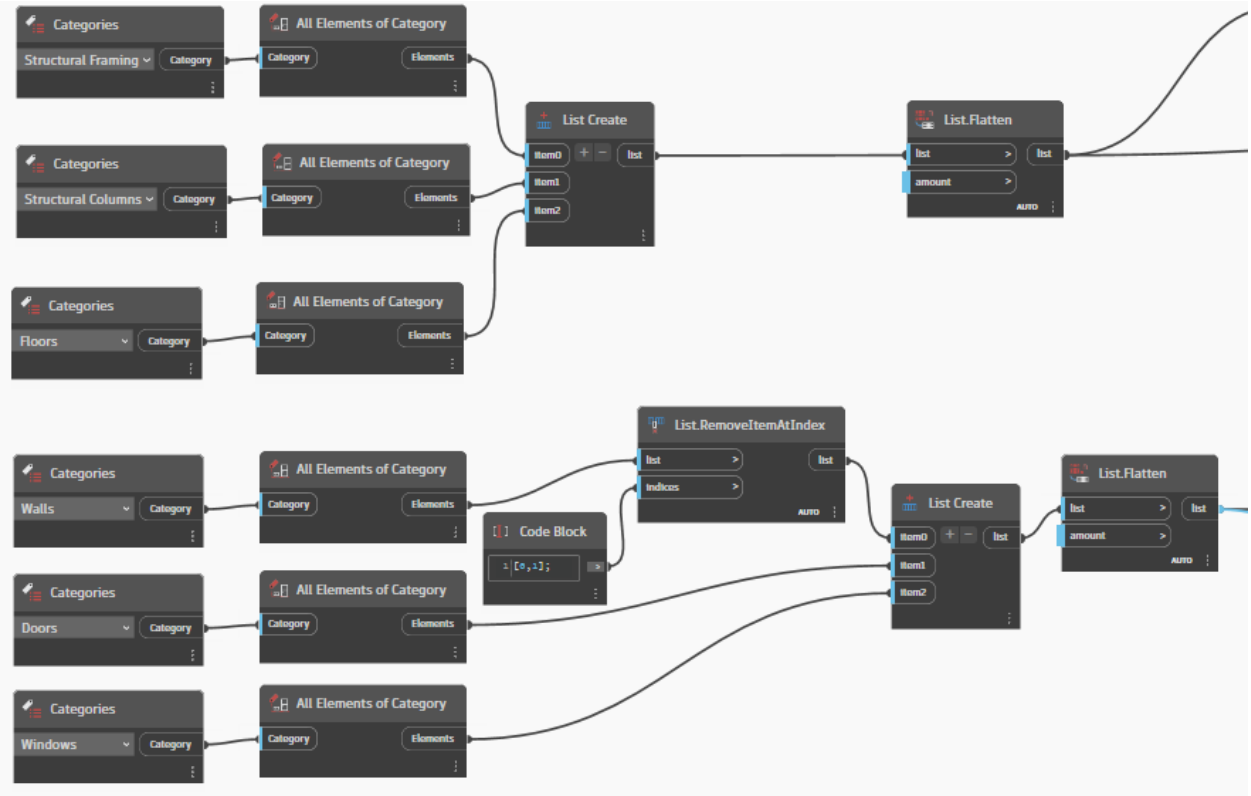


Figure 45. Part 1 of the Architecture of the ASP Algorithm

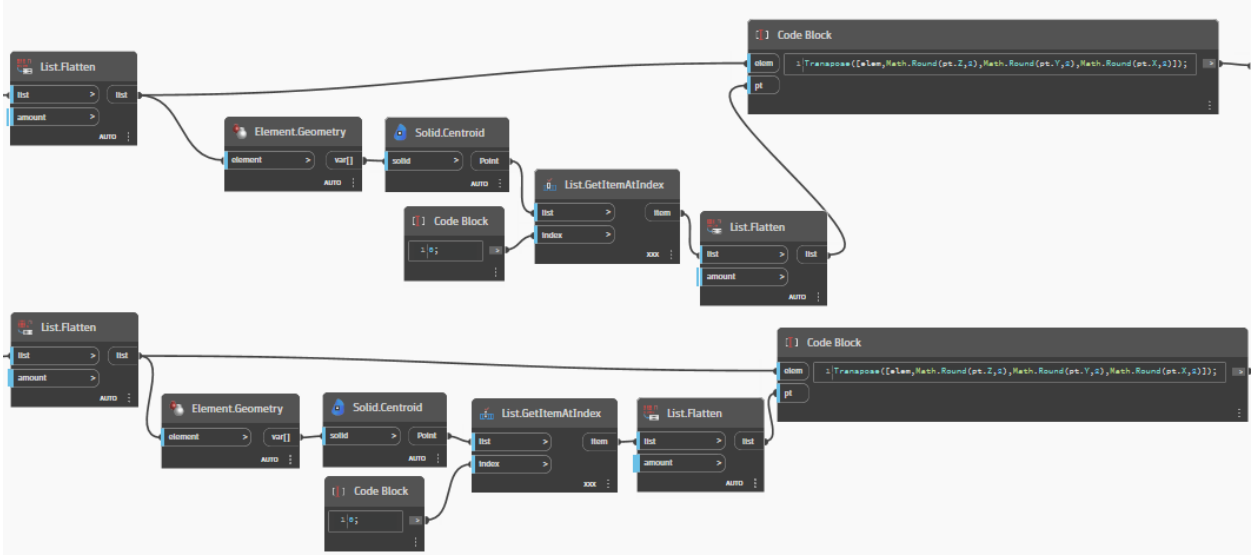


Figure 46. Part 2 of the Architecture of the ASP Algorithm

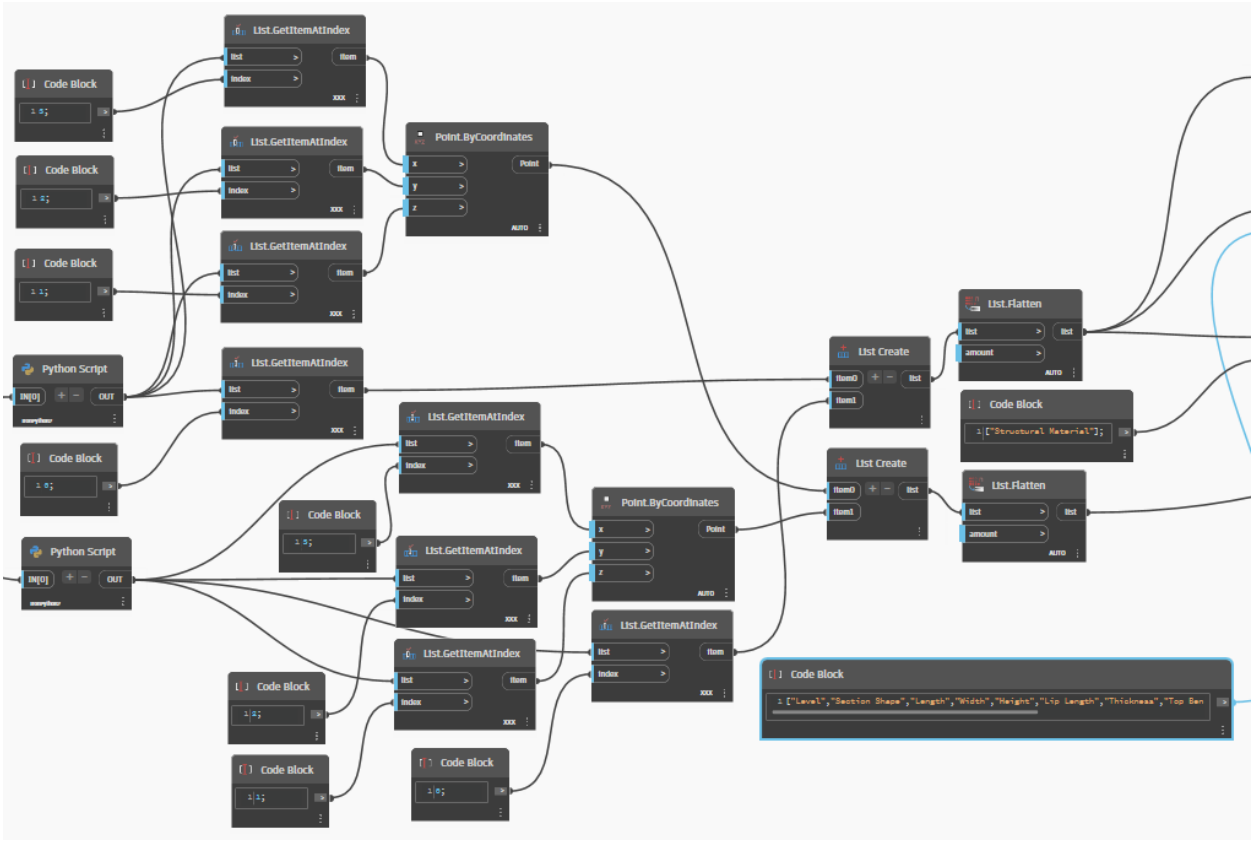


Figure 47. Part 3 of the Architecture of the ASP Algorithm

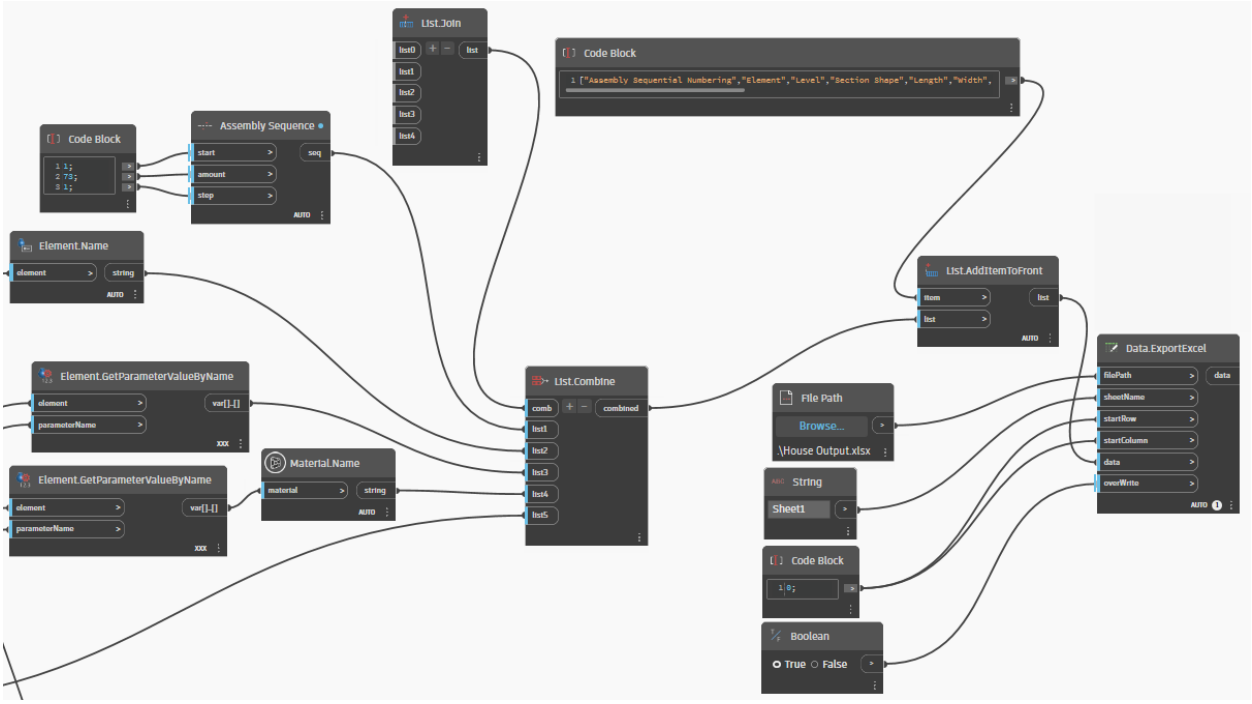


Figure 48. Part 4 of the Architecture of the ASP Algorithm

Algorithm 1

```
asplst ← categories(Structural Columns);
asplst ← categories(Structural Framing);
asplst ← categories(Walls);
asplst ← categories(Floors);
asplst ← asplst.Geomertry();
asplst ← asplst.Centroid();
sortybyXYZ(asplst):
    asplst.sort(key=lambda asplst: (asplst.X, asplst. Y, asplst.Z));
    return asplst;
end
input ← IN[0];
for i in input:
    if i.categories() ! = Sheets:
        result_frame ← append(sortbyXYZ(i));
    else:
        result_sheet ← append(sortbyXYZ(i));
end
OUT ← {result_frame, result_sheet}
```

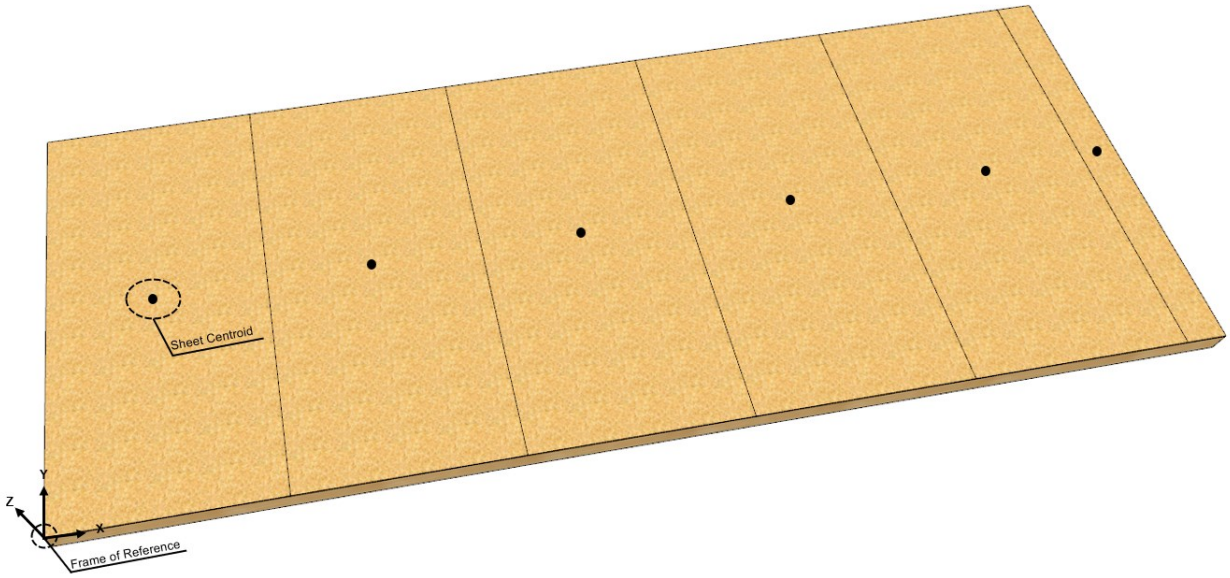


Figure 49. Centroids of the Building Components are Plotted in Dynamo

Finally, Parts 3 and 4 of the ASP algorithm are devoted to the formulation of an ordered strategy for the construction sequence of architectural elements. It describes the process of assembling a wooden panel through a progression of discrete steps: starting with the construction of the frame and progressing to the installation of wall panels (Figure 47 and Figure 48).

The prescribed sequence for erecting the frame adheres to an orderly methodology. Generally, the structural framework consists of bottom and top tracks that act as horizontal supports and columns (studs) that provide vertical reinforcement, as shown in Figure 50 for the wood case study and Figure 51 for the LGS case study. The modelled procedural construct yields a sequence vector that sequentially corresponds to the x, y, and z axis orientations. Given the specified dimensions of the wall panel—40 ft in length, 5.5 ft in depth, and 12 ft in height—the sequence vector $\vec{u}$ is calculated employing the formula $\vec{x} + \vec{y} + \vec{z}$ (representing the vector notation), resulting $\vec{u} = (0,0,40) + (0,12,0) + (5.5,0,0) = (5.5,12,40)$ as shown in Figure 52.

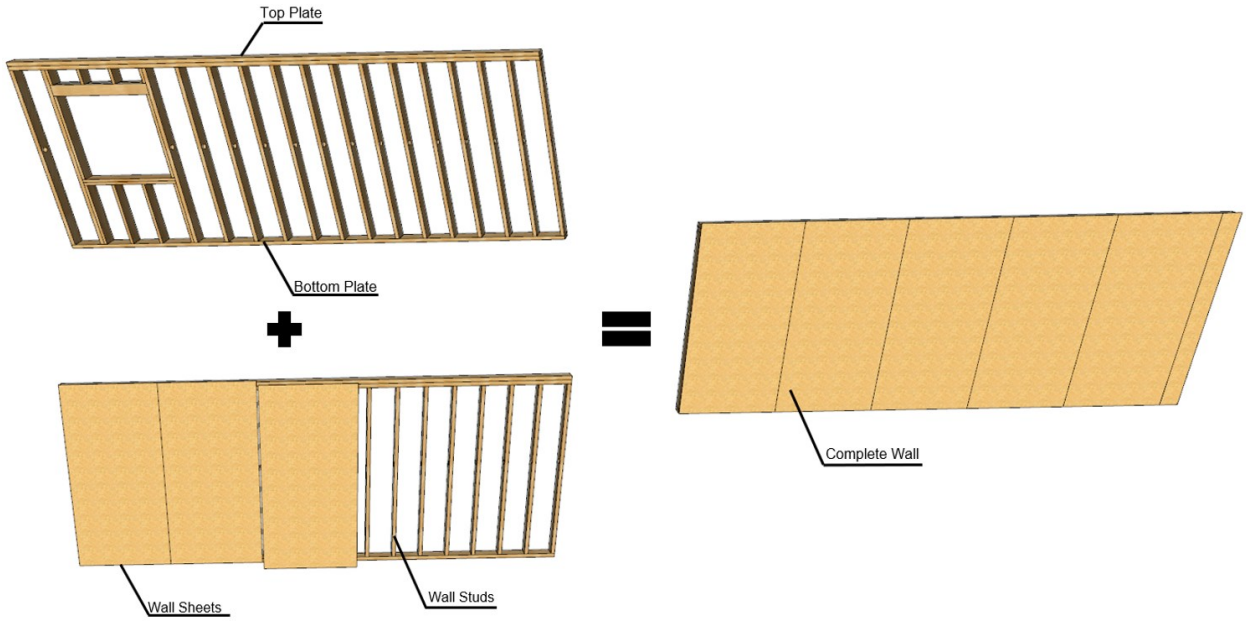


Figure 50. Assembly Process of a Wood Wall Panel - Combination of Frame and Sheet

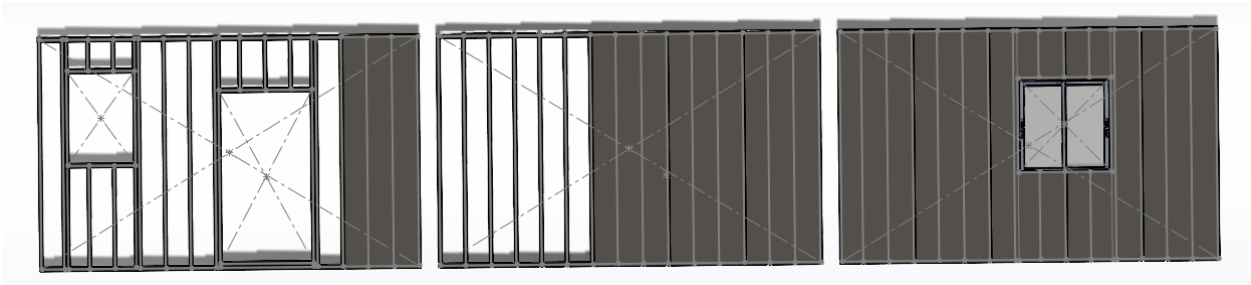


Figure 51. Assembly Process of a LGS Wall Panel - Combination of Frame and Sheet

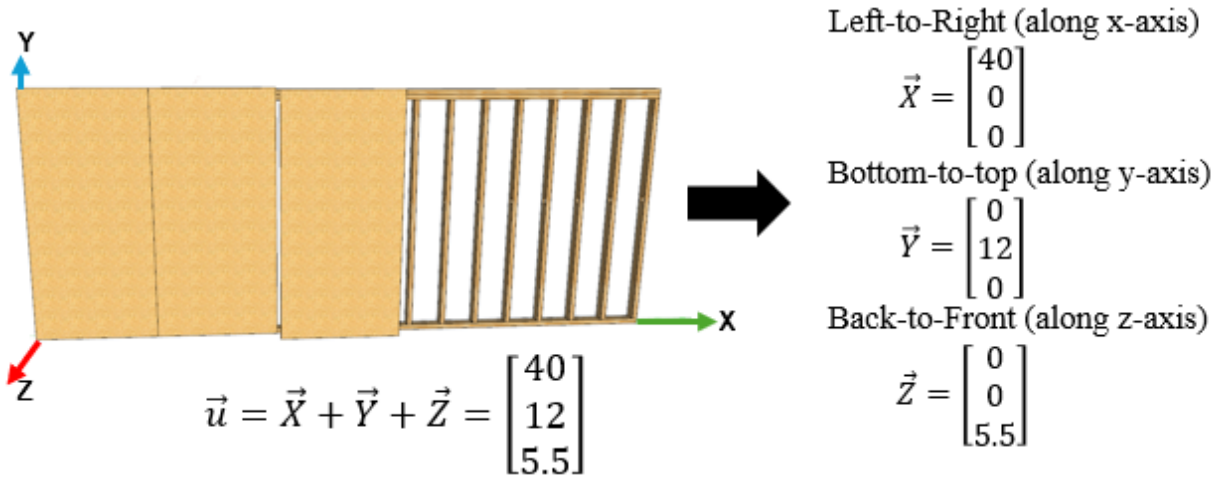


Figure 52. Sequence Processing of Vector $\vec{u}$

Within this section, the ‘Assembly Sequence ()' function is designed to compute the sequence vector $\vec{u}$ as explained. Triggering the 'Assembly Sequence ()' function puts the vector $\vec{u}$ to use, systematically ascertaining the spatial coordinates (x, y, z) for the centroids of the architectural elements. The operation initiates with the creation of a sorted list of building elements, prioritized by their ascending values along the x-axis (progressing from the lowest point upwards). For elements sharing equivalent x-coordinate values, a sorting algorithm is applied that organizes these elements first along the y-axis (spanning from left to right) and subsequently along the z-axis (stretching from the back to the front). This structured approach ultimately establishes the order in which the frame components are to be assembled. A comparable strategy is subsequently employed to extract the sequence in which the wall panels are assembled.

Autodesk Revit operates with an extensive database for project management, methodically classifying and archiving the attributes of prefabricated elements. These attributes are methodically cataloged using various indices, including element ID, category, and spatial coordinates relative to the BIM reference framework. Illustrative of this systematic organization is the instance of a stud/beam component showcased in Figure 53, wherein the stud/beam (assigned the globally unique index: 36686 is described with 17 discrete indices within the Revit database infrastructure. These indices include:

- Index #1: Represents the central axis of the stud/beam within the Revit environment.
- Index #2: Determines the foundational constraint of the stud/beam (e.g., the level of floor installation).
- Index #3: Designates the stud/beam's base vertical displacement from the floor level.
- Index #4: Assigns the floor level constraint to the stud/beam's upper boundary.

- Index #5: Represents the vertical displacement of the stud/beam's upper edge from the respective floor level.
- Index #6: Confirms the stud/beam element as a structural entity.
- Index #7: Defines the structural function of the stud/beam (e.g., as a support for horizontal loads).
- Index #8: Documents the geometric profile of the stud/beam.
- Indices #9 to #14: Detail the dimensional attributes of the stud/beam, encompassing length, breadth, height, perimeter, wall thickness, and volume.
- Indices #15 and #16: Characterize the material composition of the stud/beam.
- Index #17: Details the spatial coordinates of the stud/beam in relation to the Revit model’s referencing grid.

In the planning phase for assembly, the sequence of operations is determined based on the spatial positioning and intended order of prefabricated elements using a semantic evaluation of the BIM representation of a wall panel. This involves querying attributes related to the components within the BIM's data structure, facilitated by Dynamo, which allows Python scripting to perform semantic queries (Chong et al., 2020). For instance, Dynamo might query a specific index to extract locational coordinates of all elements in a prefabricated wall panel, organizing this data sequentially along the x, y, and z axes.

The resulting assembly list of construction components is arranged in a direction-dependent sequence according to the BIM model's coordinate system. This structured sequence is critical for the joint-level planning in the execution phase. Here, the robotic arm's coordinate system is set as the global reference, and the spatial data is recalibrated to this framework to aid intricate path planning.

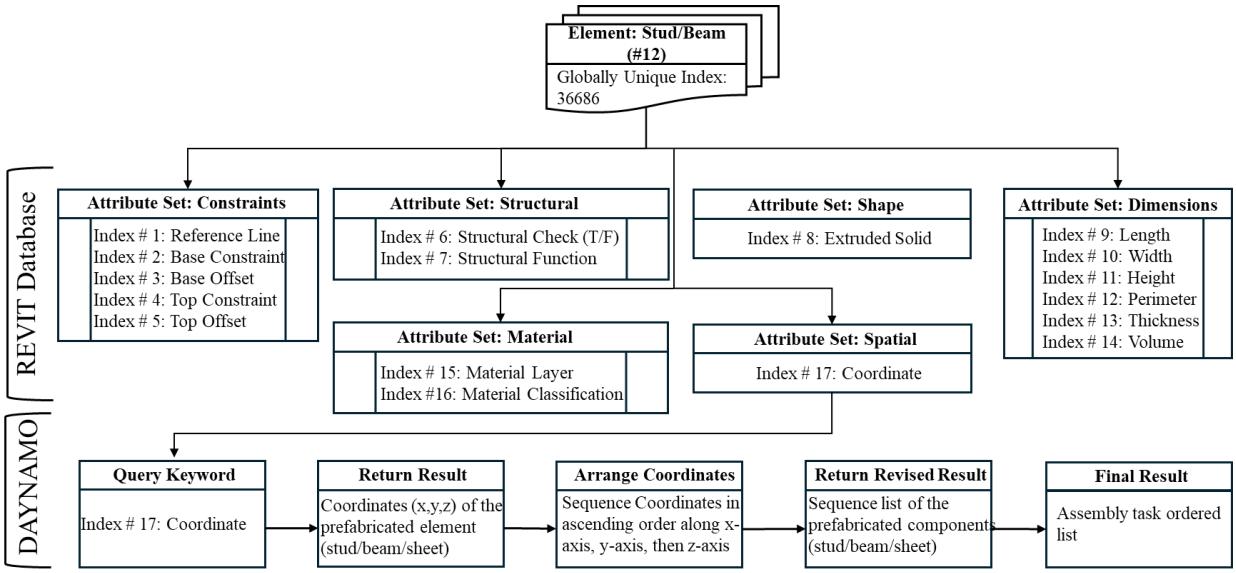


Figure 53. Revit Database and Dynamo Query

The output generated consists of the **result_frame** array, which holds the centroidal coordinates of frame elements like studs and beams, organized by their assembly sequence. Similarly, the **result_sheet** array stores the centroidal coordinates of sheet components, also arranged in their assembly order. The formula **OUT = result_frame + result_sheet** combines these two arrays, reflecting the sequential assembly process where the frame is constructed first, followed by the sheet panels. This combined data is then formatted for use within the ROS, aiding in robotic motion planning. This approach leverages for interoperable data exchange, effectively connecting the assembly planning stage in BIM with the task and motion planning stage in ROS, to be discussed in later sections.

The communication data transmission interface, as shown in Figure 54, is designed for efficient data transfer between framework modules at the Dynamo, IFC, and ROS interfaces.

The data transfer process involves several key steps:

- **Data Export**: The 'Export_IFC()' node at the Dynamo interface processes and exports the ordered assembly sequence list as an IFC file.

- **Data Formatting**: The 'Ifc2x3CV2.03D()' node at the IFC interface formats the data entries into a string format that the ROS system can interpret, following the format: "#IFC Index = ifcPropertyStringValue(Parameter Label).placement(Parameter Coordinates)".

- **Data Interpretation and Broadcasting**: At the ROS interface, the 'Subscriber()' node interprets the assembly coordinates and sequence data from the IFC tags. The 'Publisher()' node then broadcasts the joint and trajectory parameters to the robotic manipulator.

- **Robotic Motion Planning**: Utilizing the IFC data, robots calculate joint and trajectory parameters necessary for assembling the prefabricated elements. The system uses Denavit-Hartenberg (D-H) parameters to describe the joint parameters of a robot arm, defining the position and orientation of the robot joints essential for motion trajectory calculation.

This streamlined process ensures that all data related to the assembly sequence, joint parameters, and trajectory is accurately communicated across the interfaces, guiding the robotic manipulator to execute the assembly based on coordinates and parameters derived from the BIM model. The entire system ensures that each component is placed accurately according to the predefined sequence and configuration, facilitating an automated and efficient assembly process.

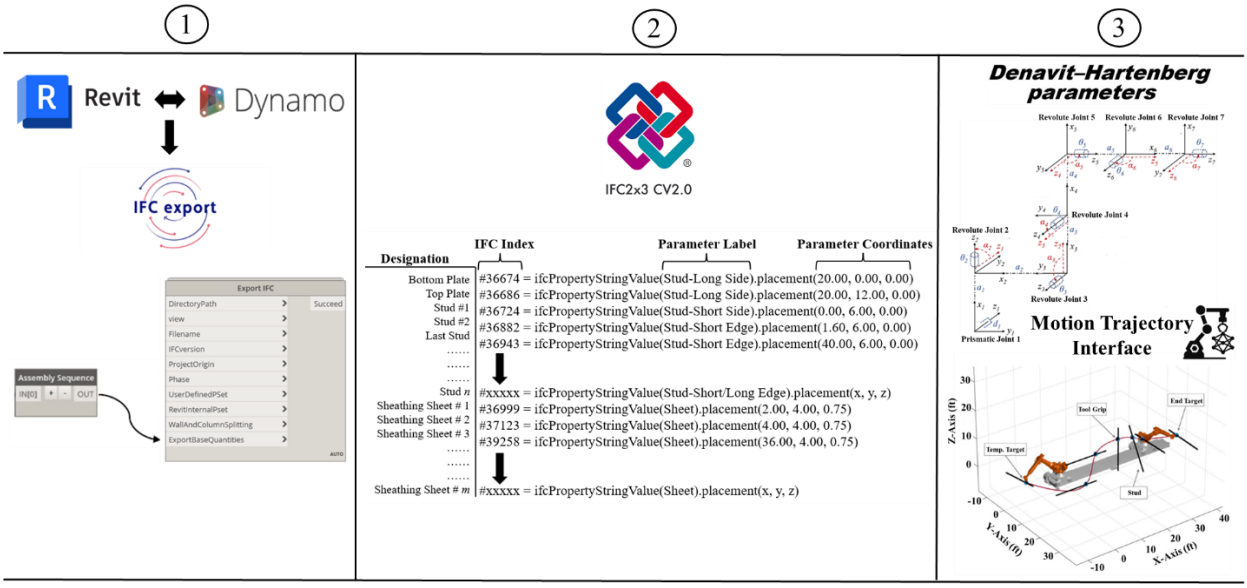


Figure 54. Communication Interface for Data Transmission Between the Modules

4.5.2 Creating Models of the Construction Materials

In the wood-based residential construction sector, framing and sheathing are crucial and time-intensive tasks in building a wood wall. Framing involves constructing the structural skeleton, consisting of vertical studs, horizontal beams, and door and window openings. The assembly of these elements requires precision to ensure structural integrity and accommodate architectural designs, making it labor-intensive. It sets the foundation for room layouts, window and door placements, and overall structural stability. Typically, 2-inch by 4-inch and 2-inch by 6-inch studs are used for their versatility and ease of handling, as noted by Allen et al. (2017). The choice between 2x4 and 2x6 studs depends on insulation needs, structural requirements, and wall thickness, with 2x4 studs used for minimal insulation needs and 2x6 studs for higher insulation or load-bearing walls.

After framing, sheathing is applied to provide a continuous, rigid surface that strengthens the structure and protects against external elements. This involves attaching large panels like plywood or Oriented Strand Board (OSB) to the frame using nails or screws. Panels are measured and cut

to accommodate door and window openings, with tightly aligned and secured seams enhancing structural integrity, thermal insulation, and weather resistance. OSB panels typically range from 1/4 in to 3/4 in in thickness and are commonly sized at 4 ft by 8 ft with other size variations available to meet specific construction needs and minimize joints.

Framing and sheathing are labor-intensive processes that demand significant manual labor, skill, and meticulousness due to precise measurement and cutting requirements, the effort of handling heavy materials, and compliance with building codes. The complexity of architectural designs can extend these stages as more intricate structures necessitate detailed work to accommodate unique features.

In the simulation environment, 3D models of wood studs and sheets are created to simulate the assembly process. These models serve as foundational templates during the mapping stage, where their dimensions and densities are adjusted based on BIM data to ensure the simulation accurately reflects the physical characteristics detailed in the BIM specifications.

4.5.3 Construction Operations and Control System Definition

In this stage, the operational behavior of the robotic system is mapped, programmed, and incorporated into the control system, with a specific focus on construction operations, primarily the assembly process (framing and sheathing). The control system is tasked with directing the robotic system's actions to execute the assembly operations, drawing upon the information extracted from the BIM data inputs.

The development of this control system is structured into six distinct steps:

1. Identification of framing target locations pertinent to the framing operation.
2. Determination of nailing target locations for the nailing operation.

3. Specification of the placement orientation of the frame in relation to the robot.
4. Development and establishment of the subroutines for the robot's operations.
5. Programming of the control system and integration of BIM data within the robot controller.
6. Definition of the operational behavior for each component within the robotic system.

Each of these steps is essential in ensuring that the robotic system functions efficiently and accurately, in accordance with the predefined assembly procedures and based on the data provided by the BIM models.

4.5.4 Determining the Framing Target Locations for the Framing Operation

Framing target locations are defined as the centroids of construction elements, calculated as Cartesian coordinates in relation to the robot's base origin. These locations represent the final assembled position of each element within the frame, derived from BIM design models. The BIM data input includes essential details such as local placement locations, dimensions, and main axis orientations of the elements.

Subsequently, the process necessitates a series of coordinate transformations. These transformations, crucial for the framing operation, are divided into eight distinct sub-steps and engage three different Cartesian coordinate systems:

1. Global coordinate system extracted from the BIM design model.
2. Local coordinate system specific to the robot and/or TCP.
3. Global coordinate system associated with the simulation world.

To streamline and simplify this transformation process, the origin of the robot's local coordinate system is aligned to coincide with the origin of the global coordinate system of the simulation

world. The detailed sub-steps (a-h) involved in these coordinate transformations are explained below and illustrated in Figure 55.

a. Adjust the BIM coordinate system's origin, O_B, to match the robot's coordinate system origin, O_R. This makes converting between the BIM and robot coordinate systems easier.
b. Calculate the centroid of the elements, taking into account their length, positioning, and the orientation of their primary axis to determine their center of mass.
c. Establish the center of mass coordinate of the initial element as the reference point for framing. This reference point acts as the foundational position for aligning all elements within the structure.
d. Check the length and width unit in both coordinate systems. Should they vary, adjust the length values in the BIM design model coordinates to align with the length unit used in the simulation environment.
e. Identify the sections (quadrants) where framing activities will take place, in relation to the robot's coordinate system.
f. Shift all elements from their initial locations on the X-Y-Z plane, using the framing reference point as the offset.
g. Adjust the orientation of the BIM coordinate system's axis to align with the specified framing zone. The required number of rotations is determined by how the wall's direction in the BIM design model compares to the robot's position. The angle at which a construction element, such as a wall, is set depends on its placement and direction in the BIM design model. Whether the wall is parallel or perpendicular to the robot dictates whether the BIM design model's Y, X, or Z dimensions align with the robot's axis.

Furthermore, the BIM design model's Z dimension is reoriented to match the X axis in the simulation's coordinate system.

h. Incorporate the offset measurements from the robot's coordinate system origin into the calculated coordinates of the elements.

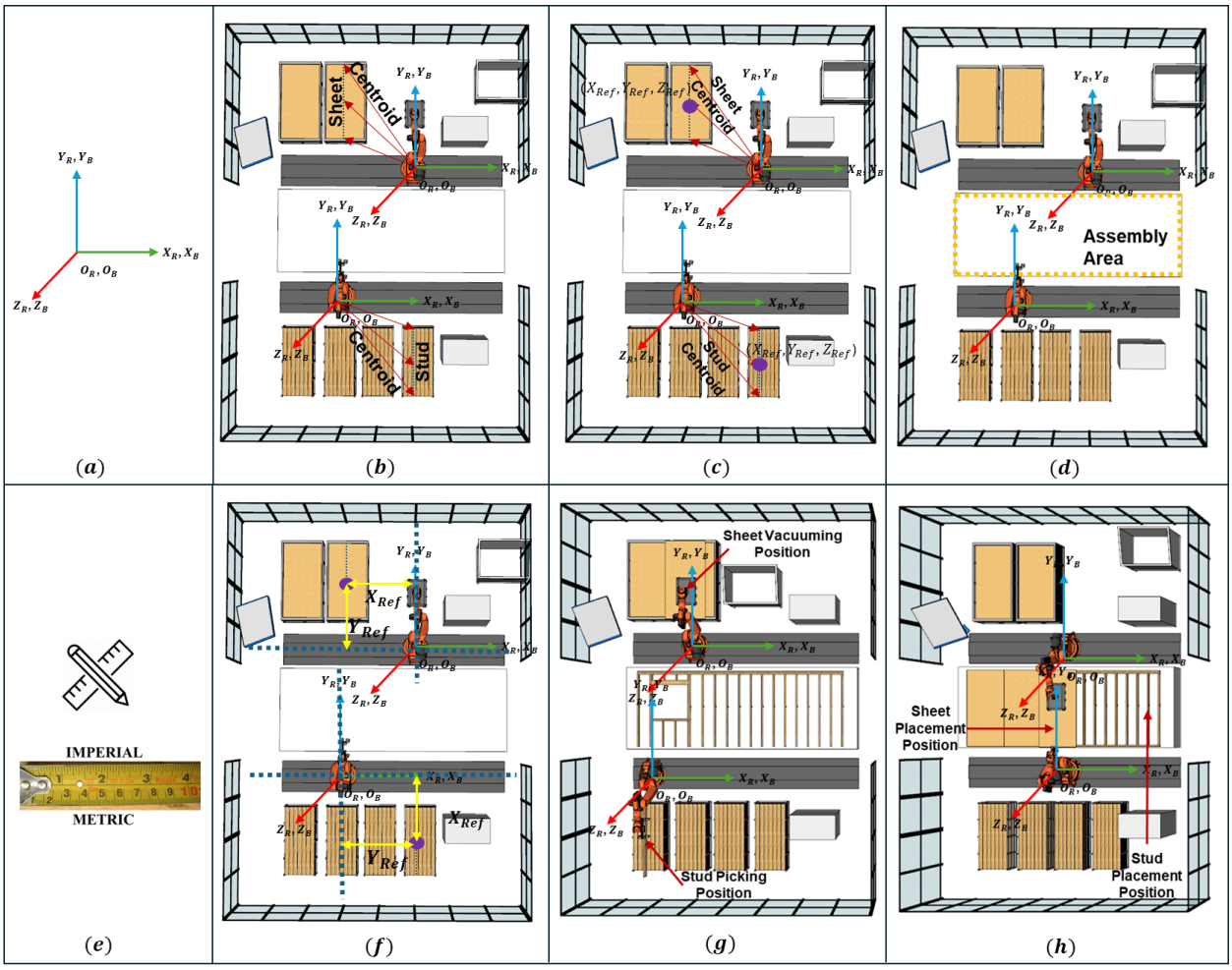


Figure 55. Sub-Steps for Determining the Framing Target Locations

At this stage, the focal point is the definition of target locations for the nailing operation. These nailing target locations are specific points where the robot executes nailing tasks to join elements within the frame, and are determined relative to the robot's coordinate system. The identification of these nailing target locations is directly derived from the previously established framing target locations following Figure 56. This derivation is accomplished through the application of

Equations (6), (7), and (8), which are formulated based on the nailing schedule prescribed for the connections between studs and tracks (both top and bottom) and between studs and sheets (top) as detailed by Allen et al. (2017). In Equation (8), the two distinct values for N_z reflect the dual nailing points along the z-axis. A visual representation of these nailing target locations within a frame structure is illustrated in Figure 57. This representation aids in visualizing the exact points where nailing operations are to be conducted by the robot, thus ensuring precision and accuracy in the assembly process.

$$N_x^{ij} = x_t^{plate_i} - \frac{thickness^{plate_i}}{2} \tag{6}$$

$$N_y^{ij} = y_t^{stud_j} \tag{7}$$

$$N_z^{ij} = \begin{cases} z_t^{plate_i} - \frac{width^{plate_i}}{6} \\ z_t^{plate_i} + \frac{width^{plate_i}}{6} \end{cases} \tag{8}$$

Where:

N_x^{ij}= x component of the nailing target location between plate *i* and stud *j*

N_y^{ij}= y component of the nailing target location between plate *i* and stud *j*

N_z^{ij}= z component of the nailing target location between plate *i* and stud *j*

$x_t^{plate_i}$= x component of the centroid of plate *i*

$z_t^{plate_i}$= z component of the centroid of plate *i*

$y_t^{stud_i}$= y component of the centroid of stud *j*

$width^{plate_i}$= width of plate *i*

$thickness^{plate_i}$= thickness of plate *i*

4.5.5 Identifing the Orientation of the Frame in Relation to the Robot

In this phase, the critical task is establishing the frame's placement orientation, which dictates the orientation of its sub-elements relative to the robot's x-axis. Orientations can be perpendicular, parallel, or at a specific angle to the x-axis. For instance, a parallel orientation aligns the frame's

length with the robot's x-axis, while a perpendicular orientation aligns the frame’s height with the robot’s y-axis. Once determined, the orientation and longitudinal axis of each element are incorporated into the controller. This information guides the directional placement of elements during framing and nailing operations, ensuring they align precisely with predefined specifications.

4.5.6 Defining and Creating the Subroutines for the Robot

The development of subroutines for framing and nailing is essential for automating these tasks in a robotic system. These subroutines, which involve picking up, moving, and operating grippers on construction elements like plates and studs, process inputs from the element's dispensing location and type, and target locations for framing and nailing. The robot's actions depend on the type of element and its sequence in the construction process, executing the relevant subroutine for each task. Each subroutine cycle corresponds to the framing of one element. Figure 56 show the nailing patterns for different types of wood studs, Figure 57 demonstrates the nailing targets coordinates, Figure 58 illustrate the typical framing and sheathing operations, and Figure 59 presents a flowchart of the framing subroutine, outlining the process flow and decision-making in the operation.

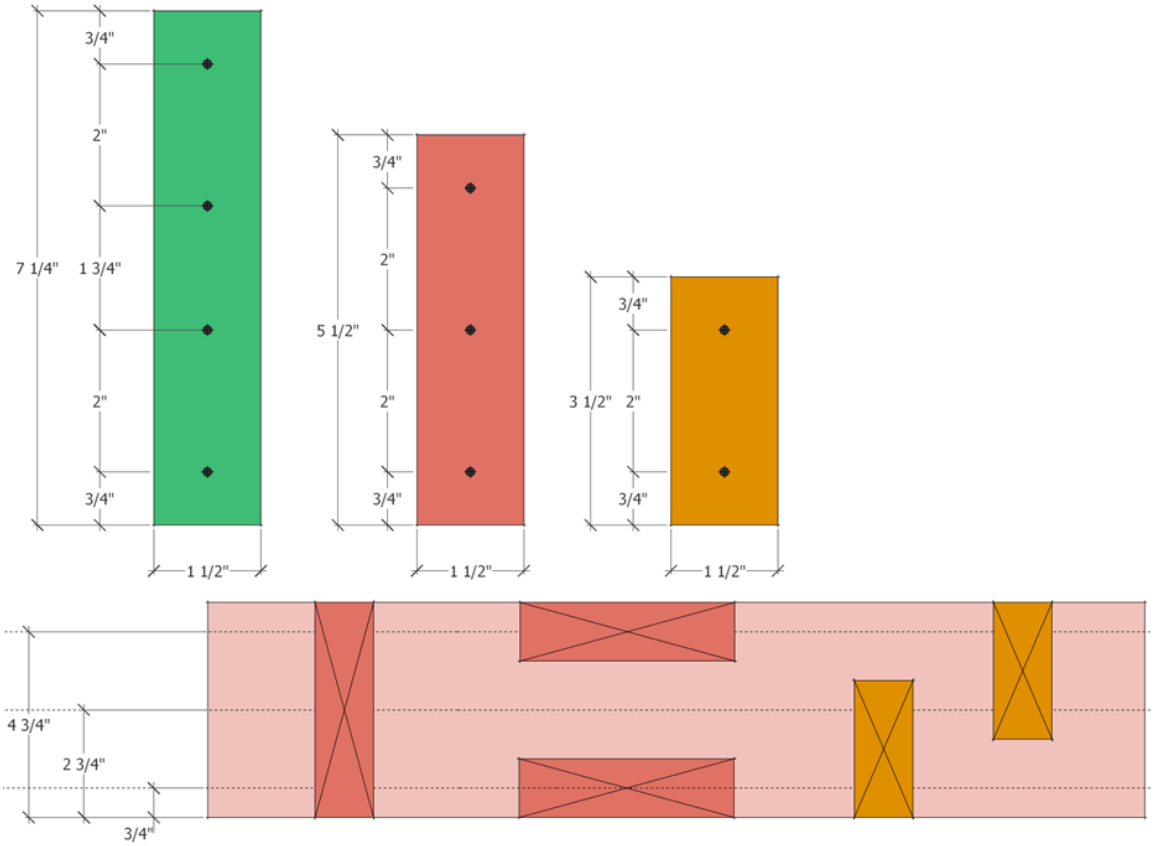


Figure 56. Nailing Patterns for 2x8, 2x6, and 2x4 inch Wood Studs

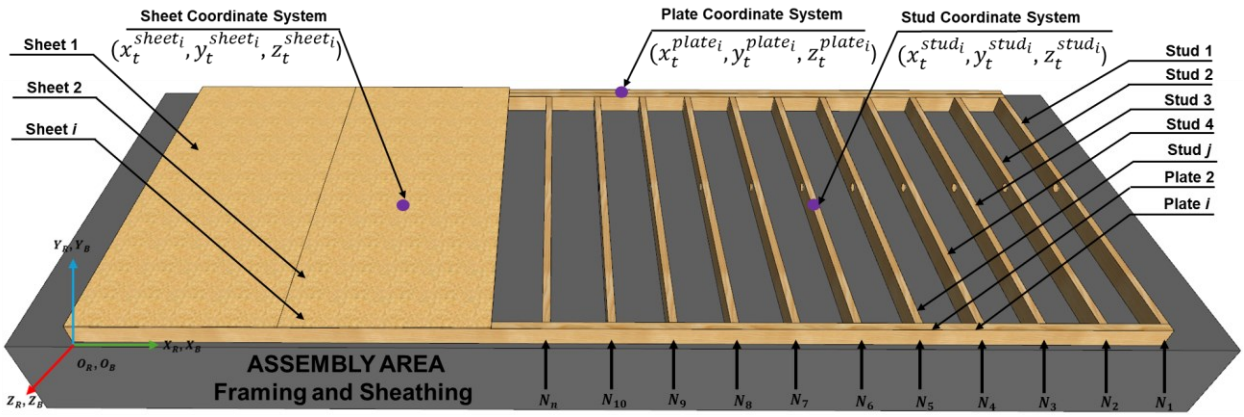


Figure 57. Definition of the Nailing Positions

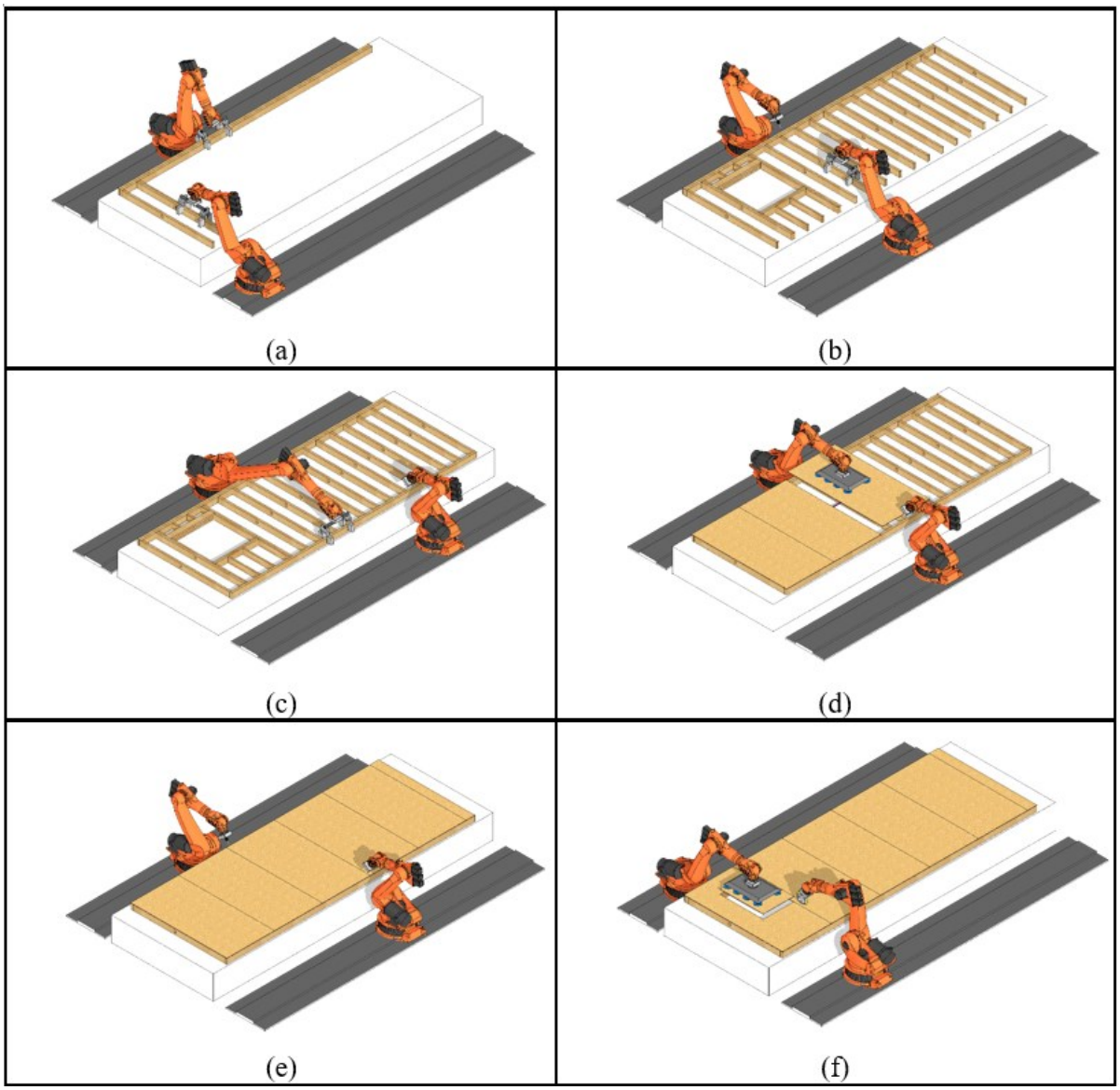


Figure 58. Framing and Sheathing Operations

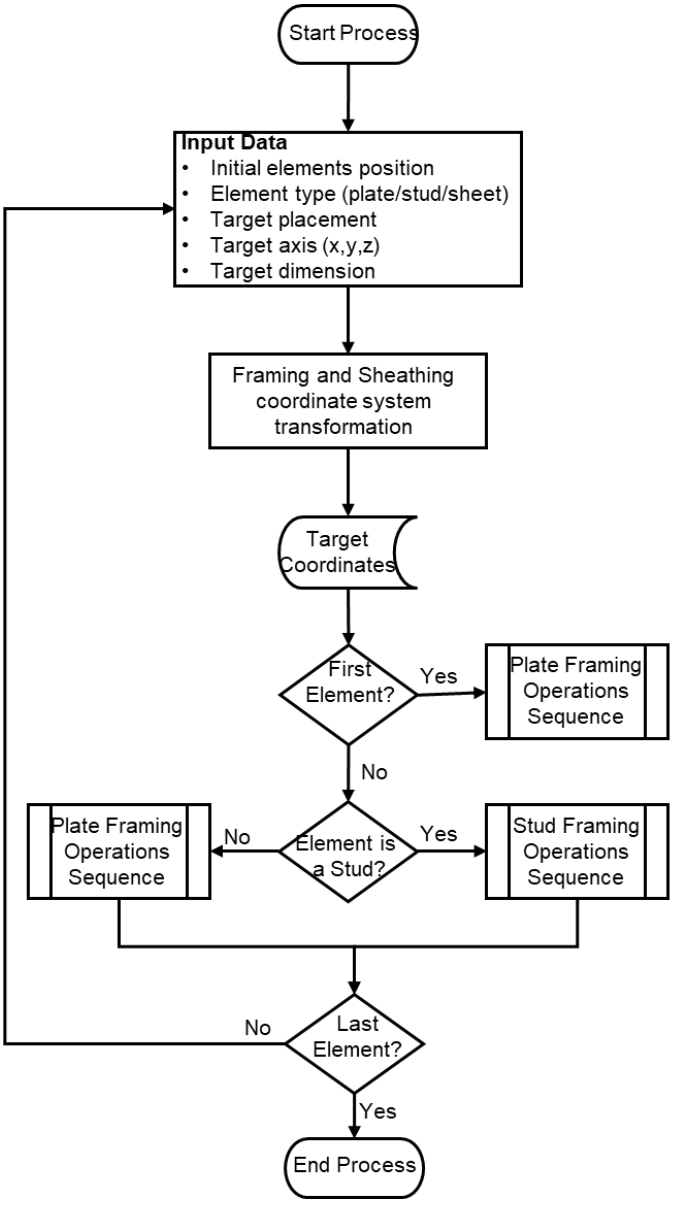


Figure 59. Framing Subroutines Algorithm Flowchart

4.5.7 Coding the Control System and BIM Data into the Robot Controller

The robotic system controllers utilize specific instructions, including subroutines and Inverse Kinematics (IK), to guide assembly processes. IK calculates the joint angles required based on the end-effector's desired position. These functionalities, along with framing and nailing subroutines, are either embedded directly in the controller or used as auxiliary modules to enhance functionality, with this research favoring the latter for increased modularity. BIM data, detailing

names, placements, axes, and dimensions of construction elements is crucially integrated into the controller. This integration allows for precise determination of target locations used in the subroutines, ensuring accurate and efficient execution of assembly operations by providing detailed information on each element.

4.6 Implementation and Validation of the Experimental Method

The practical implementation of the experimental method and the outputs obtained are detailed in this section. To assess the method's effectiveness and time efficiency, seven BIM models were used: one for development and six for validation. The initial implementation employed a baseline model (MBase), a wood-framed structure with four walls created using Autodesk Revit (Version 2024), as shown in Figure 60. This Base model served as the benchmark for the methodology.

Figure 60. BIM Base Model (MBase) for Algorithm Development

4.6.1 Implementation Software and System Setup

In the implementation of the proposed method, the Prolog programming language variant selected was B-Prolog. This choice was made due to B-Prolog's utilization of HC representation, aligning with the method's requirements. Key factors contributing to the selection of B-Prolog include its built-in rationalization capabilities and adaptability with C and Java programming languages.

In this stage, a simulation file representing the robotic system components and BIM data from previous phases is created. The development and execution of this simulation involve four steps:

Step 1: Mapping BIM building elements to construction materials in the simulation

Here, the BIM building elements (like studs, tracks, and sheets) are automatically correlated with the corresponding construction materials (such as wood or steel) within the simulation environment. Attributes including dimensions, density, and quantities of each simulated element are derived from the BIM data.

Step 2: Defining initial position and orientation of construction materials in the simulation environment

The elements to be placed within the simulation are arranged sequentially (bottom plate → studs → top plate). Each element is parallel to the others, spaced consistently to prevent misalignment and collisions during transfer. The longitudinal axis of these elements is aligned for a straight path transfer, positioned on their narrower side surfaces to streamline the robot's grasping operations and eliminate the need for element rotation during assembly.

Step 3: Generating the simulation file

This involves dynamically creating a file that encompasses information from preceding phases, such as 3D models of robotic system components, building elements, controllers, and properties.

The generation and loading of this file are executed programmatically, adhering to the model directory structure of the robotic simulator.

Step 4: Executing the simulation file

With the simulation file, containing both the assemblies and embedded controllers for the robotic system, being generated, the system is set for execution. The simulation initiates automatically upon executing this file. During the simulation, the robotic system carries out the predefined assembly subroutines as programmed.

These steps collectively facilitate the creation of a detailed and functional simulation environment where the robotic system's performance can be accurately modeled and assessed, ensuring the efficiency and accuracy of the assembly process in real-world applications.

4.6.2 Case Studies

In this phase, the simulation performance of the proposed methodology is thoroughly assessed using the applicable case studies. This evaluation is categorized into three distinct areas:

(1) **BIM-Simulator Integration Assessment**

This involves comparing the information generated in the simulation environment, such as building materials, with the building elements in the BIM design model. The primary objective is to verify the accuracy of the mapping and generation processes. This ensures that the simulation environment accurately reflects the actual BIM design model.

(2) **Assembly Process Simulation Evaluation**

The focus here is on assessing the simulation of the assembly process. This evaluation determines the effectiveness of the construction algorithms (like framing and nailing subroutines) that are

based on the BIM data. It identifies any deficiencies or inaccuracies in the robotic system components or the BIM data that might have led to suboptimal performance. The visualization feature of the simulation allows for detailed inspection of specific operations performed by the robotic system. Additionally, spatial and temporal data are analyzed to pinpoint any unexpected behaviors of the robotic system components during construction operations.

(3) **Collision Detection Analysis**

This assessment deals with identifying any potential collisions between components of the robotic system or with objects within its workspace. Such collisions can impede or completely halt the assembly operations. The detection of collisions is crucial to ensure the smooth operation of the robotic system.

This evaluation process is crucial for assessing the suitability of BIM data, the effectiveness of construction operations, and the appropriateness of robotic system components for the assembly process. It also establishes a feedback loop for possible reselection or modification of system components or BIM data to improve assembly efficiency and accuracy.

The methodology was implemented using the Unity game engine by Unity Technologies, known for its robust simulation capabilities, advanced graphics, and support for C# scripting. Unity is effective for rapid prototyping and ensures consistent behavior in simulations under the same controls (Staranowicz and Mariottini., 2011). It includes pre-configured robots and 3D objects, with a user-friendly interface that supports customization and interaction with the robotic system. For IK, the IKFast solver from OpenRAVE was used to compute joint parameters, crucial for robot motion planning and control, based on the end-effector's position and orientation. IKFast is recognized for its precision (accurate to seven decimal points), speed (7 to 50 milliseconds

depending on model precision), and ability to handle different kinematic needs like position, orientation, or both. Although IKFast was chosen for its efficiency in generating fast and precise IK solutions using analytical rather than numerical methods, alternative solvers like IKPy are also viable. IKFast excels by using closed-form solutions based on a robot arm's specific geometric and kinematic properties, ensuring fast and efficient calculations. IKFast has been integrated into the robotics framework and simulation environment using Unity and ROS to provide fast and accurate inverse kinematics solutions for robotic manipulators. This integration is especially beneficial for applications that demand real-time or near-real-time responses, including industrial automation, task management, and movement planning.

The Unity world file was developed using B-Prolog, a logic programming platform known for its unification, backtracking, and rewriting capabilities. B-Prolog was chosen for its compatibility with C# and Java, facilitating a bi-directional interface, and its seamless integration with BIM design model data extraction algorithms. Although B-Prolog was used for this project, the creation of the Unity world file is not limited to a specific programming language and could also be achieved using other languages like Python or Java.

According to the Unity User Guide, specific minimum computational requirements are needed to run the Unity simulation software effectively. For this study, experimental tests were conducted on a system exceeding these requirements. The test platform was a Windows 11 (64-bit) desktop with an Intel® Core™ i7-12700 processor at 2.10 GHz, 32 GB of SDRAM, and a NVIDIA® GeForce RTX™ 3080 graphics card, providing a robust environment for the Unity-based experimental simulations.

To test and validate the proposed method, six additional BIM instance models, M1 through M6, were evaluated as shown in Figure 61 and Figure 62. The tests aimed to assess method validity

and the applicability of new assembly rules on previously unanalyzed BIM models. Models M1 to M3, made primarily of wood, and M4 to M6, constructed of light gauge steel (LGS), represented residential projects. All models were created using Autodesk Revit, with models M1, M2, M4, and M5 developed to a Level of Development/Detail (LOD) of 300, while M3 and M6 were detailed to LOD 400, offering a more comprehensive representation suitable for fabrication and assembly. This range of LODs tested the method's adaptability across different project complexities and details.

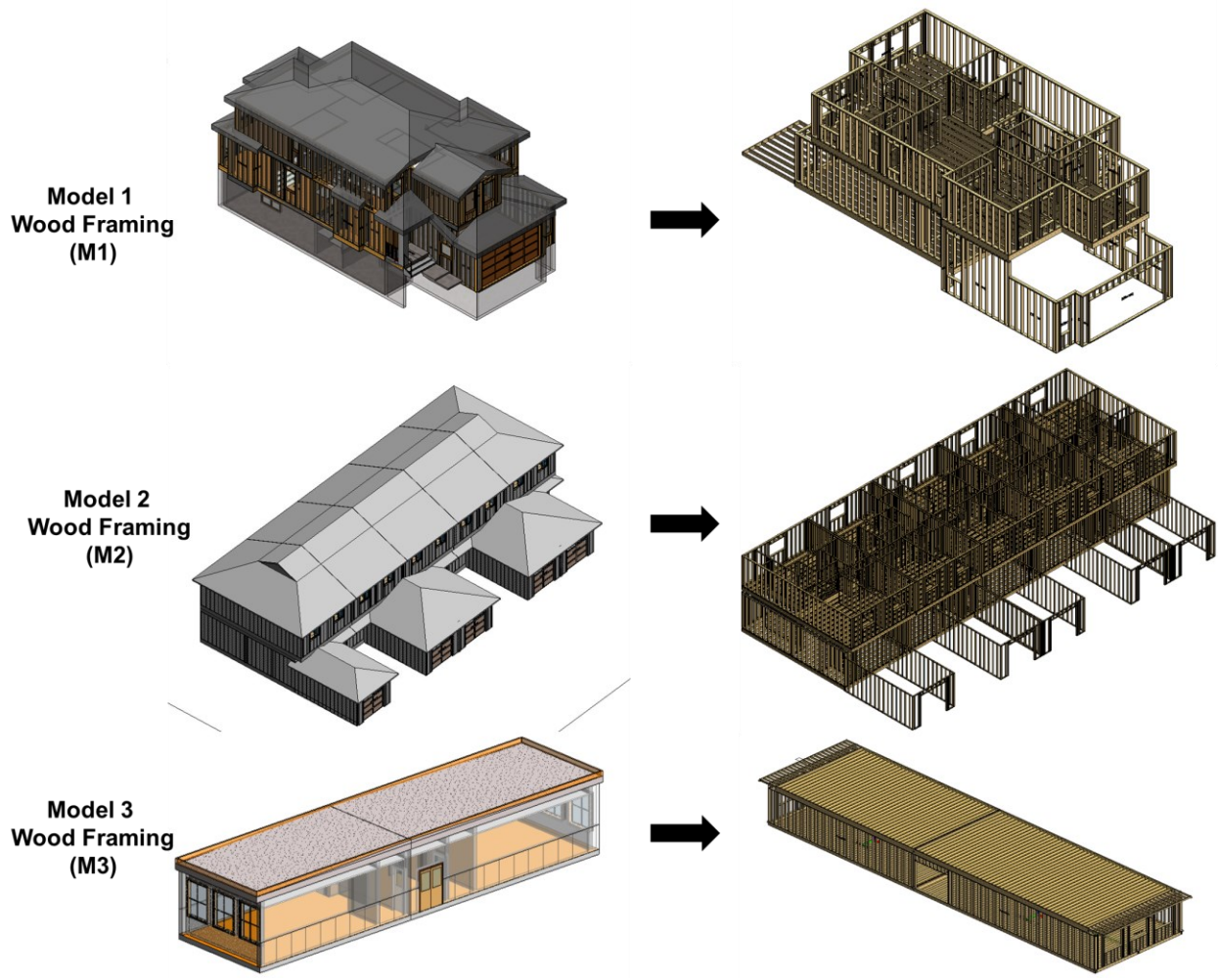


Figure 61. BIM Testing Models with Wood Framing M1, M2, M3

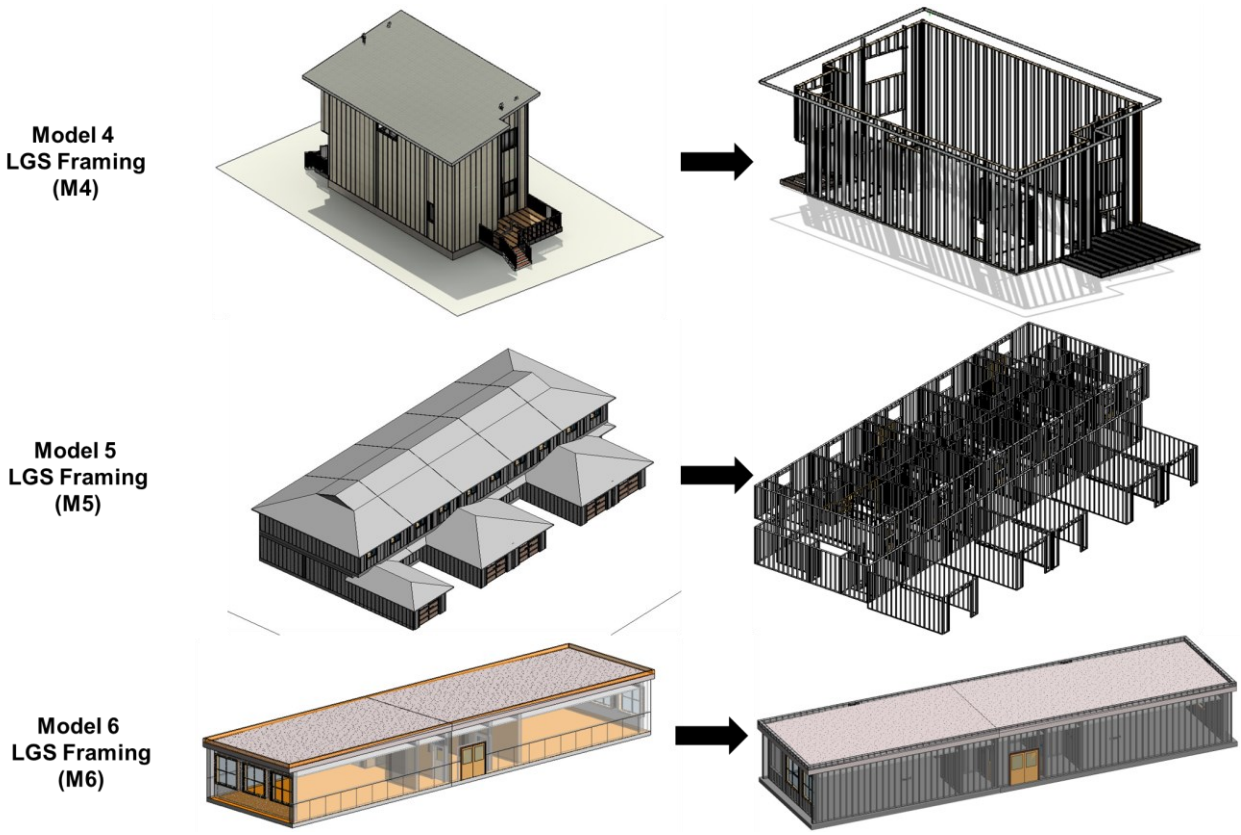


Figure 62. BIM Testing Models with Steel Framing M4, M5, M6

Step 1: Transforming BIM instance models into rule-based facts

Initially, each of the testing BIM models was exported from Autodesk Revit to an IFC file format. Subsequently, these IFC files were transformed into rule-based facts using the IfcOpenShell_Python_BlenderBIM_Addon prototype, as shown in Figure 63, which shows instances from an IFC file and their corresponding rule-based facts. Following conversion, a preprocessing phase involving a Python algorithm was applied to the rule-based facts, detailed in Figure 64. Figure 65 further illustrates the transformation of rule-based facts before and after preprocessing. This sequence of transformations is essential for accurately and efficiently extracting and analyzing data from the BIM models.

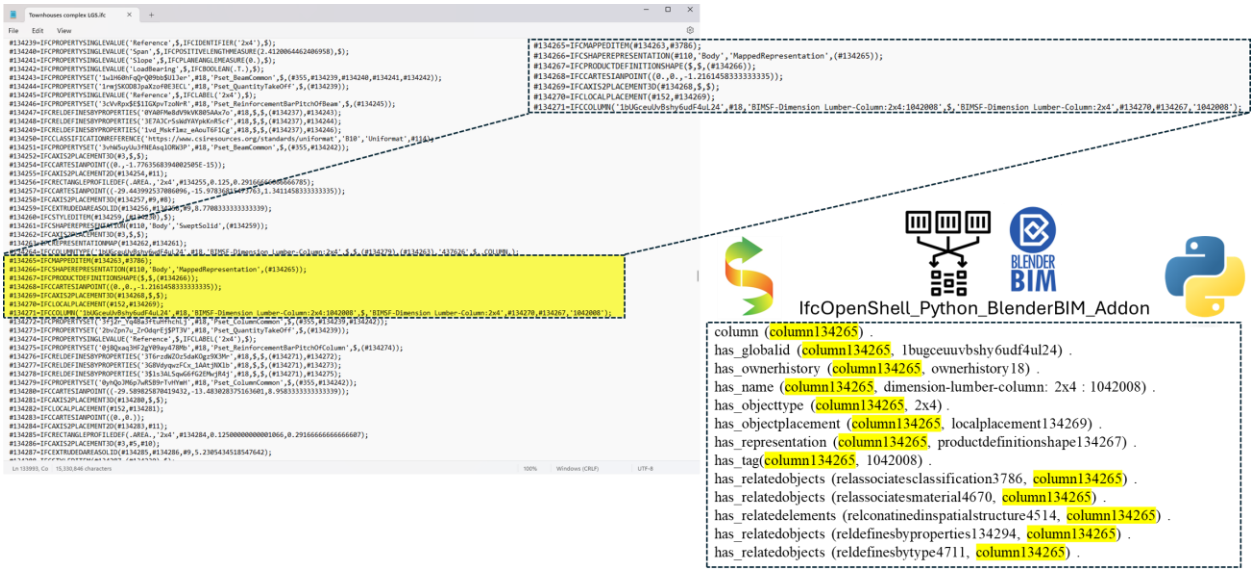


Figure 63. Sample Conversion from IFC to Rule-Based Facts

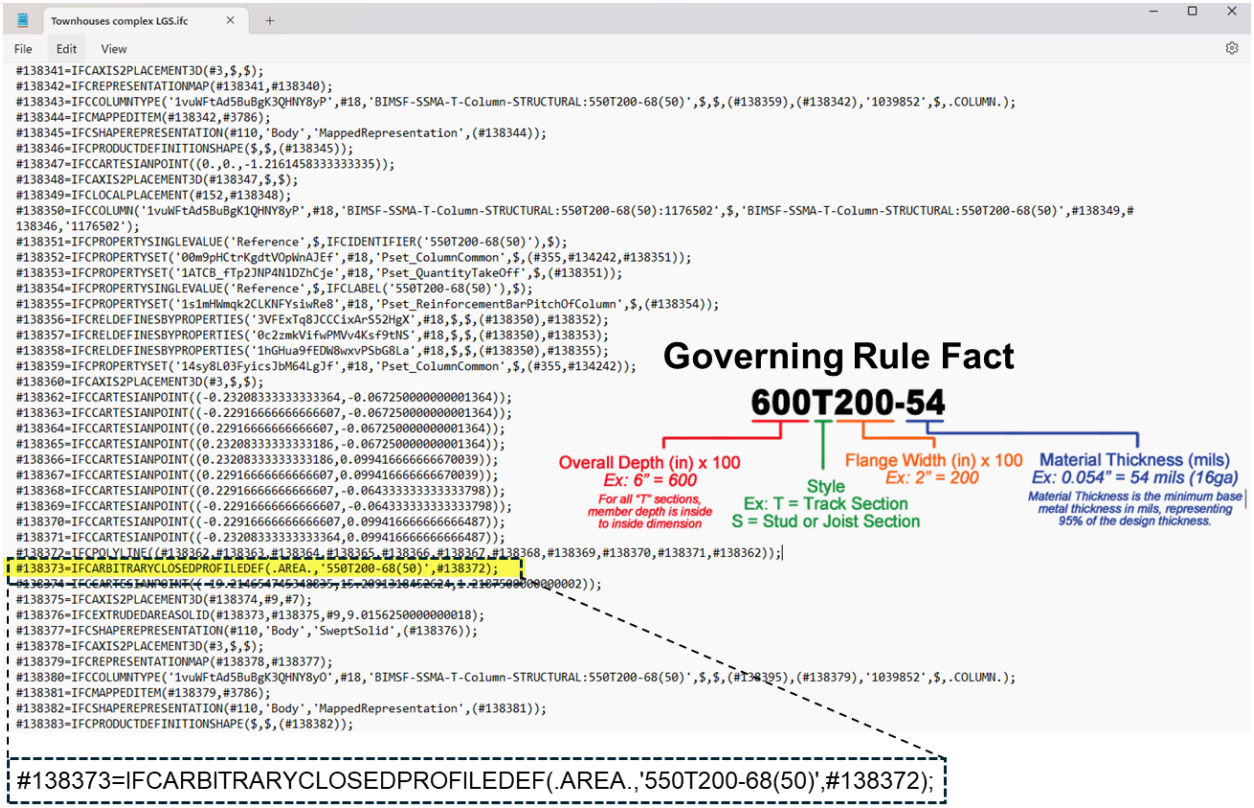


Figure 64. Sample of Preprocessed Rule-Based Facts

Steps 2-3: Process of developing and executing rule-based facts

Algorithms for rule-based facts were formulated to extract and infer data from BIM instance models. This development was based on a randomly selected subset of rule-based facts from the Base model. Initially, a set of 59 rule-based facts was established, comprising 26 extraction rules, 17 derivation rules, 8 processing rules, and 8 fixation rules as detailed in

Table 6.

Subsequent to the development of these rule-based facts, they were put to the test using the Base model during the rule-based rationalization stage. This testing procedure was not only confined to the Base model but was also replicated across six additional test models (M1-M6), employing a similar approach. This systematic application and validation of the rule-based facts across various models underscore the versatility and adaptability of the developed rule-based facts in extracting and interpreting data from different BIM instance models.

Table 6. List of Rules Developed Using the Base Model

1	A	Interior Wall	IfcWallStandardCase	Swept Solid	Rectangle	Dimensions
2	A					Quantity
3	A					Material
4	A	Rough Opening (Interior Wall)	IfcOpeningElement	Swept Solid	Rectangle	Quantity
5	A					Dimensions
6	B					Area
7	B					Total Area
8	B	Interior Wall	IfcWallStandardCase	Swept Solid	Rectangle	Area
9	B					Net Area
10	A	Exterior Wall	IfcWallStandardCase	Swept Solid	Rectangle	Dimensions
11	A					Quantity
12	A					Material
13	A		IfcOpeningElement		Rectangle	Quantity

14	A	Rough Opening (Exterior Wall)		Swept Solid		Dimensions
15	B					Area
16	B					Total Area
17	B	Exterior Wall	IfcWallStandardCase	Swept Solid	Rectangle	Area
18	B					Net Area
19	A	Floor	IfcSlab	Swept Solid	Rectangle	Dimensions
20	A					Quantity
21	A					Material
22	A	Rough Opening (Floor)	IfcOpeningElement	Swept Solid	Rectangle	Quantity
23	A					Dimensions
24	B					Area
25	B					Total Area
26	B	Floor	IfcSlab	Swept Solid	Rectangle	Area
27	B					Net Area
28	A	Roof	IfcRoof	Swept Solid	Rectangle	Dimensions
29	A					Quantity
30	A					Material
31	B					Area
32	A	Column	IfcColumn	Swept Solid	Rectangle	Dimensions
33	A					Quantity
34	A					Material
35	B					Volume
36	B					Weight
37	A	Beam	IfcBeam	Swept Solid	Rectangle	Dimensions
38	A					Quantity
39	A					Material
40	B					Volume
41	B					Weight
42	A	Rough Opening	IfcOpeningElement		Rectangle	Dimensions

43	A			Swept Solid		Quantity
44	C	Interior Wall	IfcWallStandardCase	Wood/Steel		Material
45	C			Stud/Sheet	Rectangle	Type
46	C	Exterior Wall	IfcWallStandardCase	Wood/Steel		Material
47	C			Stud/Sheet	Rectangle	Type
48	C	Floor	IfcSlab	Wood/Steel		Material
49	C			Stud/Sheet	Rectangle	Type
50	C	Roof	IfcRoof	Wood/Steel		Material
51	C			Stud/Sheet	Rectangle	Type
52	D	Interior Wall	IfcWallStandardCase	Wood	Nails	Fixation
53	D			Steel	Screws	
54	D	Exterior Wall	IfcWallStandardCase	Wood	Nails	Fixation
55	D			Steel	Screws	
56	D	Floor	IfcSlab	Wood	Nails	Fixation
57	D			Steel	Screws	
58	D	Roof	IfcRoof	Wood	Nails	Fixation
59	D			Steel	Screws	
Rule A: Extraction; **Rule B:** Derivation; **Rule C:** Processing; **Rule D:** Fixation						

Step 4: Assessment metrics

The experimental outcomes for testing six models based on key performance indicators – sensitivity, specificity, precision, accuracy, and F1 measure – are discussed. Initially, Table 7 provides an in-depth analysis of the results for the development model, designated as MBase. This is followed by a comprehensive presentation of results from the validation tests conducted using six distinct BIM instance models (M1-M6) as shown in Appendix A.

A consolidated overview of the performance results, including the calculated values for the key performance indicators, for all case studies is shown in

Table 8. The final output is a report generated in a .csv format.

Table 7. Results of the Base Model (Mbase)

Component/ Element	**Item**	**# of Relevant RBF**	**# of Extracted RBF**	**# of Correctly Extracted RBF**
Exterior Wall	Quantity (Integer)	4	4	4
Exterior Wall	Dimensions (L,W,H)	12	12	12
Exterior Wall	Material (String)	12	12	12
Exterior Wall	Rough Opening Quantity (Integer)	8	8	8
Exterior Wall	Rough Opening Dimensions (L,W,H)	24	24	24
Interior Wall	Quantity (Integer)	5	5	5
Interior Wall	Dimensions (L,W,H)	15	15	15
Interior Wall	Material (String)	15	15	15
Interior Wall	Rough Opening Quantity (Integer)	2	2	2
Interior Wall	Rough Opening Dimensions (L,W,H)	4	4	4
Slab	Quantity (Integer)	2	2	2
Slab	Dimensions (L,W,H)	4	4	4
Slab	Material (String)	3	3	3
Slab	Rough Opening Quantity (Integer)	-	-	-
Slab	Rough Opening Dimensions (L,W,H)	-	-	-

Roof	Quantity (Integer)	1	1	1
Roof	Dimensions (L,W,H)	3	3	3
Roof	Material (String)	3	3	3
Roof	Rough Opening Quantity (Integer)	-	-	-
Roof	Rough Opening Dimensions (L,W,H)	-	-	-
Column	Quantity (Integer)	85	85	85
Column	Dimensions (L,W,H)	255	255	255
Column	Material (String)	255	255	255
Beam	Quantity (Integer)	62	62	62
Beam	Dimensions (L,W,H)	186	186	186
Beam	Material (String)	186	186	186
Total Instances		1142	1142	1142

Table 8. Summary of the Testing Results

Type	**Model**	**Sensitivity (*Sn.*) %**	**Specificity (Sp.) %**	**Precision (Pr.) %**	**Accuracy (Acc.) %**	**F1-measure (F1m.) %**
Wood	Base	100	100	100	100	100
Wood	1	91	91	84	91	87
Wood	2	91	91	100	91	95
Wood	3	99.6	99.6	100	99.6	100
LGS	4	93.2	93.2	66.3	93.2	77.5
LGS	5	91	91	100	91	95
LGS	6	99.6	99.6	100	99.6	100

Steps 5-6: Performance enhancement through rule-based stepwise refinement and assembly rules extension

Following the evaluation of the test models, enhancements were made to both the rule-based stepwise refinement algorithm and the assembly rules to improve rationalization performance.

Within the stepwise refinement algorithm, modifications were applied to two specific predicates, has_cfsfaces# and has_segments#. These adjustments were designed to optimize the performance of Prolog's basic represntation and SOL functions. The improvement was accomplished by revising the algorithm to remove the numbering sequence from the two modified predicates. Subsequently, an additional 22 new assembly rules were incorporated into the original assembly rules set – comprising 16 extraction rules and 6 processing rules – to support the rationalization capabilities of the ruleset (Table 9).

Following the implementation of Steps 5 and 6, the refined method demonstrated proficiency in extracting and inferring comprehensive information from the BIM instance models. The method's ability to adapt to these variations underscores its flexibility and effectiveness in analyzing and processing complex BIM data.

Table 9. Additional Rules List from the Testing Models (M1-M6)

Rule #	**Rule Type**	**Component/ Element**	**Source**	**Representation**	**Definition**	**Information**
1	A	Interior Wall	IfcWallStandardCase	Swept Solid	Arbitrary Closed	Dimensions
2	A			Clipping	Rectangle	
3	A			Clipping	Arbitrary Closed	
4	A		IfcWall	Clipping	Rectangle	Material
5	A	Exterior Wall	IfcWallStandardCase	Clipping	Rectangle	Dimensions
6	A			Clipping	Arbitrary Closed	
7	C					Material
8	C	Exterior Wall	IfcWall			Object Type
9	C	Rough Opening	IfcOpeningElement			

10	A	Floor	IfcSlab	Swept Solid	Arbitrary Closed	Dimensions
11	C					Material
12	A	Roof	IfcRoof	Facetedbrep		Dimensions
13	A		IfcSlab	Swept Solid	Arbitrary Closed	
14	A		IfcSlab			Material
15	A	Rough Opening	IfcOpeningElement	Swept Solid	Arbitrary Closed	Dimensions
16	A			Facetedbrep		
17	C	Column	IfcColumn	Swept Solid		Material
18	A	Beam	IfcBeam	Swept Solid	Rectangle	Dimensions
19	A			Swept Solid	Arbitrary Closed	
20	A			Clipping	Arbitrary Closed	
21	A			Clipping	Rectangle	
22	C			Facetedbrep		Material
Rule A: Extraction; **Rule B:** Derivation; **Rule C:** Processing; **Rule D:** Fixation						

4.6.3 Testing the Reasonableness of Assembly Sequence

The Base model and the six testing models were drafted and tested using Autodesk Revit and then simulated in Dynamo API using the ASP Algorithm to identify the locations of prefabricated elements within the reference frame of the Revit model, as well as to establish their respective assembly coordinates and sequence. The outcomes of this application for the Base model shown in Figure 67 are detailed in Table 10. The results demonstrate the ASP algorithm's capability to generate a logical assembly sequence for the Base model. Initially, the framework is constructed (encompassing components 1–97 as listed in Table 15 for Panels 1 to 4), followed by the enclosure of wall panels (involving OSB sheets for each frame). The framework is constituted by bottom and top beams (tracks) and columns (studs).

A wood framed wall consists of vertically aligned studs, typically measuring 2 x 4 or 2 x 6, spaced uniformly at either 16 inches or 24 inches at the center. These studs are anchored between top and bottom tracks, forming the structural framework as shown in Figure 65 and Figure 66. The top track could be either single or double.

Following is a list of the most common framing members in a typical wood wall frame:

1. **Stud** - vertical framing member that runs from bottom plate to the top plate.
2. **Bottom track** - a horizontal framing member which runs along the bottom of the wall.
3. **Top plate** (single or double) - a horizontal framing member which runs along the top of the wall. A double top plate is most common on load-bearing walls unless the roof rafters or trusses and floor joists stack directly over the studs in the wall.
4. **Header** - a horizontal member which spans an opening for a window or door.
5. **King stud** - stud to left or right side of a window or door that is continuous from bottom plate to the top plate.
6. **Jack stud** - stud to the left or right of a window or door that runs from the bottom plate to the underside of a header.
7. **Cripple stud** - a stud located either above or below a framed opening, that does not run the full height of the wall.
8. **Sill plate** - a horizontal plate below a window unit.

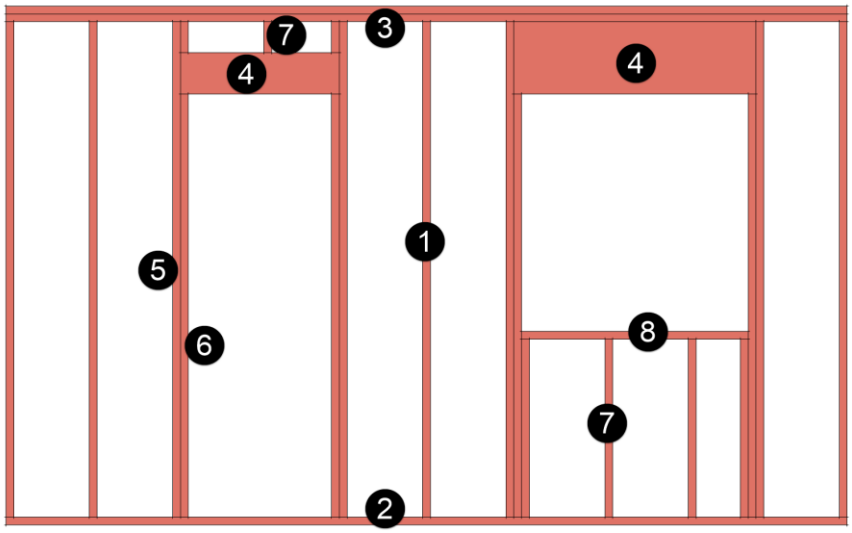


Figure 65. Common Framing Members for a Typical 2x6 Wall

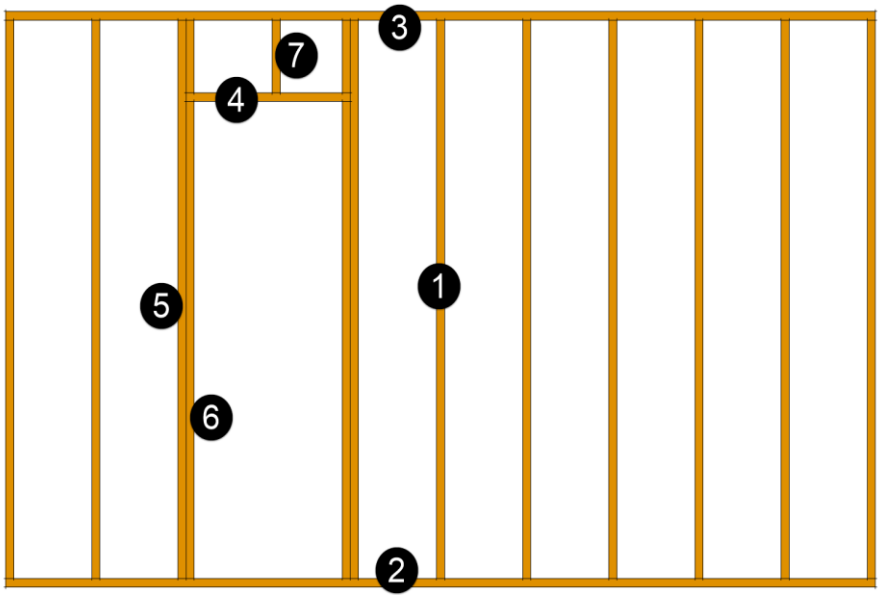


Figure 66. Common Framing Members for a Typical 2x4 Wall

These BIM design models for the panels were created using Autodesk Revit (Version 2024). The specific dimensions and details for each test panel from the Base model are provided throughout Figure 68-Figure 74. This comprehensive approach to model creation ensured a robust and varied set of assembly scenarios with respect to Table 11-Table 14 respectively, for effectively testing and validating the proposed methodology within the experimental framework.

For the experimental testing aligned with the proposed methodology, four exterior wall multi-panels were developed and utilized as test cases extracted from the Base model. These test cases encompass:

1. **Rectangular Frame Design**: This design incorporates a rectangular frame as window or door component assembly built with 2"x6" elements.
2. **Wood Structure Design**: This model consists of a wood structure with four exterior walls. The wall frames in this design have a height of 8 feet, with intermediate studs placed at 16-inch intervals in the center. The studs used for the frames are sized at 2"x6".

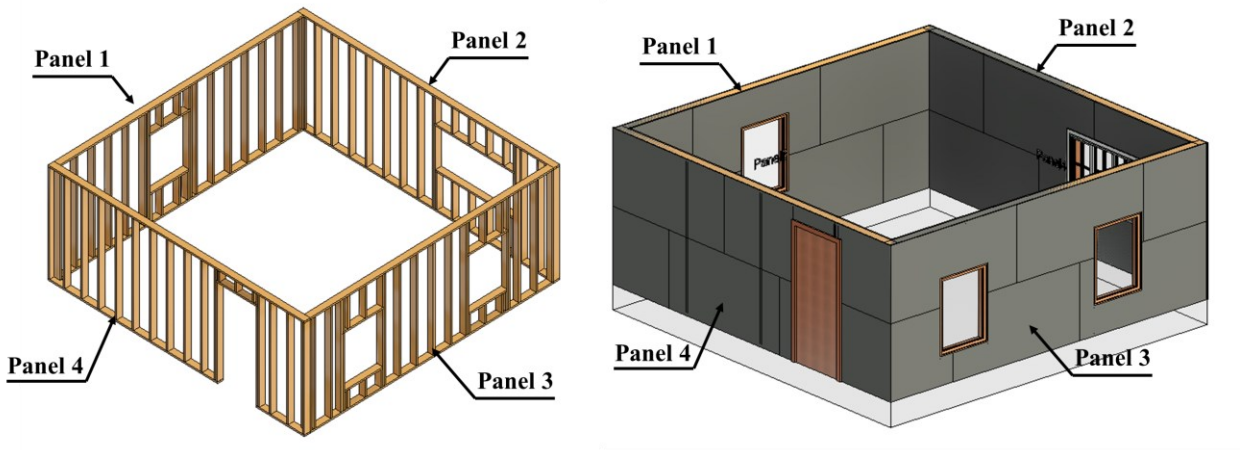


Figure 67. Base Model with Framing and Sheathing

Table 10. Assembly Sequence and Coordinates of the Prefabricated Base Model Components

Multi-Panel Exterior Walls Combo (Unit: Meters)	Panel ID	Sequence	Components	Coordinates	Panel ID	Sequence	Components	Coordinates
	4	1	Bottom Track	3.04, 1.21, 0.044	1	30	Bottom Track	3.04, 1.21, 0.044
		2	Stud 1	0.4, 1.21, 0.044		31	Stud 1	0.4, 1.21, 0.044
		3	Stud 2	0.8, 1.21, 0.044		32	Stud 2	0.8, 1.21, 0.044
		4	Stud 3	1.2, 1.21, 0.044		33	Stud 3	1.2, 1.21, 0.044
		5	Stud 4	1.6, 1.21, 0.044		34	Stud 4	1.6, 1.21, 0.044
		6	Stud 5	2.0, 1.21, 0.044		35	Window Assembly	2.2, 1.21, 0.044
		7	Door Assembly	2.4, 1.21, 0.044		36	Stud 5	2.6, 1.21, 0.044

		8	Stud 6	2.8, 1.21, 0.044		37	Stud 6	3, 1.21, 0.044
		9	Stud 7	3.2, 1.21, 0.044		38	Window Assembly	3.6, 1.21, 0.044
		10	Stud 8	3.6, 1.21, 0.044		39	Stud 7	4, 1.21, 0.044
		11	Stud 9	4.0, 1.21, 0.044		40	Stud 8	4.8, 1.21, 0.044
		12	Stud 10	4.4, 1.21, 0.044		41	Stud 9	5.2, 1.21, 0.044
		13	Stud 11	4.8, 1.21, 0.044		42	Stud 10	5.6, 1.21, 0.044
		14	Stud 12	5.2, 1.21, 0.044		43	Stud 11	6, 1.21, 0.044
		15	Stud 13	5.6, 1.21, 0.044		44	Stud 12	6.4, 1.21, 0.044
		16	Stud 14	6.0, 1.21, 0.044		45	Stud 13	6.8, 1.21, 0.044
		17	Stud 15	6.4, 1.21, 0.044		46	Stud 14	7.2, 1.21, 0.044
		18	Window Assembly	6.8, 1.21, 0.044		47	Top Track	3.04, 3.65, 0.044
		19	Stud 16	7.2, 1.21, 0.044		48	OSB Sheet 1	0.6, 1.21, 0.025
		20	Stud 17	7.6, 1.21, 0.044		49	OSB Sheet 2	1.8, 2.13, 0.025
		21	Top Track	3.04, 3.65, 0.044		50	OSB Sheet 3	3, 0.91, 0.025
		22	OSB Sheet 1	0.6, 1.21, 0.025		51	OSB Sheet 4	4.2, 1.21, 0.025
		23	OSB Sheet 2	1.8, 1.21, 0.025		52	OSB Sheet 5	5.4, 1.21, 0.025
		24	OSB Sheet 3	3, 1.82, 0.025		53	OSB Sheet 6	6.6, 1.21, 0.025
		25	OSB Sheet 4	4.2, 1.21, 0.025		54	OSB Sheet 7	7.2, 1.21, 0.025
		26	OSB Sheet 5	5.4, 1.21, 0.025				
		27	OSB Sheet 6	6.6, 1.21, 0.025				
		28	OSB Sheet 7	7.2, 1.82, 0.025				
		29	OSB Sheet 8	7.2, 0.6, 0.025				
Multi-Panel Exterior Walls Combo (Unit: Meters)	**Panel ID**	**Sequence**	**Components**	**Coordinates**	**Panel ID**	**Sequence**	**Components**	**Coordinates**
	2	55	Bottom Track	3.04, 1.21, 0.044	**3**	81	Bottom Track	3.04, 1.21, 0.044
		56	Stud 1	0.4, 1.21, 0.044		82	Stud 1	0.4, 1.21, 0.044
		57	Stud 2	0.6, 1.21, 0.044		83	Stud 2	0.8, 1.21, 0.044
		58	Window Assembly	1.7, 1.21, 0.044		84	Stud 3	1.2, 1.21, 0.044
		59	Stud 3	2.1, 1.21, 0.044		85	Window Assembly	5.6, 1.21, 0.044

		60	Stud 4	2.5, 1.21, 0.044		86	Stud 4	6, 1.21, 0.044
		61	Stud 5	2.9, 1.21, 0.044		87	Stud 5	6.4, 1.21, 0.044
		62	Stud 6	3.3, 1.21, 0.044		88	Stud 7	6.8, 1.21, 0.044
		63	Stud 7	3.7, 1.21, 0.044		89	Stud 8	7.2, 1.21, 0.044
		64	Stud 8	4.1, 1.21, 0.044		90	Top Track	3.04, 3.65, 0.044
		65	Stud 9	4.5, 1.21, 0.044		91	OSB Sheet 1	0.6, 1.21, 0.025
		66	Stud 10	4.9, 1.21, 0.044		92	OSB Sheet 2	2.8, 2.286, 0.025
		67	Window Assembly	6, 1.21, 0.044		93	OSB Sheet 3	5, 2.29, 0.025
		68	Stud 11	6.4, 1.21, 0.044		94	OSB Sheet 4	2.8, 0.762, 0.025
		69	Stud 12	6.8, 1.21, 0.044		95	OSB Sheet 5	5, 0.76, 0.025
		70	Stud 13	7.2, 1.21, 0.044		96	OSB Sheet 6	6.2, 1.21, 0.025
		71	Top Track	3.04, 3.65, 0.044		97	OSB Sheet 7	7.2, 1.21, 0.025
		72	OSB Sheet 1	0.4, 1.21, 0.025				
		73	OSB Sheet 2	1.92, 1.97, 0.025				
		74	OSB Sheet 3	1.92, 0.76, 0.025				
		75	OSB Sheet 4	3.12, 1.21, 0.025				
		76	OSB Sheet 5	4.32, 1.21, 0.025				
		77	OSB Sheet 6	5.52, 1.21, 0.025				
		78	OSB Sheet 7	6.72, 1.97, 0.025				
		79	OSB Sheet 8	6.72, 0.76, 0.025				
		80	OSB Sheet 9	7.2, 1.21, 0.025				

WALL "Panel1": 2x6 @ 16" OC

Panel1 - Track Schedule			
Label	Type	Length (mm)	Count
T0	2x6	5791	2
T1	2x6	914	2
T2	2x6	991	2

Panel1 - Stud Schedule			
Label	Type	Length (mm)	Count
C0	2x6	379	4
C1	2x6	688	4
C2	2x6	398	4
C3	2x6	669	4
J0	2x6	1637	2
J1	2x6	1656	2
S0	2x6	2362	14

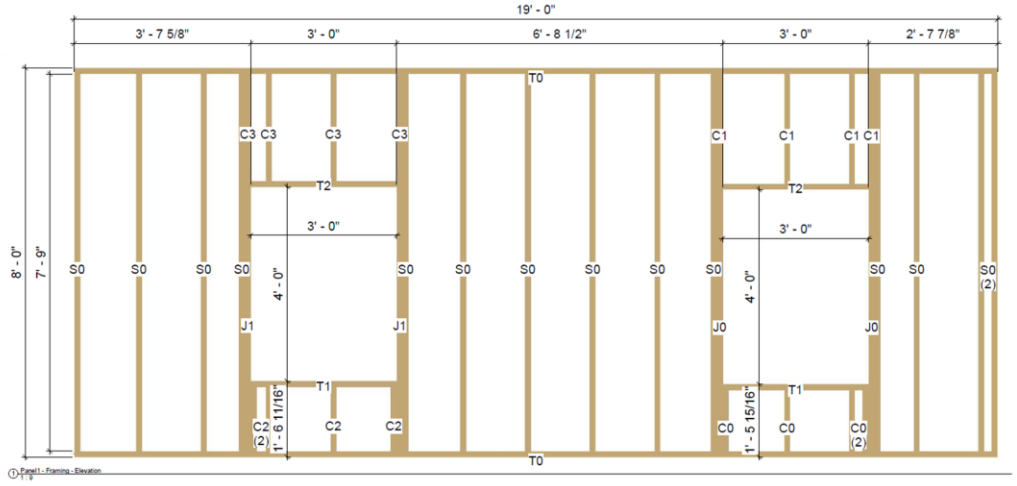


Figure 68. Base Model Panel 1 Framing

Panel1 - W0 - Track Schedule			
Label	Type	Length	Count
T1	2x6	3' - 0"	1
T2	2x6	3' - 3"	1

Panel1 - W1 - Track Schedule			
Label	Type	Length	Count
T1	2x6	3' - 0"	1
T2	2x6	3' - 3"	1

Panel1 - W0 - Stud Schedule			
Label	Type	Length	Count
C0	2x6	1' - 2 15/16"	4
C1	2x6	2' - 3 1/16"	4
J0	2x6	5' - 4 7/16"	2
S0	2x6	7' - 9"	2

Panel1 - W1 - Stud Schedule			
Label	Type	Length	Count
C2	2x6	1' - 3 11/16"	4
C3	2x6	2' - 2 5/16"	4
J1	2x6	5' - 5 3/16"	2
S0	2x6	7' - 9"	2

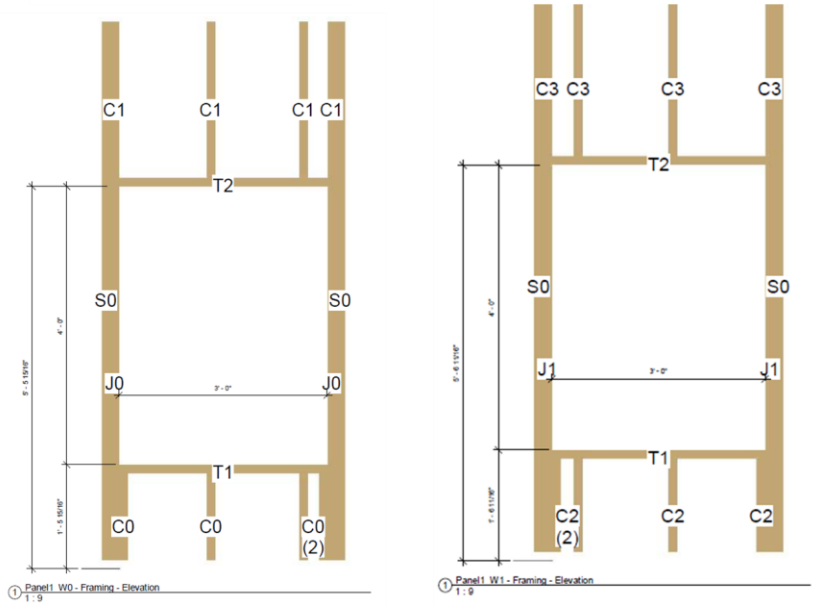


Figure 69. Base Model Panel 1 Window Framing

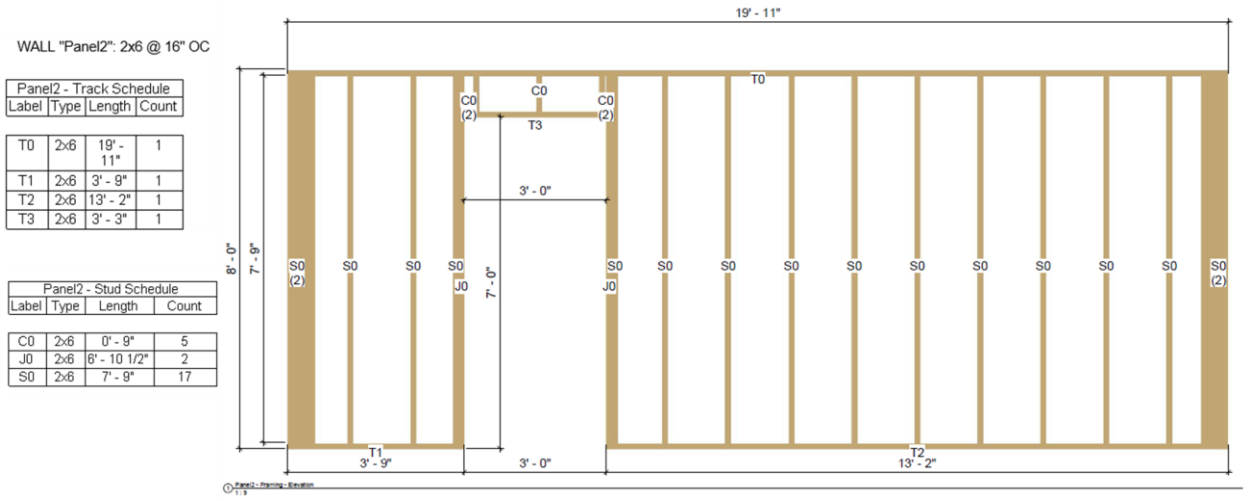


Figure 70. Base Model Panel 2 Framing

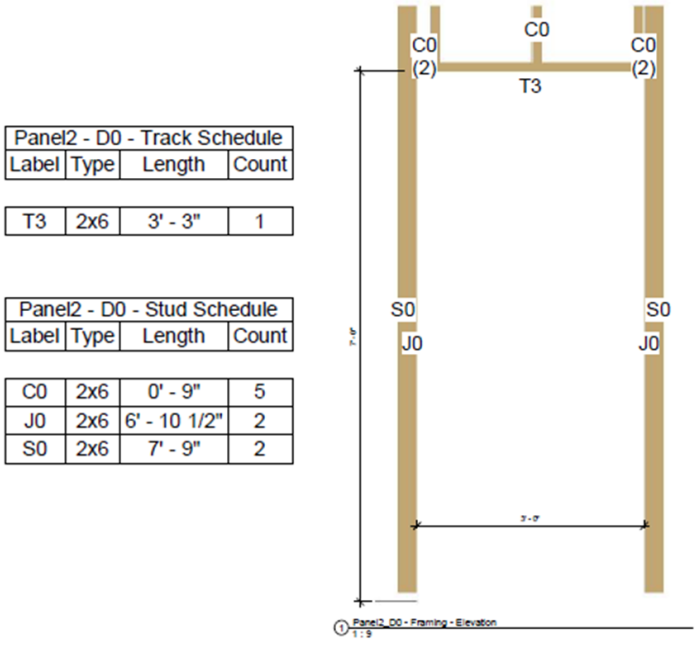


Figure 71. Base Model Panel 2 Door Framing

Panel3 - Track Schedule			
Label	Type	Length	Count
T0	2x6	19' - 0"	2
T1	2x6	3' - 0"	1
T2	2x6	3' - 3"	1

Panel3 - Stud Schedule			
Label	Type	Length	Count
C0	2x6	1' - 11 1/4"	4
C1	2x6	1' - 6 3/4"	4
J0	2x6	6' - 0 3/4"	2
S0	2x6	7' - 9"	16

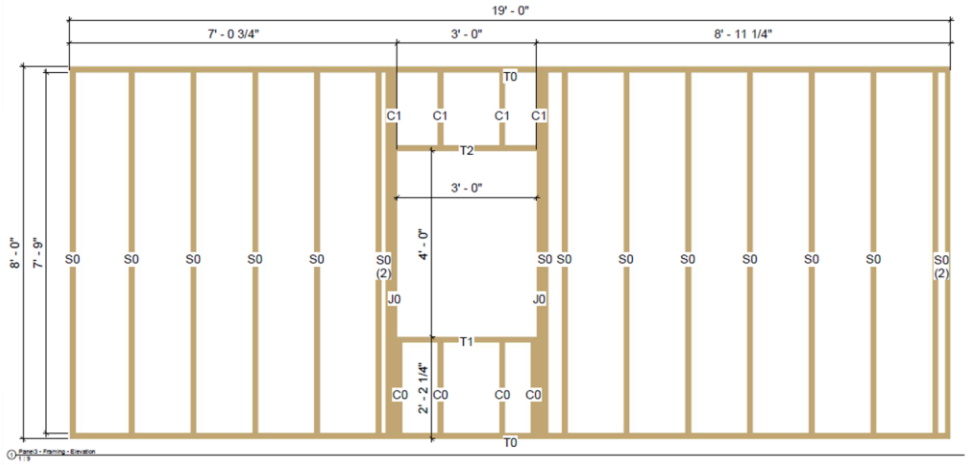


Figure 72. Base Model Panel 3 Framing

Panel3 - W0 - Track Schedule			
Label	Type	Length	Count
T1	2x6	3' - 0"	1
T2	2x6	3' - 3"	1

Panel3 - W0 - Stud Schedule			
Label	Type	Length	Count
C0	2x6	1' - 11 1/4"	4
C1	2x6	1' - 6 3/4"	4
J0	2x6	6' - 0 3/4"	2
S0	2x6	7' - 9"	2

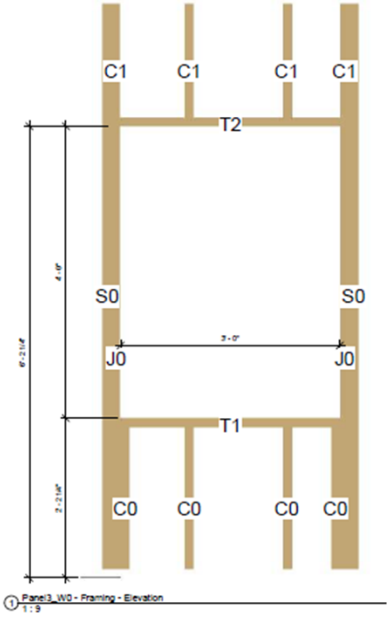
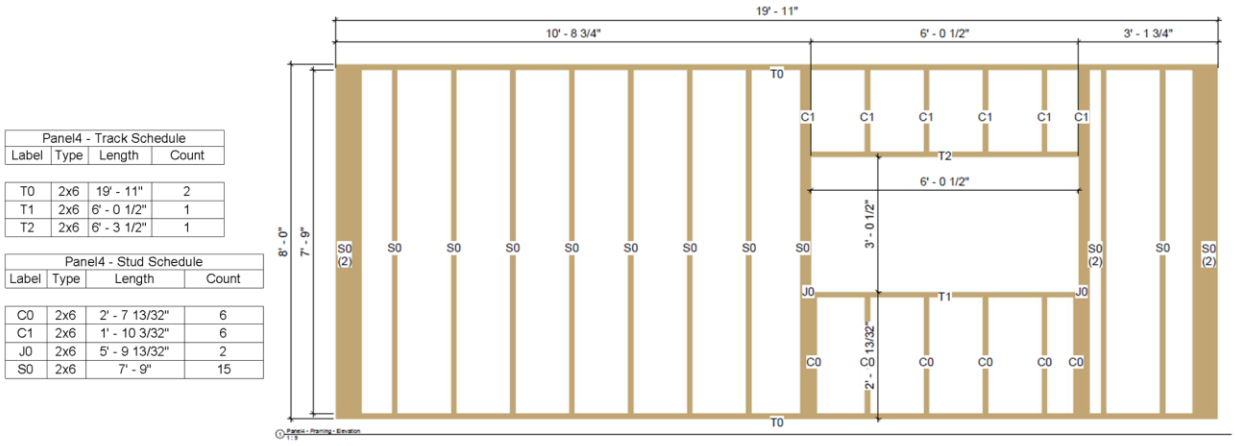


Figure 73. Base Model Panel 3 Window Framing



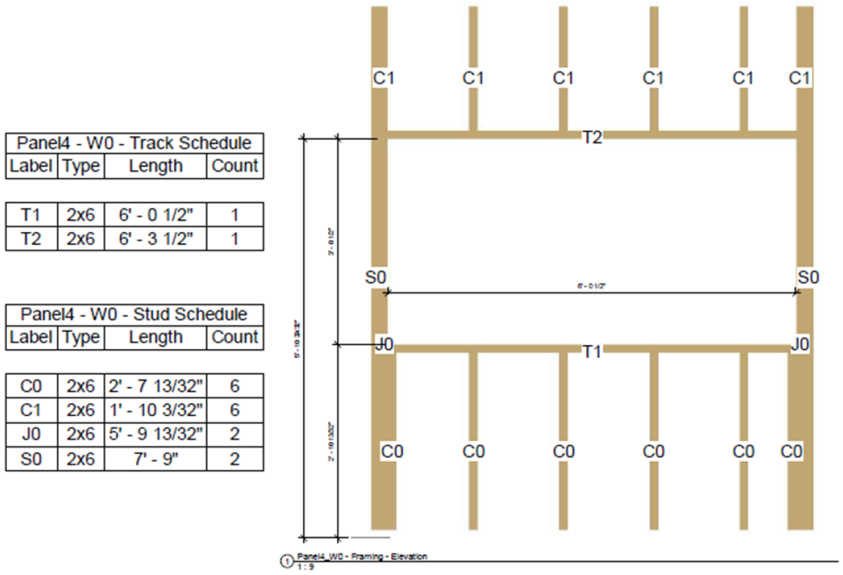
Panel4 - Track Schedule

Label	Type	Length	Count
T0	2x6	19' - 11"	2
T1	2x6	6' - 0 1/2"	1
T2	2x6	6' - 3 1/2"	1

Panel4 - Stud Schedule

Label	Type	Length	Count
C0	2x6	2' - 7 13/32"	6
C1	2x6	1' - 10 3/32"	6
J0	2x6	5' - 9 13/32"	2
S0	2x6	7' - 9"	15

Figure 74. Base Model Panel 4 Framing



Panel4 - W0 - Track Schedule

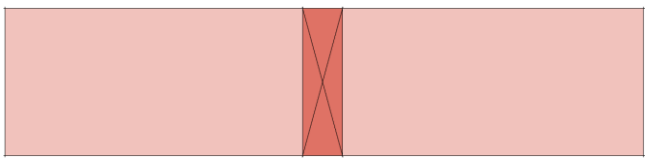
Label	Type	Length	Count
T1	2x6	6' - 0 1/2"	1
T2	2x6	6' - 3 1/2"	1

Panel4 - W0 - Stud Schedule

Label	Type	Length	Count
C0	2x6	2' - 7 13/32"	6
C1	2x6	1' - 10 3/32"	6
J0	2x6	5' - 9 13/32"	2
S0	2x6	7' - 9"	2

Figure 75. Base Model Panel 4 Window Framing

Table 11. Assembly Rules: Case of Vertical Studs

Plate Size	**Configuration Name**	**Visual Representation**
2 x 6	2 x 6 Vertical Stud	

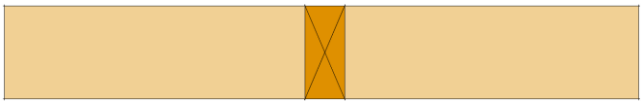
2 x 6	2 x 4 Staggered Studs	
2 x 4	2 x 4 Vertical Stud	

Table 12. Assembly Rules: Case of Horizontal Studs

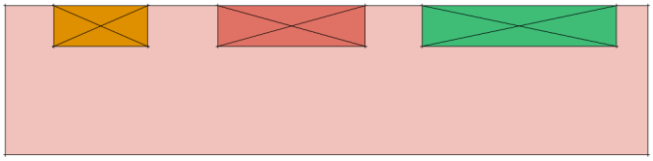
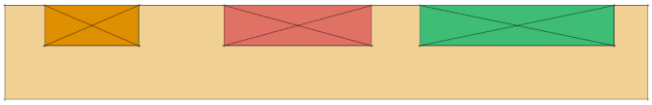
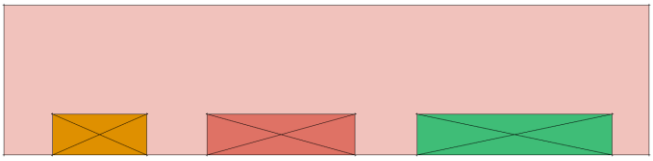
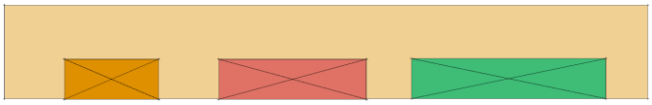
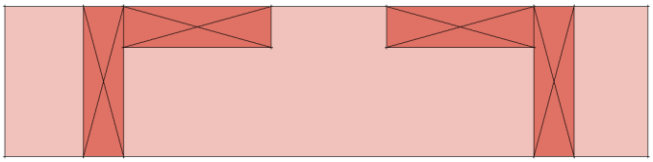
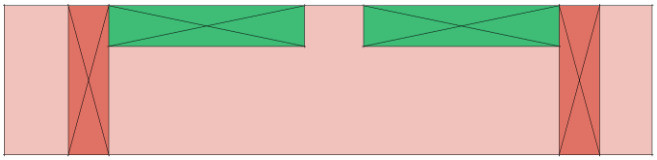
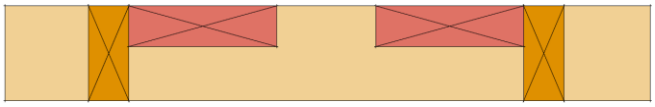
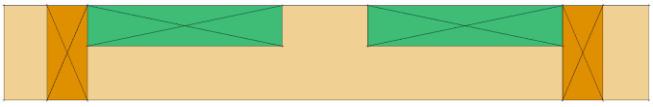
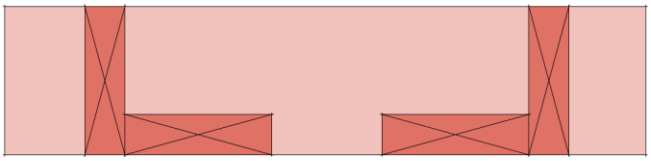
Plate Size	**Configuration Name**	**Visual Representation**
Top		
2 x 6	2 x 4, 2 x 6, and 2x8 Horizontal Stud	
2 x 4	2 x 4, 2 x 6, and 2x8 Horizontal Stud	
Bottom		
2 x 6	2 x 4, 2 x 6, and 2x8 Horizontal Stud	
2 x 4	2 x 4, 2 x 6, and 2x8 Horizontal Stud	

Table 13. Assembly Rules: Case of “L” Components

Plate Size	**Configuration Name**	**Visual Representation**

Top		
2 x 6	2 x 6 Vertical Stud 2 x 4 Horizontal Stud	
2 x 6	2 x 6 Vertical Stud 2 x 6 Horizontal Stud	
2 x 6	2 x 6 Vertical Stud 2 x 8 Horizontal Stud	
2 x 4	2 x 4 Vertical Stud 2 x 4 Horizontal Stud	
2 x 4	2 x 4 Vertical Stud 2 x 6 Horizontal Stud	
2 x 4	2 x 4 Vertical Stud 2 x 8 Horizontal Stud	
Bottom		
2 x 6	2 x 6 Vertical Stud 2 x 6 Horizontal Stud	

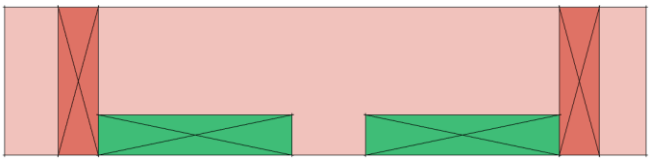
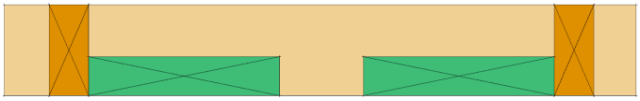
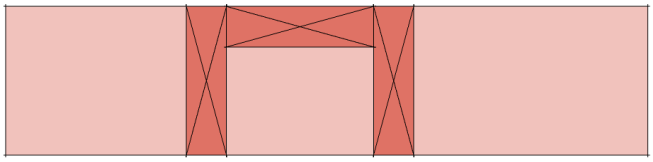
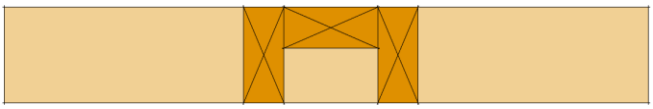
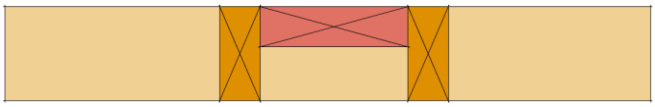
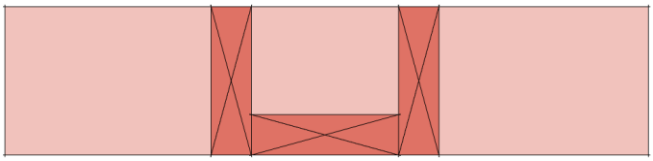
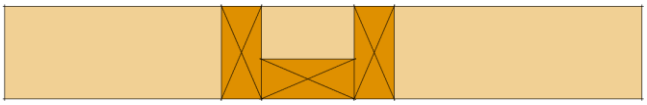
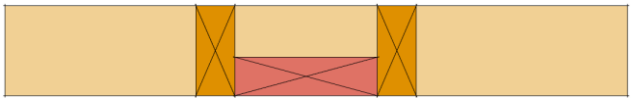
2 x 6	2 x 6 Vertical Stud 2 x 8 Horizontal Stud	
2 x 4	2 x 4 Vertical Stud 2 x 4 Horizontal Stud	
2 x 4	2 x 4 Vertical Stud 2 x 6 Horizontal Stud	
2 x 4	2 x 4 Vertical Stud 2 x 8 Horizontal Stud	

Table 14. Assembly Rules: Case of “U” Components

Plate Size	**Configuration Name**	**Visual Representation**
Top		
2 x 6	2 x 6 Vertical Studs 2 x 6 Horizontal Stud	
2 x 4	2 x 4 Vertical Studs 2 x 4 Horizontal Stud	
2 x 4	2 x 4 Vertical Studs and 2 x 6 Horizontal Stud	

Bottom		
2 x 6	2 x 6 Vertical Studs and 2 x 6 Horizontal Stud	
2 x 4	2 x 4 Vertical Studs and 2 x 4 Horizontal Stud	
2 x 4	2 x 4 Vertical Studs and 2 x 6 Horizontal Stud	

4.6.4 BIM Design Data Input

For the test cases derived from the BIM design models, the extracted and derived BIM data were utilized as inputs in subsequent phases of the experiment. This extraction process involved deconstructing the BIM structures into individual components, such as wall frames. Within each wall frame, studs (represented as IFC columns) and tracks (represented as IFC beams) were identified and grouped according to their corresponding component, like walls. Additionally, an automated process defined the assembly sequence for each element within a wall frame, based on their spatial relationships. This information, extracted and deduced from the BIM design models, was then fed into the simulation for the assembly of wall frames, employing a rule-based method implemented in B-Prolog for data analysis and derivation.

4.6.5 Robotic System Model Selection

Key components of the robotic system used in the simulation included:

- **Robots**: Four robots were selected for their reach and payload capacities: Robots A and C are KUKA KR IONTEC 2500/50 (with a reach of 2.50 m and payload of 50 kg) and Robots B and D are KUKA KR QUANTEC PA 3200/120 (with a reach of 3.20 m and payload of 120 kg), as shown in Figure 76 (a and b).
- **Linear Movement Units**: Two floor-mounted ABB linear units KL 4000 with an overall length of 14 meters, as shown in Figure 77. The linear unit functions as a single-axis track system, primarily serving as an external axis for the robot. The robot is mounted on a carriage that enables movement along this linear axis. This carriage is propelled by its own drive and is controlled by the robot's controller. The range of the carriage's movement is restricted by programmable software limit switches, and additional protection is provided by mechanical stops (buffers) positioned on the rack side.
- **End-Effectors**: The simulation utilized two parallel grippers for stable grasping of construction materials (studs/tracks) during framing operations, as shown in Figure 78(a). Figure 78(b) presents a variety of fixturing tools, including a nailing gun, circular saw, router, and screw gun model employed for fixing and processing wood elements in different operations. Additionally, Figure 78(c) shows a custom vacuum rig equipped with suction cups to lift and move OSB sheets.
- **Material Handling Devices**: Robots can pick the required materials (studs/tracks/sheets) from the material bays where they are stored. Additionally, two chain conveyors were designed and installed on the assembly table to transport the finished prefabricated panel to the next station for other operations, as shown in Figure 79.
- **Fixture**: The factory floor serves as the fixture for the assembly and other processes.

This comprehensive setup ensures a realistic and effective simulation of the robotic assembly process, leveraging the detailed BIM data and advanced robotic components for accurate replication and analysis of real-world construction scenarios.

4.6.6 Simulation Environment Setup

In the simulation environment, detailed models representing construction materials, specifically 2” × 6” elements and 4” x 8” OSB sheets, were developed. These models initially had a uniform length of 20 ft long tracks and 8 ft long studs, a dimension set to be adjusted later in accordance with the specific requirements of the BIM design models during the mapping process.

Furthermore, each component of the robotic system was designed and/or imported into the simulation environment. This step involved the aggregation of studs/sheets and various robotic system components into cohesive assemblies. In the ROS simulation platform, the “.urdf” file format is utilized to encapsulate an assembly. The Unified Robotic Description Format (URDF) is an XML file format used in ROS to describe all elements of a robotic cell. This format effectively stores all pertinent information related to a 3D asset, ensuring that each component is accurately represented within the simulation.

The final step involved adding the 3D models to the simulation and strategically organizing and positioning each component of the robotic system within the simulated world. The arrangement and spatial configuration of these components in the simulation environment are presented in Figure 80. This organized setup in the simulation not only reflects the physical arrangement but also facilitates efficient interaction and coordination among different elements of the robotic system during the simulated assembly process.

(a) Robots A and C **(b) Robots B and D**

Figure 76. Robot Models: (a) KUKA KR IONTEC 2500/50, (b) KUKA KR QUANTEC PA 3200/120

Figure 77. ABB Linear Unit KL 4000

(a) Grippers Assembly **(b) Multi-Tools Rack** **(c) Vacuum Lifter Rig**

Figure 78. Robot End-Effectors

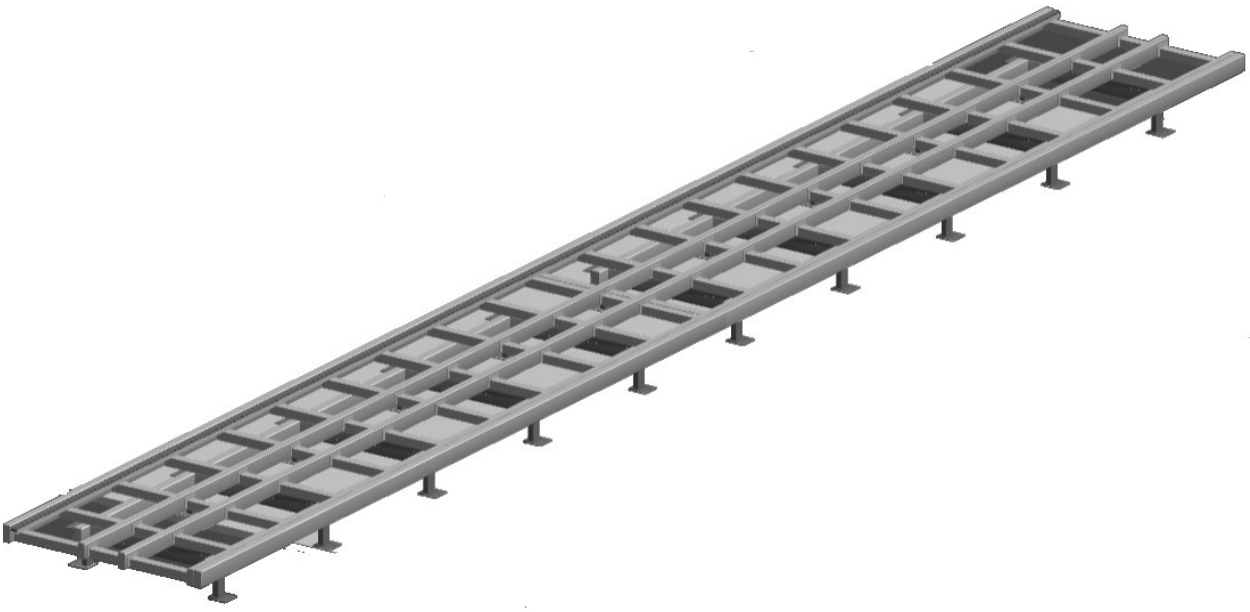


Figure 79. Assembly Table with Conveyors

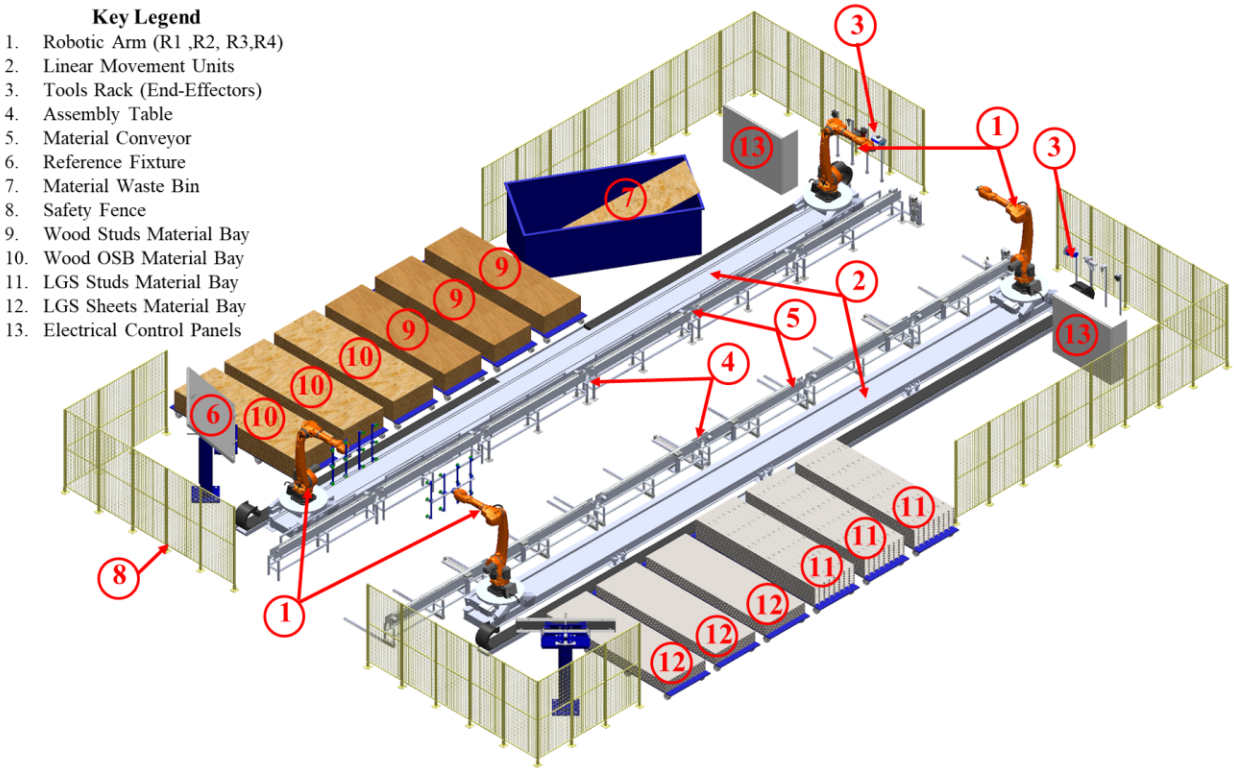


Figure 80. Spatial Layout of the Simulation Components

4.6.7 Simulation Generation and Execution

In the ROS framework, processes are managed as nodes within a graph structure, connected through topics, enabling message exchanges, service calls, and data access from a parameter server. The ROS Master coordinates these interactions, facilitating direct peer-to-peer communications among nodes, which is beneficial for robotic systems involving interconnected hardware and external computers.

ROS structures these nodes and topics in world files using VRML97 standards and ROS-specific nodes, all encoded in B-Prolog. These world files contain all model assets and configurations for the robotic system.

Upon loading the ROS world file, the simulation automatically commenced. Within this simulated environment, the robotic system executed the assembly process, guided by the framing and nailing subroutines and informed by the input from the BIM data. Figure 81 shows a snapshot of the robot engaged in the framing operation including pick and place, clamping, and nailing. While Figure 82 shows a snapshot of the robot engaged in the sheathing operation including pick and place, holding, nailing, routing, and cutting. Additionally, Figure 83 shows the picking mechanism of the different stud's configuration: vertical or horizontal. This setup allowed for a comprehensive and realistic simulation of the robotic assembly process, providing valuable insights into the practical application of robotic systems in construction environments.

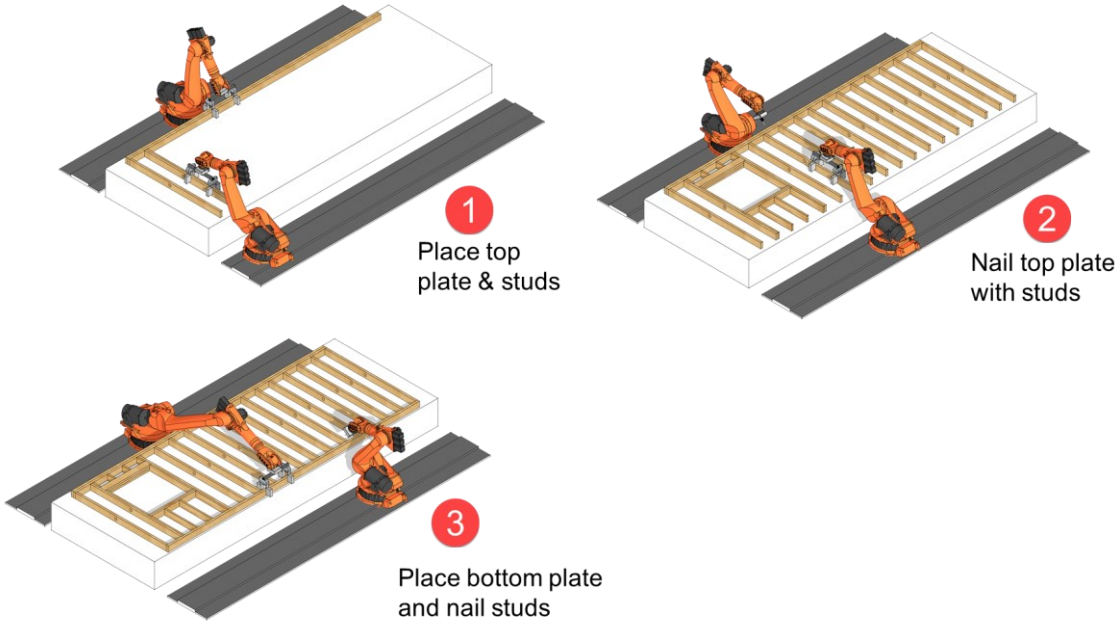


Figure 81. Framing and Nailing Operations

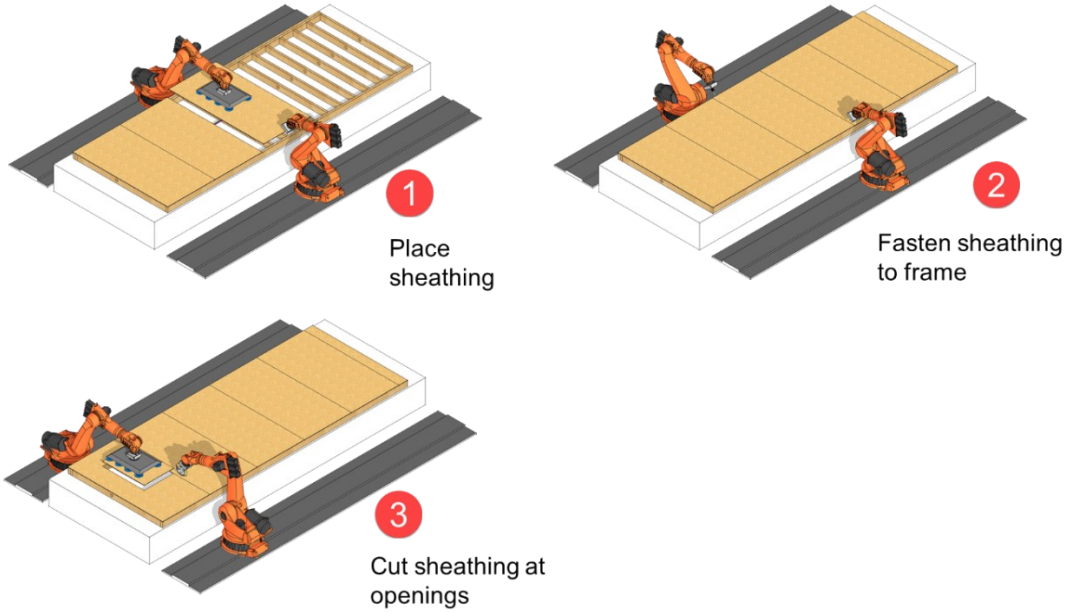


Figure 82. Sheathing and Nailing/Cutting Operations

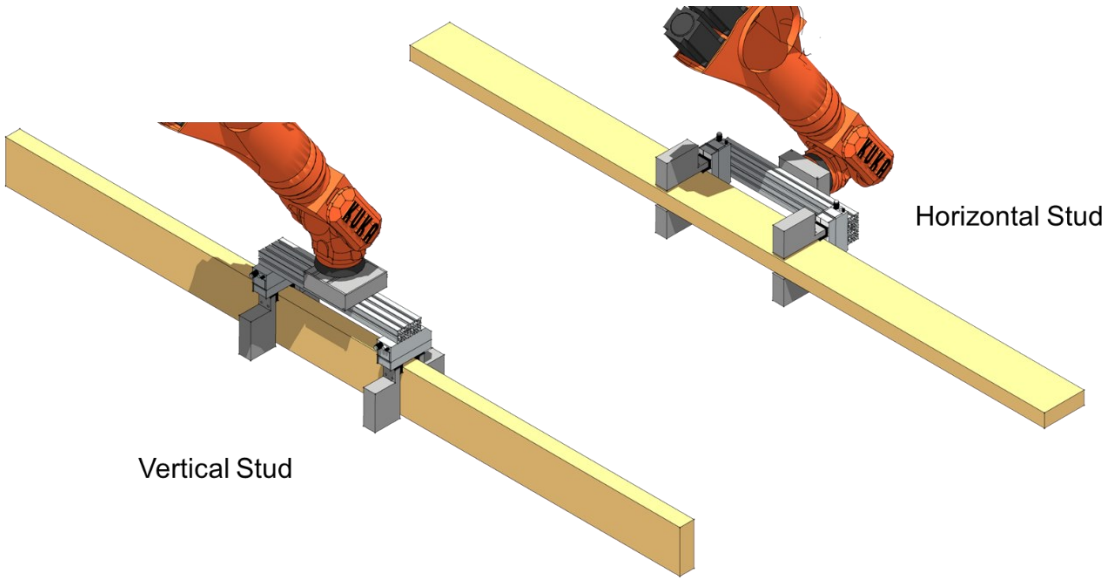


Figure 83. Pick and Place Mechanism

4.6.8 BIM-Simulator Integration

The successful integration of BIM with the ROS robotic simulator was demonstrated in the simulation of the assembly process for each test model, as shown in Figure 84 and Figure 85. The outcomes of the BIM-ROS integration revealed a high level of consistency and equivalence between the elements mapped in the simulation and those in the design model. This not only validated the effectiveness of the mapping process but also underscored the efficiency and accuracy of the automated generation process for building elements in the simulation. By automating this process, the methodology eliminated the need for manual mapping and creation of each building element object, which would typically be a time-consuming, complex, and error-prone task.

In an OSC setting, the integration of BIM with robotic simulations becomes instrumental for operational efficiency and precision. Specifically, the adoption of multi-panel concept in robotic fabrication is crucial for streamlining the assembly of multiple building sections before onsite installation. By optimizing and merging panels together as indicated in Figure 86, the user could

specify the fabricated panels unit, maximum length, minimum length, gap between panels, and exclude panels that are less than specific length for optimization purposes.

In the developed user interface, robotic operations for each multi-panel, including cutting, drilling, and nailing, are displayed in the center of gravity (COG) table at the bottom. For calculating the COG, by default, only the structural layer is considered based on the layer's material assigned density value. Additionally, it accounts for the weight of windows and doors in the COG calculation. Also, the use can specify if holes should be placed on the top, bottom, or both ends of each panel, and determine the number of holes per panel. In Figure 86 an outline of the panel displays a blue marker indicating the calculated COG and markers for each drill hole location. If a drill hole conflicts with any structural elements like studs, it will be marked in red. After confirming that all robotic operations are accurate and there are no conflicts, then the 'Export Excel file' button to save the output as shown in Figure 87 and Figure 88.

A developed recipe file generator for BIM-robotic applications automates the creation of instruction sets from BIM data, enabling robots to execute complex tasks reliably. The sample recipe file output demonstrates how these instructions guide robotic actions effectively as illustrated in Figure 89. Moreover, the seamless integration of the BIM scene with the robotic simulator scene ensures that the digital model and the robotic actions are perfectly synchronized, enhancing the predictability and scalability of offsite construction processes as demonstrated in Figure 90.

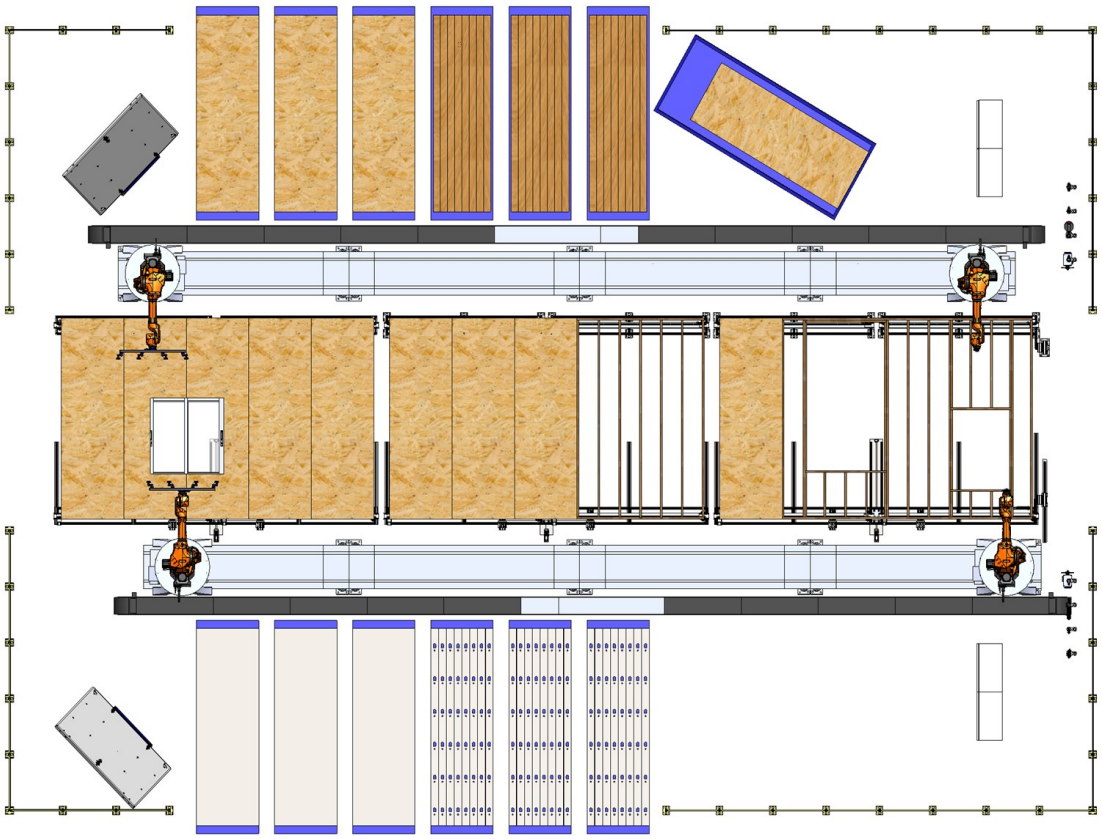


Figure 84. Simulation Result for Robotic Cell with Wood Framing and Sheathing

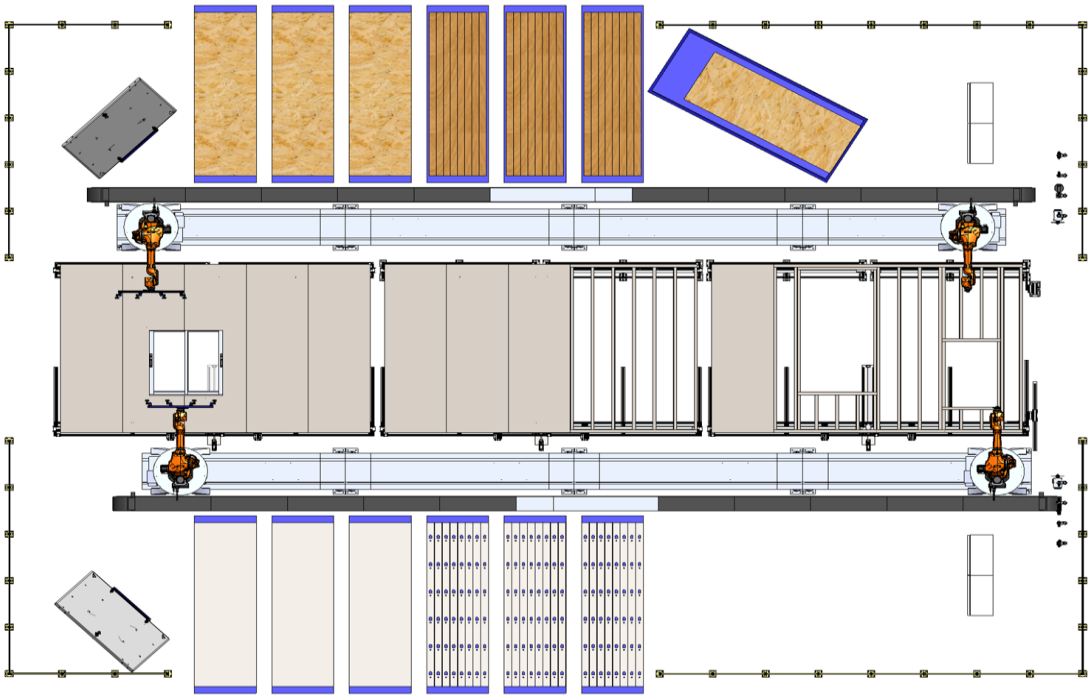


Figure 85. Simulation Result for Robotic Cell with LGS Framing and Sheathing

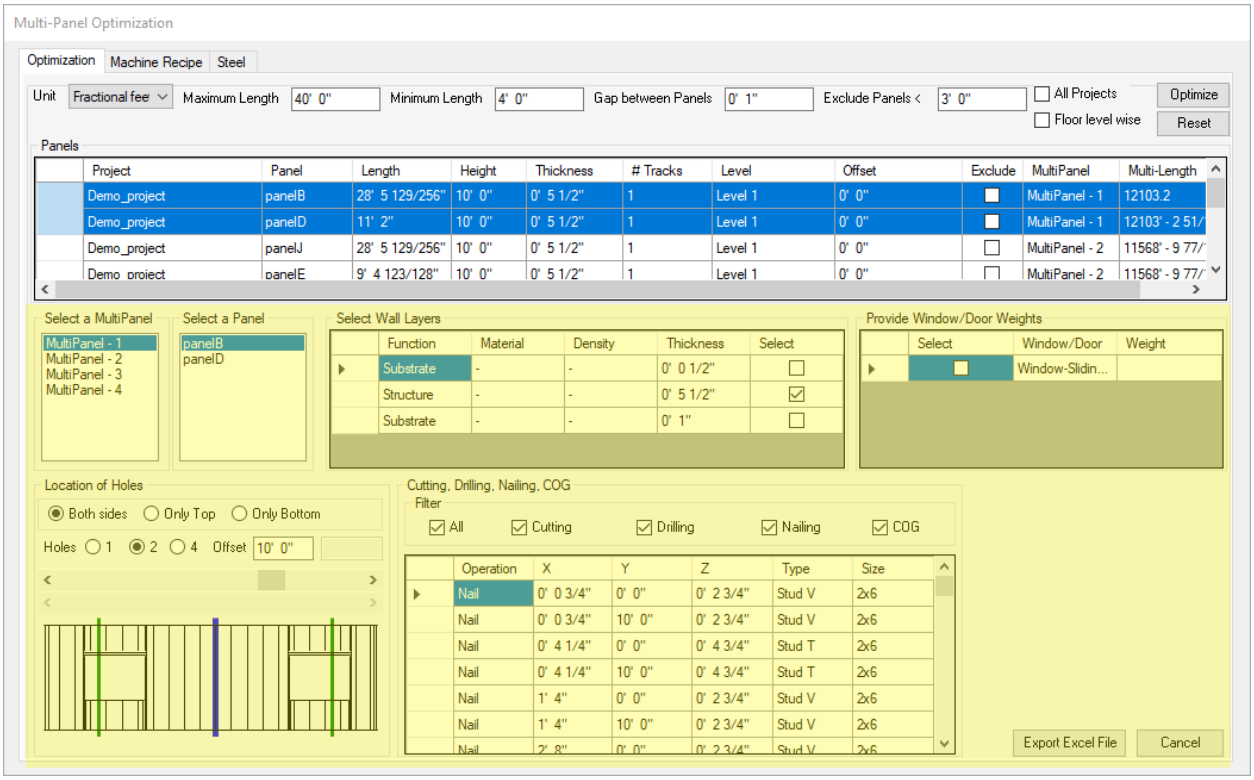


Figure 86. Multi-Panels Optimizer Generator

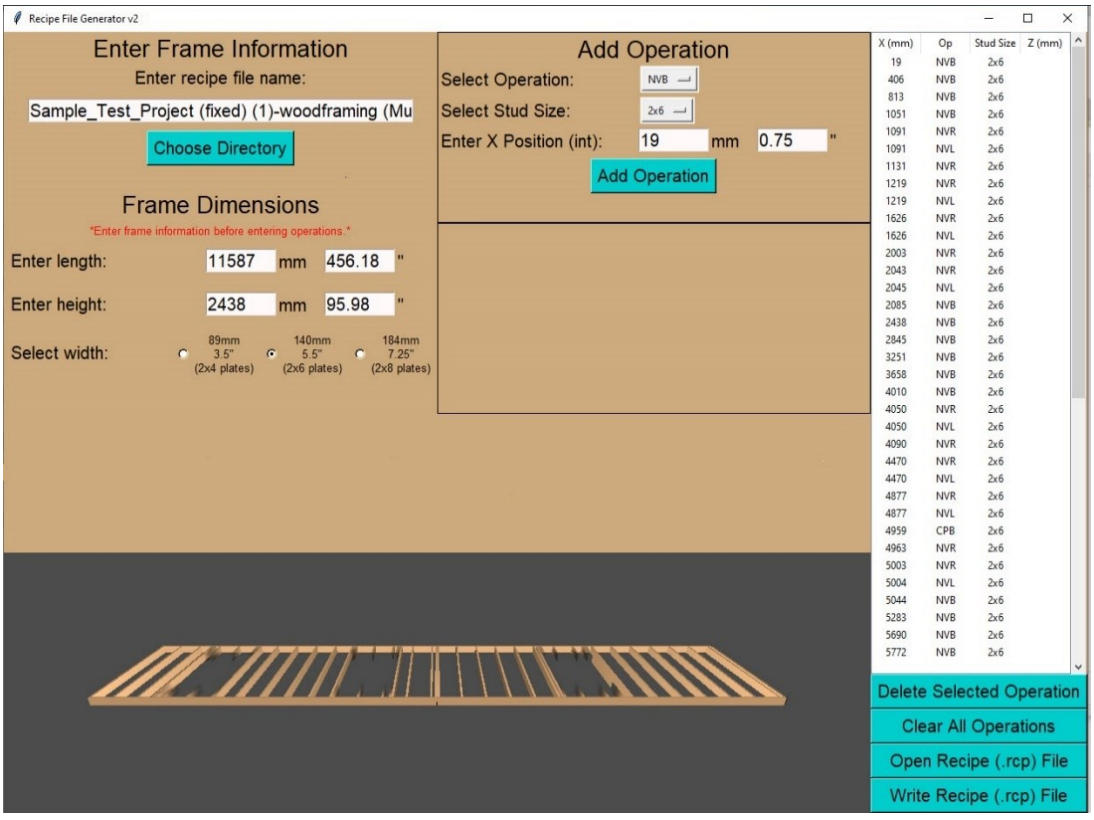


Figure 87. Simulated Multi-Panel Robotic Operations for Panels 1 and 3 of the Base Model

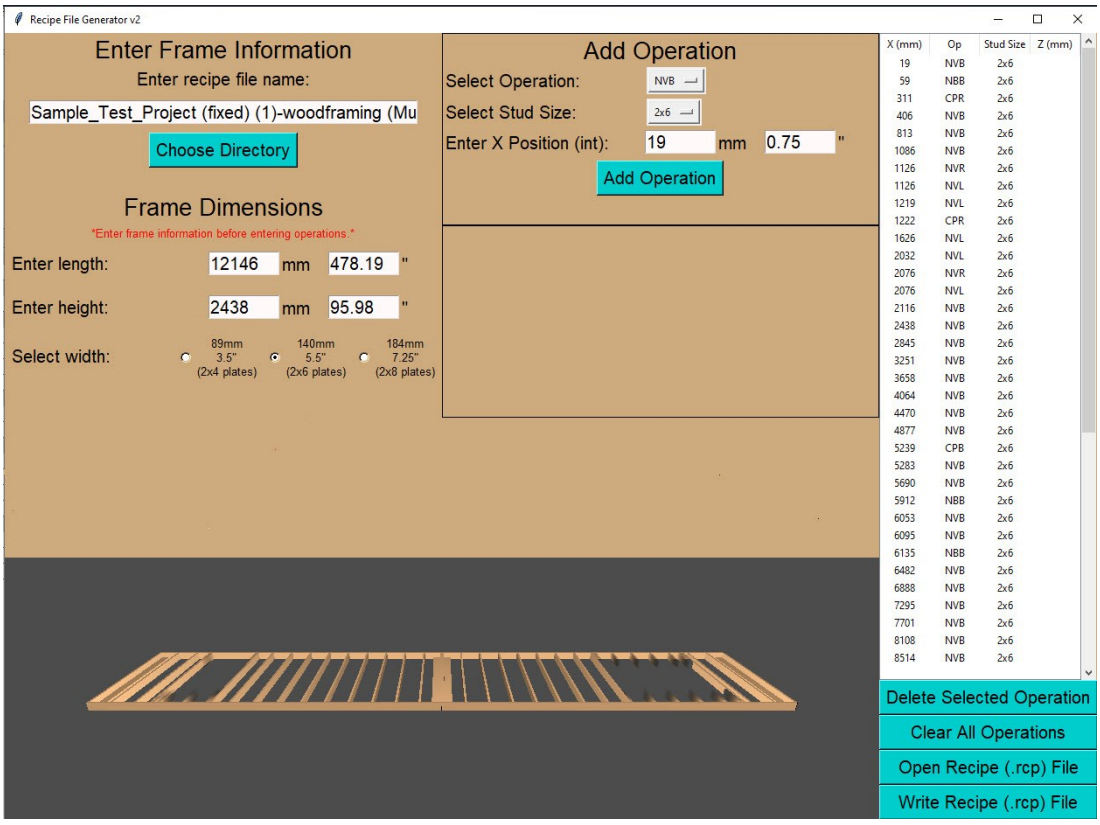
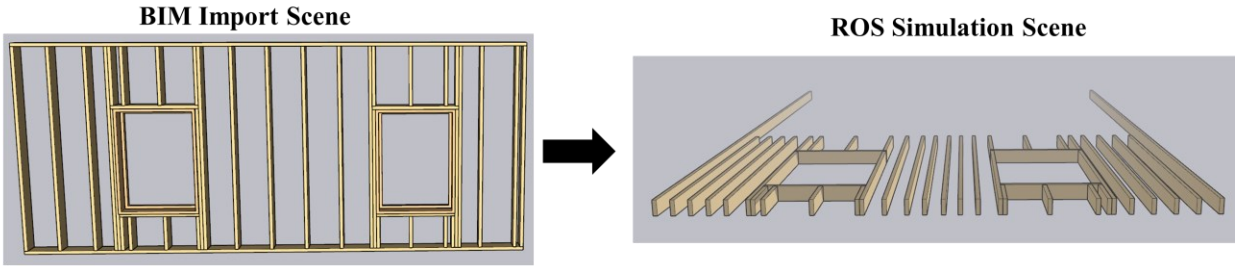


Figure 88. Simulated Multi-Panel Robotic Operations for Panels 2 and 4 of the Base Model

Frame_Length,12146Frame_Height,2438Frame_Width,140Num_Of_Operations,57Xpositions[0],19Xpositions[1],59Xpositions[2],311Xpositions[3],406Xpositions[4],813Xpositions[5],1086Xpositions[6],1126Xpositions[7],1126Xpositions[8],1219Xpositions[9],1222Xpositions[10],1626Xpositions[11],2032Xpositions[12],2076Xpositions[13],2076Xpositions[14],2116Xpositions[15],2438Xpositions[16],2845Xpositions[17],3251Xpositions[18],3658Xpositions[19],4064Xpositions[20],4470Xpositions[21],4877Xpositions[22],5239Xpositions[23],5283Xpositions[24],5690Xpositions[25],5912Xpositions[26],6053Xpositions[27],6095Xpositions[28],6135Xpositions[29],6482Xpositions[30],6888Xpositions[31],7295Xpositions[32],7701Xpositions[33],8108Xpositions[34],8514Xpositions[35],8920Xpositions[36],9289Xpositions[37],9329Xpositions[38],9329Xpositions[39],9369Xpositions[40],9733Xpositions[41],9733Xpositions[42],10140Xpositions[43],10140Xpositions[44],10546Xpositions[45],10546Xpositions[46],10952Xpositions[47],10952Xpositions[48],11168Xpositions[49],11208Xpositions[50],11210Xpositions[51],11250Xpositions[52],11314Xpositions[53],11359Xpositions[54],11765Xpositions[55],11987Xpositions[56],12129
Operations[0],'NVB'Operations[1],'NBB'Operations[2],'CPR'Operations[3],'NVB'Operations[4],'NVB'Operations[5],'NVB'Operations[6],'NVR'Operations[7],'NVL'Operations[8],'NVL'Operations[9],'CPR'Operations[10],'NVL'Operations[11],'NVL'Operations[12],'NVR'Operations[13],'NVL'Operations[14],'NVB'Operations[15],'NVB'Operations[16],'NVB'Operations[17],'NVB'Operations[18],'NVB'Operations[19],'NVB'Operations[20],'NVB'Operations[21],'NVB'Operations[22],'CPB'Operations[23],'NVB'Operations[24],'NVB'Operations[25],'NBB'Operations[26],'NVB'Operations[27],'NVB'Operations[28],'NBB'Operations[29],'NVB'Operations[30],'NVB'Operations[31],'NVB'Operations[32],'NVB'Operations[33],'NVB'Operations[34],'NVB'Operations[35],'NVB'Operations[36],'NVB'Operations[37],'NVL'Operations[38],'NVR'Operations[39],'NVR'Operations[40],'NVR'Operations[41],'NVL'Operations[42],'NVR'Operations[43],'NVL'Operations[44],'NVR'Operations[45],'NVL'Operations[46],'NVR'Operations[47],'NVL'Operations[48],'NVR'Operations[49],'NVR'Operations[50],'NVL'Operations[51],'NVB'Operations[52],'CFB'Operations[53],'NVB'Operations[54],'NVB'Operations[55],'NBB'Operations[56],'NVB'StudSize[0],'2x6'StudSize[1],'2x6'StudSize[2],'2x6'StudSize[3],'2x6'StudSize[4],'2x6'StudSize[5],'2x6'StudSize[6],'2x6'StudSize[7],'2x6'StudSize[8],'2x6'StudSize[9],'2x6'StudSize[10],'2x6'StudSize[11],'2x6'StudSize[12],'2x6'StudSize[13],'2x6'StudSize[14],'2x6'StudSize[15],'2x6'StudSize[16],'2x6'StudSize[17],'2x6'StudSize[18],'2x6'StudSize[19],'2x6'StudSize[20],'2x6'StudSize[21],'2x6'StudSize[22],'2x6'StudSize[23],'2x6'StudSize[24],'2x6'StudSize[25],'2x6'StudSize[26],'2x6'StudSize[27],'2x6'StudSize[28],'2x6'StudSize[29],'2x6'StudSize[30],'2x6'StudSize[31],'2x6'StudSize[32],'2x6'StudSize[33],'2x6'StudSize[34],'2x6'StudSize[35],'2x6'StudSize[36],'2x6'StudSize[37],'2x6'StudSize[38],'2x6'StudSize[39],'2x6'StudSize[40],'2x6'StudSize[41],'2x6'StudSize[42],'2x6'StudSize[43],'2x6'StudSize[44],'2x6'StudSize[45],'2x6'StudSize[46],'2x6'StudSize[47],'2x6'StudSize[48],'2x6'StudSize[49],'2x6'StudSize[50],'2x6'StudSize[51],'2x6'StudSize[52],'2x6'StudSize[53],'2x6'StudSize[54],'2x6'StudSize[55],'2x6'StudSize[56],'2x6'K_Zpositions[0],0K_Zpositions[1],0K_Zpositions[2],0K_Zpositions[3],0K_Zpositions[4],0K_Zpositions[5],0K_Zpositions[6],0K_Zpositions[7],0K_Zpositions[8],0K_Zpositions[9],0K_Zpositions[10],0K_Zpositions[11],0K_Zpositions[12],0K_Zpositions[13],0K_Zpositions[14],0K_Zpositions[15],0K_Zpositions[16],0K_Zpositions[17],0K_Zpositions[18],0K_Zpositions[19],0K_Zpositions[20],0K_Zpositions[21],0K_Zpositions[22],0K_Zpositions[23],0K_Zpositions[24],0K_Zpositions[25],0K_Zpositions[26],0K_Zpositions[27],0K_Zpositions[28],0K_Zpositions[29],0K_Zpositions[30],0K_Zpositions[31],0K_Zpositions[32],0K_Zpositions[33],0K_Zpositions[34],0K_Zpositions[35],0K_Zpositions[36],0K_Zpositions[37],0K_Zpositions[38],0K_Zpositions[39],0K_Zpositions[40],0K_Zpositions[41],0K_Zpositions[42],0K_Zpositions[43],0K_Zpositions[44],0K_Zpositions[45],0K_Zpositions[46],0K_Zpositions[47],0K_Zpositions[48],0K_Zpositions[49],0

Figure 89. Sample Recipe File Output



BIM to ROS Communication/Data Transfer Protocol

Object=Element(TypeofController,NameofElement,LocalPlacement[3],MainAxis[3],Dimension[3])

```
element1=Element(supervisor,'beam1529',-53.3954,-23.958,0.0625,1.0,0.0,0.0,17.2962,0.125,0.291667)
element2=Element(supervisor,'beam1567',-33.2391,-23.958,0.0625,1.0,0.0,0.0,2.47917,0.125,0.291667)
element3=Element(supervisor,'column2341',-46.7235,-23.958,0.125,0,0,1.0,2.75,0.291667,0.125)
element4=Element(supervisor,'column2251',-50.7235,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element5=Element(supervisor,'column2273',-49.3901,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element6=Element(supervisor,'column2295',-48.0568,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element7=Element(supervisor,'column2363',-45.3901,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element8=Element(supervisor,'column2385',-44.0568,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element9=Element(supervisor,'column2407',-42.7235,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element10=Element(supervisor,'column2429',-41.3901,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element11=Element(supervisor,'column2451',-40.0568,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element12=Element(supervisor,'column2473',-38.7235,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element13=Element(supervisor,'column2495',-37.3901,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element14=Element(supervisor,'column2517',-36.161,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element15=Element(supervisor,'column2539',-33.1766,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element16=Element(supervisor,'column2561',-32.0568,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
element17=Element(supervisor,'column2583',-53.3329,-23.958,0.125,0,0,1.0,7.84375,0.291667,0.125)
```

Figure 90. Integration of BIM Scene with Robotic Simulator Scene

4.6.9 Robotic Assembly Process Evaluation

The developed framework for this study was applied to simulate the robotic assembly process of wood panels using the multi-wall assembly process, focusing on framing and sheathing tasks. The simulations, which involved four selected robots, successfully implemented the framing, sheathing, and nailing algorithms. As all panels were successfully assembled and fastened, the durations of the framing and sheathing processes for each wall panel were recorded and summarized, including the nailing process, as presented in Table 15. Two scenarios were compared. Two robots were utilized in Scenario A while four robots were utilized in Scenario B.

Table 15. Simulation Results of the Multi-Panel Automated Framing and Sheathing Processes

Framing Operations			
Scenarios		**Framing Status**	**Framing Time (min)**

	Panel ID	Pick and Place	Nailing	Pick and Place	Nailing
A Two Robots	1+3	Completed	Completed	4.53	2.53
	2+4	Completed	Completed	3.95	1.98
B Four Robots	1+3	Completed	Completed	2.22	1.24
	2+4	Completed	Completed	1.92	0.96
Sheathing Operations					
Scenarios	**Panel ID**	**Sheathing Status**		**Sheathing Time (min)**	
		Pick and Place	**Nailing**	**Pick and Place**	**Nailing**
A Two Robots	1+3	Completed	Completed	2.94	4.50
	2+4	Completed	Completed	3.22	5.47
B Four Robots	1+3	Completed	Completed	1.45	2.20
	2+4	Completed	Completed	1.46	2.68

This research distinguishes itself from earlier studies such as those by Kim et al. (2021) and Meschini et al. (2016) by focusing on industrial robots better suited for wood construction due to their higher payload capacities, whereas previous studies examined general-purpose robots for tasks like indoor painting and simulations of automated processes.

Robotic automation in construction doesn't have a one-size-fits-all solution; several viable options exist, influenced by factors like the size of manufacturing spaces and the cost of robotic components. Thus, different robotic configurations could have been used in this study.

The impact of this methodology is significant in the AEC domain, enhancing the evaluation of construction robotics and Virtual Design and Construction (VDC). It assesses the productivity of robotic systems and their operational behavior in specific construction activities. Moreover, the control systems developed for simulation—integrating algorithms and BIM—can be transferred to physical robots for real-world applications, as demonstrated in studies by Wagner et al. (2020) and Yang and Kang (2021), highlighting the practical benefits of these robotic advancements.

The performance evaluation of the simulation followed the criteria established in the methodology's evaluation phase, involving a comparative verification of the mapping and generation processes from BIM to ROS for consistency and accuracy. The simulation scrutinized the assembly process, where the robotic system utilized custom construction algorithms. After executing the assembly subroutines, the frames were virtually inspected and compared against the original BIM models to ensure accuracy and constancy. The methodology's iterative nature allows for continuous improvement. Adjustments to the robotic system components led to repeated simulations to enhance performance and achieve desired outcomes. The simulation also included an assessment of potential collisions to confirm that the robotic system operated without interference or errors, validating the effectiveness and reliability of the methodology.

4.7 Results and Discussions

An in-depth comparative analysis of the performance between two implementations of the proposed method – one utilizing a learning curve and another with the initial set of the assembly rules – is shown in Figure 91, Figure 92, and Figure 93 respectively. The figures indicate an improvement in results precision, accuracy, and F1 across the tested models (M1-M6). In addition to its capabilities in information extraction and inference, the algorithm demonstrated proficiency in accurately classifying various components' properties and relationships. This includes

determining the functions of walls (whether internal or external) and slabs (as roof or floor), setting product properties, and identifying the relationships between walls and openings, among other factors.

Furthermore, while the output information from the algorithm is initially formatted as a general file type (.csv), which is suitable for applications such as quantity takeoff and fabrication processes (like cutting), there is flexibility to adapt this output to other file formats. This adaptability is crucial for meeting the specific needs of downstream applications. For instance, the file format can be converted to .recipe for CNC code generation or .py for programming robotic controllers, depending on the application requirements.

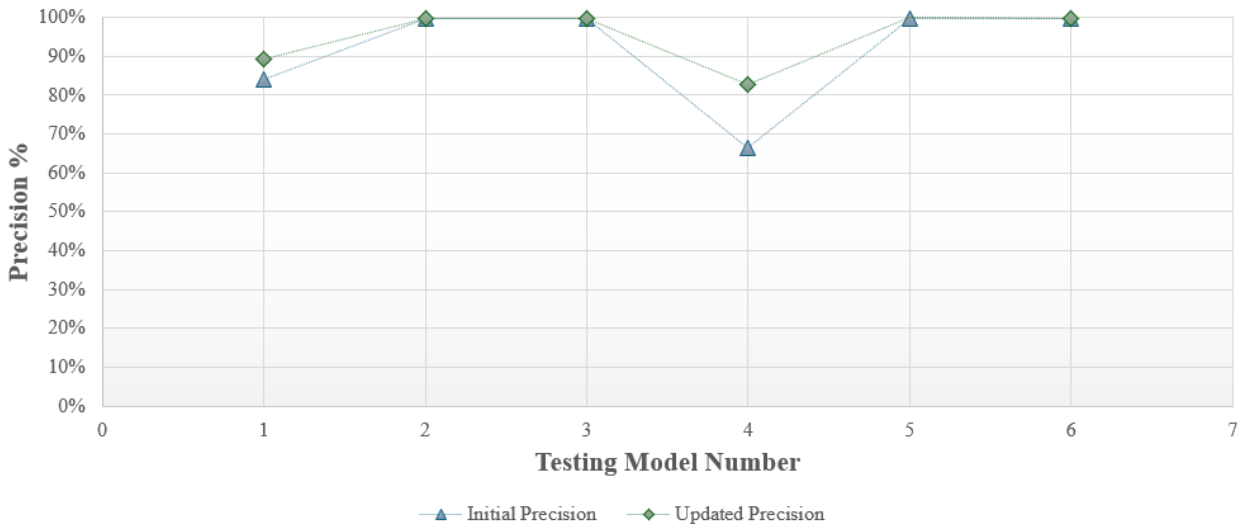


Figure 91. Learning Curve for Precision

Figure 92. Learning Curve for Accuracy

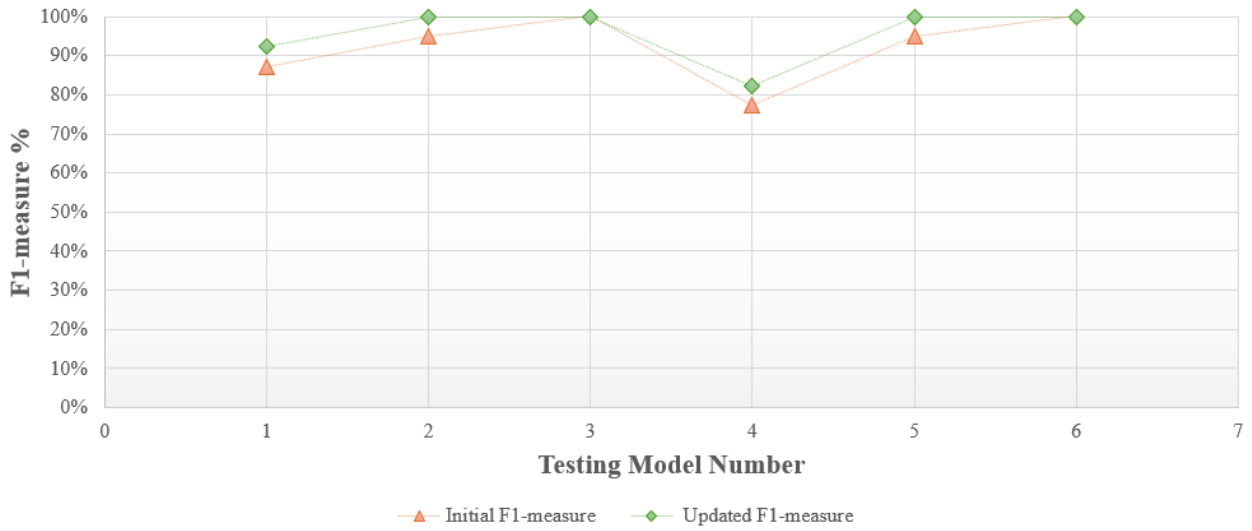


Figure 93. Learning Curve for F1 Measure

Figure 94 illustrates a comparative analysis of a model's performance on both training and testing datasets across five key performance metrics: Sensitivity, Specificity, Precision, Accuracy, and the F1 Measure. Specifically, the model demonstrates better performance in the testing phase for each metric, which is somewhat unusual as models generally tend to perform better during training due to overfitting. Sensitivity and Specificity are both at 94% for training but increase to 98% for testing, indicating a high true positive rate and the model's ability to correctly identify negatives. Precision rises from 92% during training to 95% in testing, suggesting that the model is reliable in

its positive predictions. Accuracy, a general measure of performance, shows a similar uplift from 94% to 98%, and the F1 Measure, which balances precision and sensitivity, improves from 92% to 96%. These results show that the model is robust and generalizes well to new data, with a remarkable consistency across different evaluative parameters in testing scenarios.

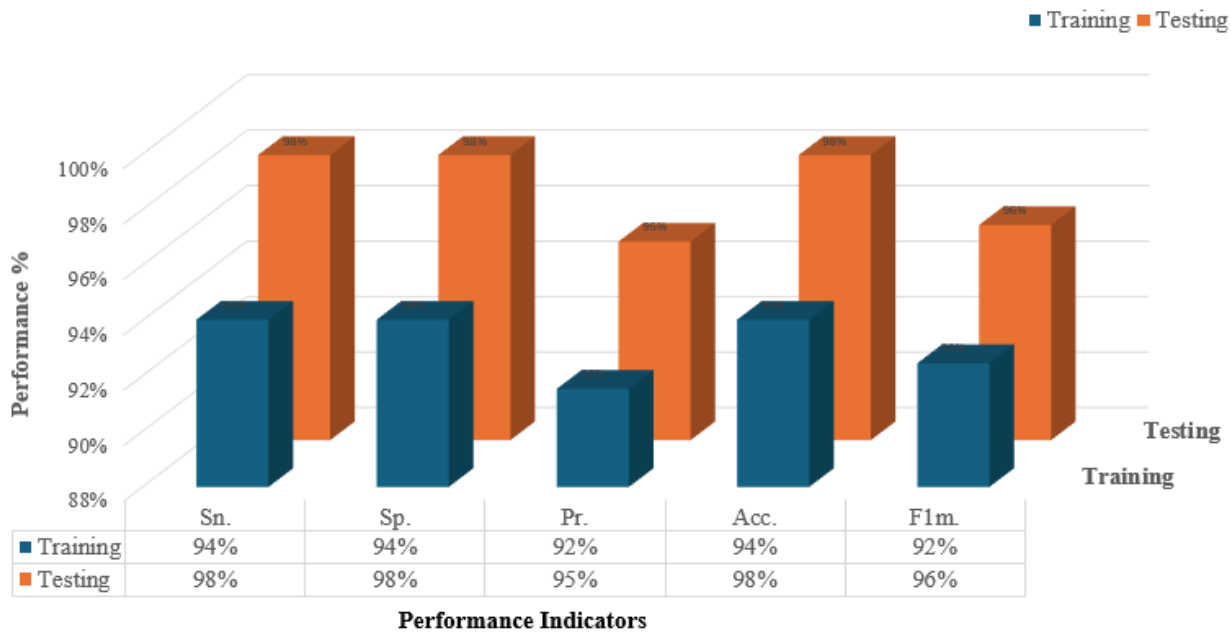


Figure 94. Graphical Representation of Performance Indicators for Testing Models

For a quantitative assessment of the performance in determining the assembly sequence, an analytical examination was conducted on the coordinates of each stud element within the organized sequence list. The analysis revealed that the element's coordinates were methodically ordered in ascending order. This sequential arrangement of coordinates endorses a systematic assembly logic that follows a pre-determined progression. This experiment aimed to evaluate the effectiveness of the proposed methodology, which utilizes BIM data as input for simulating robotic systems in the assembly of wood/LGS wall panels.

4.7.1 Nailing Target Locations Validation

To assess the accuracy of the nailing target locations defined in the methodology, a test comparing the computed values of these locations was conducted. The nailing target locations were initially determined using Equations (6), (7), and (8). These computed values were then verified by manually calculating the nailing location values for each element, utilizing data from the simulation. This manual calculation considered the position of each element's centroid and its dimensions relative to the robot's coordinate system after the framing process.

The results for test Panel 1 are detailed in Table 17. In this table, 'Location' refers to the connection points, as illustrated in Figure 95.

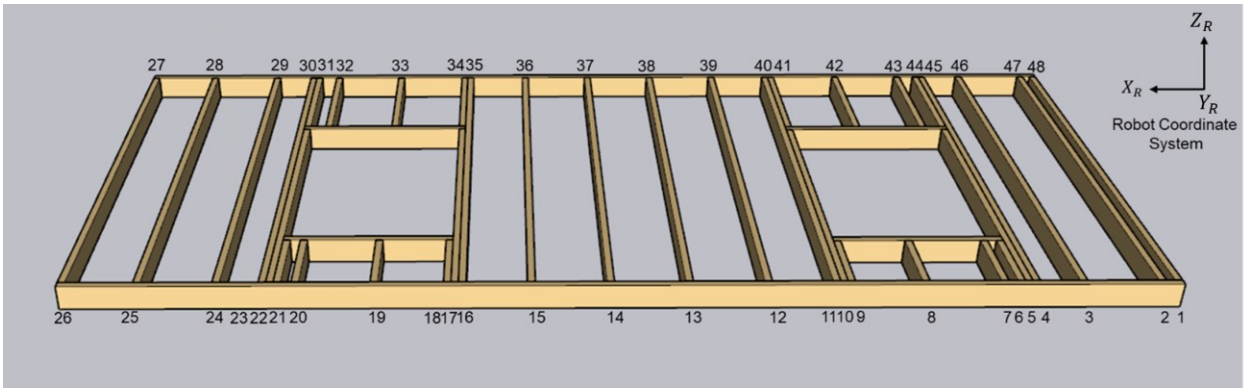


Figure 95. Locations of the Connection Points for the Nailing Operations for Panel 1 of the Base Model

The table outline N_x, N_y, and N_z as the nailing target locations determined using the automated proposed method, and X, Y, and Z as the manually traditional computed nailing locations. The differences between these sets of values are represented as X_{diff}, Y_{diff}, and Z_{diff}, indicating the relative change (in percentage) between the two approaches. This comparative analysis provides a quantitative assessment of the methodology's accuracy in determining nailing target locations, which is essential for ensuring precision in robotic nailing operations in wood frame construction.

4.7.2 Comparison to Manual Extraction Process

The efficiency of the proposed methodology was evaluated in terms of time, comparing it to manual operations. This assessment involved manually extracting necessary information for simulating the assembly process, adhering to a six-step procedure detailed in Table 16. The manual process entailed extracting relative data from the BIM design models.

The time performance metric for the proposed methodology was based on the duration required to generate the simulation file (specifically, the ROS world file) starting from the BIM authoring tool. This comparison aimed to quantify the time-saving advantages of the automated methodology over traditional manual methods in preparing simulations for the assembly process.

Table 16. List of Steps for the Manual Extraction Process

Step	Manual Task Description
1	Identify all relevant building elements (i.e., stud, track, sheet) of the panels and their attributes from the BIM model.
2	Extract the geometrical dimensions, main axis parameters, object names, and local placement locations for all building elements.
3	Calculate the COG of the extracted building elements.
4	Create and model the building elements in the ROS simulation environment based on the extracted attributes.
5	Position the building elements in the material bay of the simulation environment.
6	Import the data in the robot controller for the framing and sheathing operation including nailing fixation.

Table 17. Result of the Validation for the Nailing Target Locations of Panel 1 of the Base Model

Location	2" x 6" Studs and Plates (3 nails per connection)														
	Proposed Method - Automated Coordinates (mm)					Traditional Method - Manual Coordinates (mm)					Variance - Difference %				
	Nx	Ny	Nz1	Nz2	Nz3	Nx	Ny	Nz1	Nz2	Nz3	Nx-diff	Ny-diff	Nzdiff-1	Nzdiff-2	Nzdiff-3
1	19	19.05	19.05	69.85	120.65	20.2	20.55	19.55	70.45	121.05	5.94%	7.30%	2.56%	0.85%	0.33%
2	59	19.05	19.05	69.85	120.65	60.2	20.55	19.55	70.45	121.05	1.99%	7.30%	2.56%	0.85%	0.33%
3	311	19.05	19.05	69.85	120.65	323.0	20.55	19.55	70.45	121.05	3.72%	7.30%	2.56%	0.85%	0.33%
4	406	19.05	19.05	69.85	120.65	418.0	20.55	19.55	70.45	121.05	2.87%	7.30%	2.56%	0.85%	0.33%
5	813	19.05	19.05	69.85	120.65	825.0	20.55	19.55	70.45	121.05	1.45%	7.30%	2.56%	0.85%	0.33%
6	1086	19.05	19.05	69.85	120.65	1136.0	20.55	19.55	70.45	121.05	4.40%	7.30%	2.56%	0.85%	0.33%
7	1126	19.05	19.05	69.85	120.65	1176.0	20.55	19.55	70.45	121.05	4.25%	7.30%	2.56%	0.85%	0.33%
8	1219	19.05	19.05	69.85	120.65	1269.0	20.55	19.55	70.45	121.05	3.94%	7.30%	2.56%	0.85%	0.33%
9	1222	19.05	19.05	69.85	120.65	1272.0	20.55	19.55	70.45	121.05	3.93%	7.30%	2.56%	0.85%	0.33%
10	1626	19.05	19.05	69.85	120.65	1676.0	20.55	19.55	70.45	121.05	2.98%	7.30%	2.56%	0.85%	0.33%
11	2032	19.05	19.05	69.85	120.65	2112.0	20.55	19.55	70.45	121.05	3.79%	7.30%	2.56%	0.85%	0.33%
12	2076	19.05	19.05	69.85	120.65	2156.0	20.55	19.55	70.45	121.05	3.71%	7.30%	2.56%	0.85%	0.33%
13	2116	19.05	19.05	69.85	120.65	2196.0	20.55	19.55	70.45	121.05	3.64%	7.30%	2.56%	0.85%	0.33%
14	2438	19.05	19.05	69.85	120.65	2518.0	20.55	19.55	70.45	121.05	3.18%	7.30%	2.56%	0.85%	0.33%
15	2845	19.05	19.05	69.85	120.65	2925.0	20.55	19.55	70.45	121.05	2.74%	7.30%	2.56%	0.85%	0.33%
16	3251	19.05	19.05	69.85	120.65	3351.0	20.55	19.55	70.45	121.05	2.98%	7.30%	2.56%	0.85%	0.33%
17	3958	19.05	19.05	69.85	120.65	4058.0	20.55	19.55	70.45	121.05	2.46%	7.30%	2.56%	0.85%	0.33%
18	4064	19.05	19.05	69.85	120.65	4189.0	20.55	19.55	70.45	121.05	2.98%	7.30%	2.56%	0.85%	0.33%
19	4470	19.05	19.05	69.85	120.65	4595.0	20.55	19.55	70.45	121.05	2.72%	7.30%	2.56%	0.85%	0.33%
20	4877	19.05	19.05	69.85	120.65	5002.0	20.55	19.55	70.45	121.05	2.50%	7.30%	2.56%	0.85%	0.33%
21	5239	19.05	19.05	69.85	120.65	5384.0	20.55	19.55	70.45	121.05	2.69%	7.30%	2.56%	0.85%	0.33%
22	5283	19.05	19.05	69.85	120.65	5428.0	20.55	19.55	70.45	121.05	2.67%	7.30%	2.56%	0.85%	0.33%
23	5690	19.05	19.05	69.85	120.65	5835.0	20.55	19.55	70.45	121.05	2.49%	7.30%	2.56%	0.85%	0.33%

24	5912	19.05	19.05	69.85	120.65	6057.0	20.55	19.55	70.45	121.05	2.39%	7.30%	2.56%	0.85%	0.33%
25	6053	19.05	19.05	69.85	120.65	6215.0	20.55	19.55	70.45	121.05	2.61%	7.30%	2.56%	0.85%	0.33%
26	6095	19.05	19.05	69.85	120.65	6257.0	20.55	19.55	70.45	121.05	2.59%	7.30%	2.56%	0.85%	0.33%
27	-6095	-2476.5	-2495.55	-2546.35	-2597.15	-6255.0	-2478.8	-2597.6	-2547.0	-2479.42	2.56%	0.09%	3.93%	0.02%	-4.75%
28	-6053	-2476.5	-2495.55	-2546.35	-2597.15	-6213.0	-2478.8	-2597.6	-2547.0	-2479.42	2.58%	0.09%	3.93%	0.02%	-4.75%
29	-5912	-2476.5	-2495.55	-2546.35	-2597.15	-6052.0	-2478.8	-2597.6	-2547.0	-2479.42	2.31%	0.09%	3.93%	0.02%	-4.75%
30	-5690	-2476.5	-2495.55	-2546.35	-2597.15	-5830.0	-2478.8	-2597.6	-2547.0	-2479.42	2.40%	0.09%	3.93%	0.02%	-4.75%
31	-5283	-2476.5	-2495.55	-2546.35	-2597.15	-5423.0	-2478.8	-2597.6	-2547.0	-2479.42	2.58%	0.09%	3.93%	0.02%	-4.75%
32	-5239	-2476.5	-2495.55	-2546.35	-2597.15	-5379.0	-2478.8	-2597.6	-2547.0	-2479.42	2.60%	0.09%	3.93%	0.02%	-4.75%
33	-4470	-2476.5	-2495.55	-2546.35	-2597.15	-4590.0	-2478.8	-2597.6	-2547.0	-2479.42	2.61%	0.09%	3.93%	0.02%	-4.75%
34	-4064	-2476.5	-2495.55	-2546.35	-2597.15	-4184.0	-2478.8	-2597.6	-2547.0	-2479.42	2.87%	0.09%	3.93%	0.02%	-4.75%
35	-3251	-2476.5	-2495.55	-2546.35	-2597.15	-3351.0	-2478.8	-2597.6	-2547.0	-2479.42	2.98%	0.09%	3.93%	0.02%	-4.75%
36	-2845	-2476.5	-2495.55	-2546.35	-2597.15	-2935.0	-2478.8	-2597.6	-2547.0	-2479.42	3.07%	0.09%	3.93%	0.02%	-4.75%
37	-2438	-2476.5	-2495.55	-2546.35	-2597.15	-2528.0	-2478.8	-2597.6	-2547.0	-2479.42	3.56%	0.09%	3.93%	0.02%	-4.75%
38	-2116	-2476.5	-2495.55	-2546.35	-2597.15	-2206.0	-2478.8	-2597.6	-2547.0	-2479.42	4.08%	0.09%	3.93%	0.02%	-4.75%
39	-2076	-2476.5	-2495.55	-2546.35	-2597.15	-2166.0	-2478.8	-2597.6	-2547.0	-2479.42	4.16%	0.09%	3.93%	0.02%	-4.75%
40	-2032	-2476.5	-2495.55	-2546.35	-2597.15	-2122.0	-2478.8	-2597.6	-2547.0	-2479.42	4.24%	0.09%	3.93%	0.02%	-4.75%
41	-1626	-2476.5	-2495.55	-2546.35	-2597.15	-1696.0	-2478.8	-2597.6	-2547.0	-2479.42	4.13%	0.09%	3.93%	0.02%	-4.75%
42	-1222	-2476.5	-2495.55	-2546.35	-2597.15	-1292.0	-2478.8	-2597.6	-2547.0	-2479.42	5.42%	0.09%	3.93%	0.02%	-4.75%
43	-1126	-2476.5	-2495.55	-2546.35	-2597.15	-1196.0	-2478.8	-2597.6	-2547.0	-2479.42	5.85%	0.09%	3.93%	0.02%	-4.75%
44	-1086	-2476.5	-2495.55	-2546.35	-2597.15	-1156.0	-2478.8	-2597.6	-2547.0	-2479.42	6.06%	0.09%	3.93%	0.02%	-4.75%
45	-406	-2476.5	-2495.55	-2546.35	-2597.15	-446.0	-2478.8	-2597.6	-2547.0	-2479.42	8.97%	0.09%	3.93%	0.02%	-4.75%
46	-311	-2476.5	-2495.55	-2546.35	-2597.15	-351.0	-2478.8	-2597.6	-2547.0	-2479.42	11.40%	0.09%	3.93%	0.02%	-4.75%
47	-59	-2476.5	-2495.55	-2546.35	-2597.15	-61.5	-2478.8	-2597.6	-2547.0	-2479.42	4.07%	0.09%	3.93%	0.02%	-4.75%
48	-19	-2476.5	-2495.55	-2546.35	-2597.15	-21.5	-2478.8	-2597.6	-2547.0	-2479.42	11.63%	0.09%	3.93%	0.02%	-4.75%

4.7.3 Time Performance Testing

The comparison between the proposed methodology and the manual process was conducted by recording the time taken by each approach. The duration required for the manual process was logged and is presented in Table 18. This data was further analyzed to determine the relationship between time consumption and the number of elements (such as studs) in a frame model, as illustrated in Figure 96. The analysis revealed a clear trend: as the complexity of the tested model increases, the total time required for the manual process also escalates.

Table 18. Time Performance of the Manual Process

Task #	**Task Description**	**Task Execution Time (mins)**			
		Panel 1	**Panel 2**	**Panel 3**	**Panel 4**
1	Identify all relevant building elements	20.7	16.2	14.9	18.6
2	Extract the information for all building elements	11.8	15.4	7.4	12.8
3	Calculate the Center of Gravity (COG)	8.3	11.60	5.6	9.9
4	Create and model the building elements in ROS	56.5	45.80	38.1	32.6
5	Position the building elements	16.1	14.90	7.9	13.8
6	Import the data in the robot controller	20.6	16.20	13.1	15.3
Total Timing (mins)		**134**	**120.1**	**87**	**103**

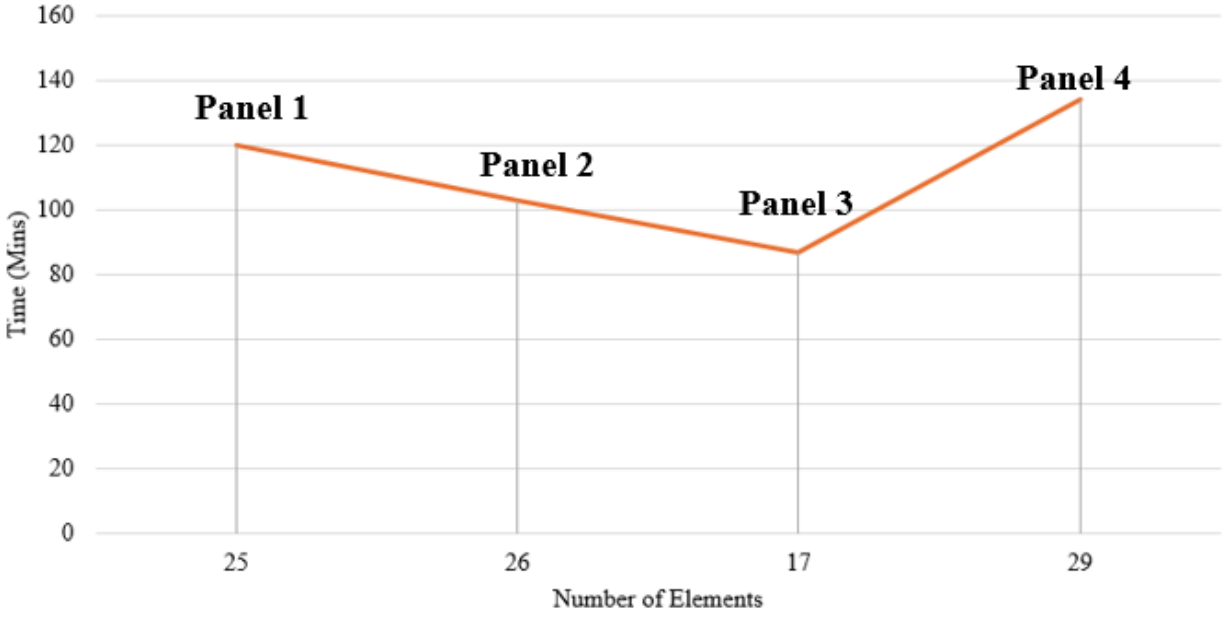


Figure 96. Plot of the Time Used in the Manual Approach Compared with the Number of Elements in a Frame for the Base Model

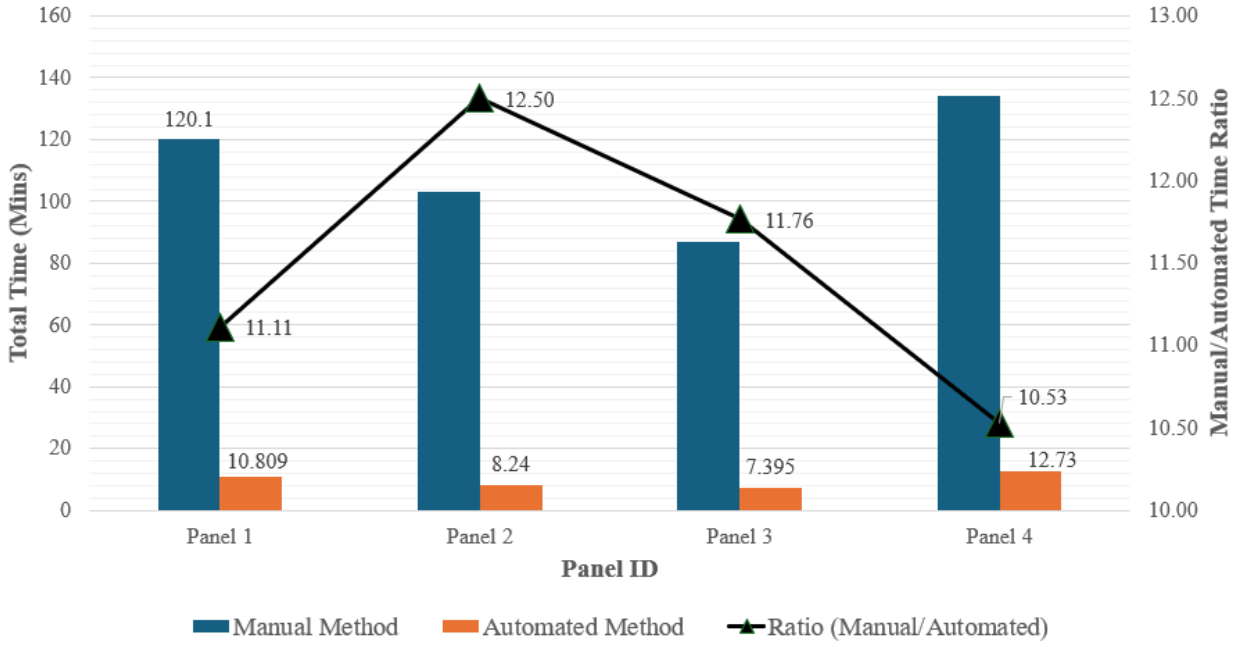


Figure 97. Time Comparison of the Manual Approach and the Automated Methodology

For the developed automated methodology, the time required was significantly less. Specifically, the durations recorded for the automated process were 10.80 minutes for Panel 1, 8.24 minutes for

Panel 2, 7.39 mins for Panel 3, and 12.73 minutes for Panel 4 as shown in Figure 97. This comparison effectively demonstrates the time efficiency of the proposed automated methodology over the manual process, particularly as the complexity and number of elements in the model increase.

Chapter 5 : Advances in Multi-Robot Task Allocation: From Manual Coordination to Algorithmic Optimization

Abstract: In recent years, Multi-Robot Systems (MRS) have garnered significant attention, propelled by their potential for a variety of real-world applications. Robotics now serves as a fundamental pillar in contemporary industrial manufacturing. A particularly compelling aspect of MRS research is Multi-Robot Task Allocation (MRTA), with a primary focus on enhancing the overall efficiency of robotic systems within production lines. A critical challenge within MRTA is achieving the optimal coordination of robotic movements for industrial tasks, aiming to optimize specific objective functions, such as reducing production cycle times. Currently, optimizing these movements in many industrial settings relies on manual efforts, which can be both costly and prone to errors. The complexity of this challenge is further amplified by factors such as redundant kinematics, the need for collision avoidance, and the potential for tasks to be executed in multiple ways. To address these issues, there has been considerable research dedicated to developing algorithms for the automatic calculation of optimal trajectories. This research offers a concise literature analysis of dynamic task allocation methods, categorizing MRTA applications and techniques into market-based, behavior-based, and optimization-based approaches. It delves into various criteria for evaluating task allocation strategies, including objective function optimization, system robustness, allocation and completion times, and the ability to reallocate tasks. The discussion highlights the strengths and weaknesses of these strategies, pointing out avenues for future research. Additionally, a statistical analysis sheds light on prevalent methods and the topic's evolution over time, identifying existing research gaps and suggesting promising directions for future studies in dynamic task allocation within multi-mobile robot systems. This article is intended for both academics and industry professionals, providing researchers with a

comprehensive summary of algorithmic strategies and offering practitioners insights into available solutions, organized by input and output parameters.

5.1 Introduction

Robotic systems increasingly substitute human labor in repetitive and hazardous tasks, with Multi-Robot Systems (MRS) being employed for their collaborative handling of complex activities. MRS utilize intentional cooperation, where diverse robots coordinate on tasks, and emergent cooperation, inspired by natural behaviors like swarming, for repetitive tasks over large areas (Cao et al., 1997).

Multi-Robot Task Allocation (MRTA), modelled in Figure 98, optimizes robot task assignments in dynamic environments, featuring challenges like moving obstacles and unpredictable events, necessitating adaptable algorithms for robot communication and learning (Gerkey and Matarić, 2004; Lerman et al., 2006).

This research reviews optimization strategies for MRTA, assessing recent developments and future research directions. It includes a survey of literature on MRTA applications, classifications, and optimizations from 2010 to 2023, emphasizing task interdependencies and temporal requirements (Gerkey and Matarić, 2004; Khamis et al., 2015; Quinton et al., 2023). The review covers task allocation strategies across various applications, focusing on problem constraints, objective functions, and methods for managing uncertainties.

In industrial applications, robots are leveraged to increase production efficiency by enhancing speed and optimizing task sequences. Significant research addresses optimizing robot control, velocity profiles, and collision-free trajectories (Werger et al., 2000; Zhao et al., 2016). Task

sequencing in robots often models problems like the Traveling Salesman Problem (TSP), which seeks the shortest route for a series of tasks while considering additional complexities like obstacle avoidance and task-specific constraints (Nagatani et al., 2009).

This study focuses on robotic task sequencing in industrial settings, exploring strategies for managing environmental obstacles and task complexities in known environments without complex logical relationships, specifically for single robots with articulated arms. The research covers offline sequencing algorithms, emphasizing sequencing methods with collision-free path planning.

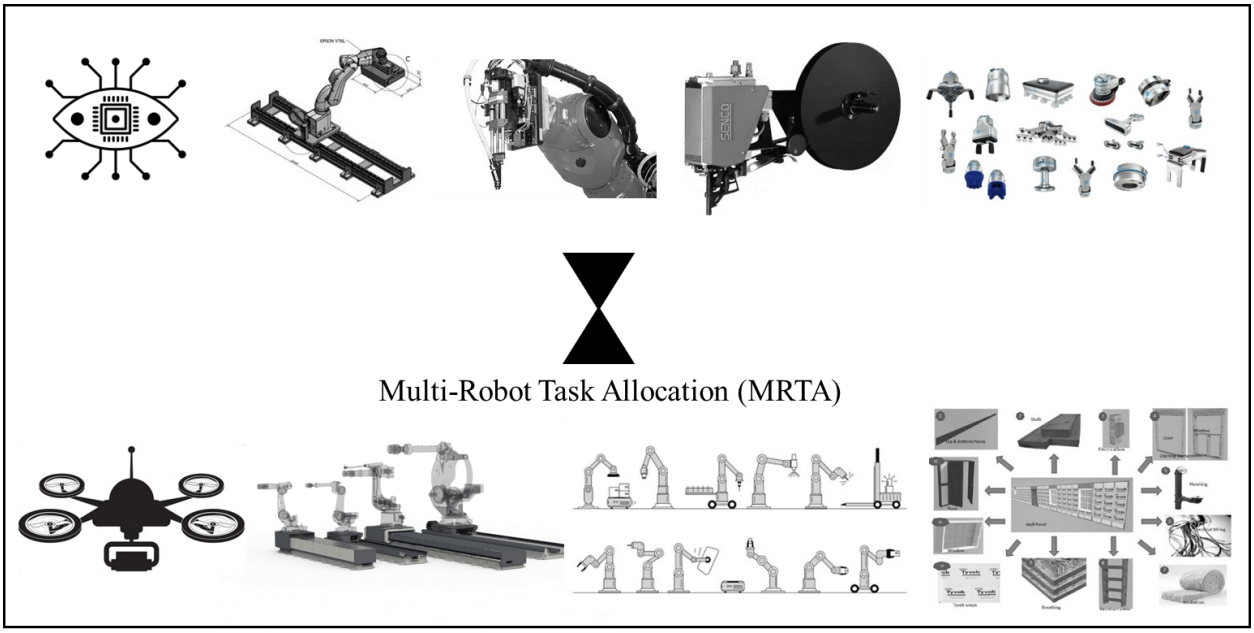


Figure 98. MRTA Problem Description

5.2 Literature Review

5.2.1 MRTA Taxonomies

Defining task types is crucial for modeling MRTA problems. According to Stentz and Zlot (2013), tasks generally fall into two categories: those performed individually by one robot, and those that are divided into sub-tasks executed by multiple robots. Tasks can be further classified as follows:

- **Elemental or Atomic tasks**: These are indivisible and cannot be broken down into sub-tasks.
- **Simple tasks**: These may either be elemental or decomposable, but if decomposed, all sub-tasks are allocated to the same robot.
- **Compound tasks**: These are divided into sub-tasks, with each sub-task performed by different robots. Each compound task has only one way of being decomposed.
- **Complex tasks**: These have multiple potential decompositions, and at least one decomposition involves multiple robots. The sub-tasks within a complex task can be simple, compound, or complex in nature.

To further categorize MRTA problems, Gerkey and Matarić (2004) proposed a taxonomy based on the capabilities of the robots, the requirements of the tasks, and the timing of task assignments:

- **Single-task robots (ST)/Multi-task robots (MT)**: Robots that execute only single task at a given time versus those capable of handling multiple tasks concurrently.
- **Single-robot tasks (SR)/Multi-robot tasks (MR)**: One robot per task versus multiple robots for execution.
- **Instantaneous assignment (IA)/Time-extended assignment (TA)**: Tasks assigned and performed immediately without future planning versus tasks planned and assigned over a longer time horizon.

Using this framework, MRTA problems can be distinctly characterized, leading to eight possible types of problems, such as MT-SR-IA, where robots can execute multiple tasks simultaneously, each task requires only one robot, and the allocation is made instantaneously. This classification helps in understanding and solving various MRTA challenges, as shown in Figure 99.

This study reviews major developments in robot task sequencing over the last 13 years, focusing on TSP-like problems applied in robotics (Giordani et al., 2010; Kanakia et al., 2016; Stentz, 2006). It classifies problems by input and output parameters, helping practitioners choose appropriate methods and guiding researchers toward unresolved issues and potential research directions.

Cooperative MRS are extensively studied for their ability to handle complex tasks more efficiently than single robots by collaborating (Fang et al., 2018; Palmer et al., 2018). Multiple robots offer advantages such as fault tolerance and the ability to reconfigure in response to failures, finding uses in industries like manufacturing and construction (Liu et al., 2017; Rishwaraj and Ponnambalam, 2017).

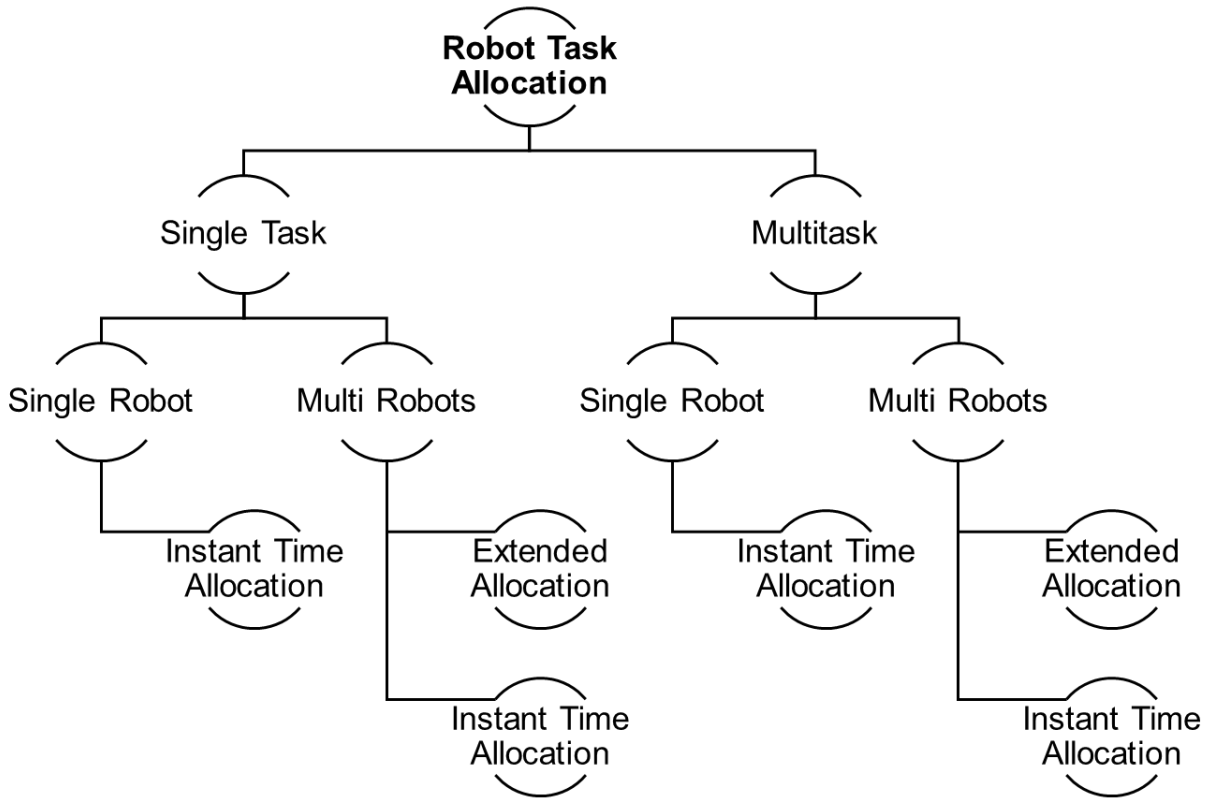


Figure 99. Basic MRTA Taxonomy

MRS coordination can be either centralized, where a server manages tasks allocation but risks failure if the server goes down, or distributed, where robots independently manage tasks, suitable for larger teams in unstable communication environments (Johnson et al., 2016; Semwal et al., 2017). Each method has its trade-offs related to communication reliability and consensus on task distribution (Giordani et al., 2010).

Task allocation in MRS has evolved, with Gerkey and Mataric (2004) initially identifying eight types, later simplified to four by Korsah et al. (2013) in their iTax classification, which factors in resource interdependencies and task costs (refer Figure 100). Strategies for task allocation range from auction-based to optimization-based, with the latter often providing quicker, more optimal solutions for complex scenarios (Darmanin and Bugeja, 2017).

This model identifies four types of dependencies affecting the costs:

- **No Dependencies (ND)**: The cost of each robot-task pair is independent of any other.
- **In-Schedule Dependencies (ID)**: The cost of a robot-task pair depends on other tasks assigned to the same robot, reflecting intra-schedule dependencies.
- **Cross-Schedule Dependencies (XD)**: The cost is influenced not only by the tasks assigned to the same robot but also by the tasks scheduled for other robots, indicating inter-schedule dependencies.
- **Complex Dependencies (CD)**: Costs are affected by the schedules of other robots, involving complex interactions between task decomposition and allocation.

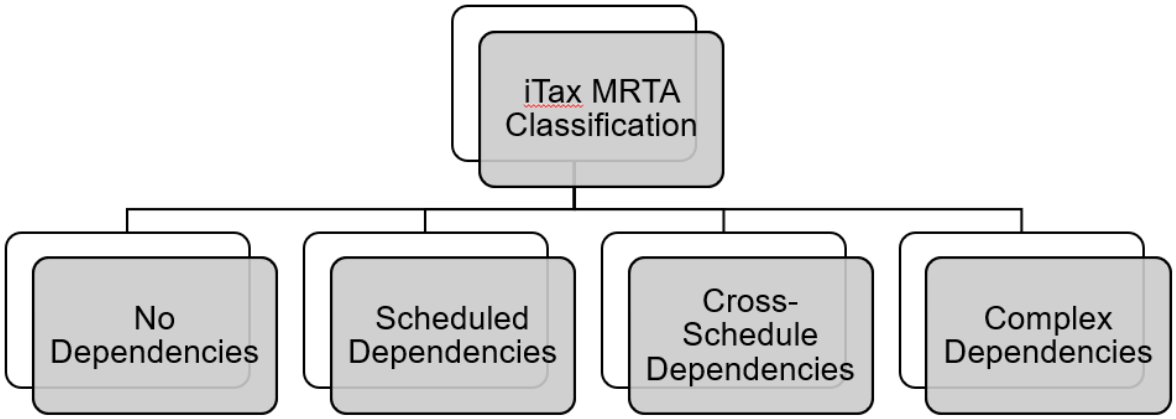


Figure 100. iTax Classification

Further refining task assignment's time and sequence aspects, Nunes et al. (2017) developed the MRTA-TOC taxonomy under the Time Extended (TA) assignment framework. This taxonomy categorizes constraints into two types as shown in Figure 101:

- **Time Windows (TW)**: This represents the time intervals during which tasks can start and finish, with specified earliest and latest permissible times.
- **Synchronization and Precedence constraints (SP)**: These constraints dictate the order in which tasks must be executed, creating cross-schedule dependencies that affect scheduling across multiple robots.

This enhanced approach provides a structured framework for analyzing MRTA problems, particularly in complex scenarios where the interdependencies and timing of tasks play critical roles.

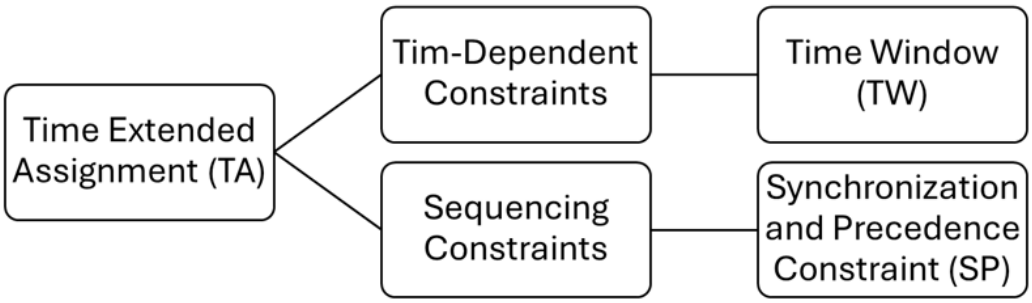


Figure 101. MRTA-TOC Taxonomy

5.2.2 MRTA Problem Definition

This section describes the MRTA problem. Define $J_A = \{j_1 j_{2,}, j_{3,}, \ldots\ldots, j_{n,}\}$ and $J_B = \{j_1 j_{2,}, j_{3,}, \ldots\ldots, j_{m,}\}$ as the set of tasks to be allocated, and $R_A = \{r_{1,} r_{2,}, r_{3,}, \ldots\ldots, r_{n,}\}$ and $R_B = \{r_{1,} r_{2,}, r_{3,}, \ldots\ldots, r_{m,}\}$ as the team of robots. In equation (1), the variable *AA* signifies the assignment of the task set *JJ* to the robot set *RR* as in Equation 1.

$$A: f \rightarrow R \tag{1}$$

If a robot *r* is allocated a task *i*, then task allocation problem can be modelled as $A_{i,r}$ =1 else $A_{i,r} = 0$.

- Let $M_{i,}$ [*M* is the matrix of required utility values to execute *m* tasks by *n* robots].
- Let $M_{r,}$ [*M* is the matrix of available utility values to execute *m* tasks by *n* robots].
- Let $T_s = \{T_{s1,} T_{s2,}, T_{s3,}, \ldots\ldots, T_{sx,}\}$ be the initial time of *x* tasks.
- Let $T_w = \{T_{w1,} T_{w2,}, T_{w3,}, \ldots\ldots, T_{wx,}\}$ be the waiting time of the tasks to start.
- Let $T_\Delta = \{T_{\Delta 1,} T_{\Delta 2}, T_{\Delta 3}, \ldots\ldots, T_{\Delta x}\}$ be the set of time span of the tasks.
- Let $D_{i,r}$ be the linear distance travelled by the robot *r* to execute the task *i*.
- Let $C = \{C_{1,} C_{2,}, C_{3,}, \ldots\ldots, C_{x,}\}$ be the set of completed tasks.

Task assignment in multi-robot systems is a decision-making problem optimized under key

constraints. These constraints are detailed in Table 19. Fundamentally, the dynamic task allocation problem involves managing resources and adhering to time constraint.

- A task must be assigned to a robot that has adequate utility, as explained in Equation (2):

$$If\ M_i > M_r\ then\ A_{i,r} = 0, \forall \in f\ and\ r \in R \quad (2)$$

- The task execution time T for tasks assigned to a robot must not overlap. This constraint, shown in Equation (3), accounts for the start time, waiting time, and duration of the tasks:

$$If\ T_{ip} \geq T_{iq} + \left(W_{iq} + M_{iq}\right) where\ i_p and\ i_q \in f\ and i_p i_q \quad (3)$$

The task $j_p$must be schedule after the task i_q

- The task assignment must be conflict-free. A task must be assigned to one robot as specified in Equation (4):

$$\sum_{r \in R} T_{j,r} = 1\ \forall_j \in f \quad (4)$$

The most common objective functions Equations 5, 6, and 7 of dynamic task allocation strategy is to: minimize the robot travel distance (d), the robot waiting time (W) and to maximize the robot task completion rate (K):

$$\min : \sum_{i \in f} \sum_{r \in R} D_{i,r} \quad (5)$$

$$\min : \sum_{i \in f} W_j \quad (6)$$

$$\max : \sum_{l \in f} |K_l| \quad l \leq m \quad (7)$$

This study addresses the multi-robot dynamic task allocation problem as a combinatorial optimization challenge. A comprehensive review of existing strategies for solving this problem,

with detailed discussions to follow in the subsequent sections.

Table 19. Constraints Related to the Dynamic Task Allocation Problem

Constraints Type	**Related To**
Working Space	Dynamic obstacles, unknown surrounding, cluttered spaces, etc.
Robot Hardware	IOT malfunction, network failure, inaccurate travel distance, computational capacity, resource constraint.
Task Execution	Time-constrained tasks, dual-robot tasks, sequential tasks and task variants.

5.2.3 Optimization Using Combinations

MRTA leverages techniques from operations research, AI, ML, software engineering, applied mathematics, and computer science to optimize complex systems. AI and ML enable autonomous robot task assignment, helping robots learn from past actions and adapt to changes. MRTA involves discrete optimization, often using graph structures and mathematical modeling to identify the best solutions from a limited set of options, guided by an objective function and constrained by a finite search space.

Resolution methods in MRTA are divided into exact and approximate approaches. Exact methods, like dynamic programming, solve problems by dividing them into smaller sub-problems solved recursively, combining solutions for an optimal outcome, especially effective under uncertainty and communication constraints (Chen et al., 2021). Branch and Bound (BnB) also solves problems by iteratively resolving sub-problems and branching based on solution integrality (Martin et al., 2021).

Approximate methods provide practical solutions for large-scale optimization when real-time results are necessary, often initializing exact methods. These include heuristics, which use experiential shortcuts for quick problem-solving, and metaheuristics, which start with a feasible solution set and iteratively improve it, aiming for gradual enhancements over time (Blum and Roli, 2003).

Key approximate methods include:

- **Constructive methods** such as the GA build a solution piece by piece based on local optimization criteria (Kong et al., 2019).
- **Local search algorithms**, including SA and Tabu Search, which explore the solution space by making incremental changes to a single solution (David and Rögnvaldsson, 2021).
- **Evolutionary algorithms** like the GA, PSO, ACO, and Bee Colony Optimization (BCO), which simulate natural evolutionary processes to find optimal solutions (Shelkamy et al., 2020).

5.2.4 Related Planning Problems in Robotics

This study examines offline task sequencing algorithms for robots, focusing on assigning tasks defined as areas rather than specific object interactions, in known environments.

- Production Scheduling: Unlike task sequencing, production scheduling optimizes the operation of production facilities, focusing on resource allocation and work distribution to maximize efficiency, typically disregarding collision and kinematic factors (Dawande et al., 2007).
- Multi-Robot Task Planning: This involves distributing tasks among multiple robots to

optimize overall efficiency. Earlier methods include Maimon's (1990) clustering graph-based approach, with more recent advancements like the Stochastic Clustering Auction combined with SA to minimize local optima risks (Zhang et al., 2012, 2013).

- Task-Level Planning: Focuses on defining robotic actions based on their interactions with the environment. For instance, tasks might involve moving an object from one location to another, broken down into a sequence of specific actions. This level of planning often utilizes domain-specific planners and models to simulate tasks and interactions within the environment (Cao et al., 1991; Chien et al., 1997, 1998).
- Combining Task-Level and Path Planning: Sometimes task plans must be adjusted if path planning cannot find a collision-free route. Integrated systems that combine both planning levels help bridge the gap between symbolic and physical execution, ensuring task feasibility (Bhatia et al., 2011; Gaschler et al., 2013).
- Online Control-Based Planning: Suitable for dynamic environments where task constraints may change in real-time. This approach uses sensors and control architectures to dynamically adjust task execution based on new data, adding or removing tasks as needed (Mansard and Chaumette, 2007).
- Manipulation Planning: Concerns the planning needed for robots to relocate objects within their operational environment, considering factors like object properties and spatial constraints. This includes advanced strategies for handling specific challenges like deformable objects or complex grasping techniques (Diankov et al., 2013; Berenson, 2013).

5.2.5 Preliminaries in Planning Tasks in Robotics

This section provides an overview of the foundational concepts in robotics relevant to modelling task sequencing, particularly focusing on how robot poses are defined and utilized in optimization problems.

In robotics, defining accurately the pose of a robot's end-effector is crucial for task sequencing. It involves describing it in the T-space (Task Space) and the C-space (Configuration Space). However, before engaging in any optimization processes, defining the pose is typically described in two key spaces:

(1) **T-space**: This space is utilized to define the position and orientation of the robot's end-effector. Specifically, the task space, denoted as *TS(3)*, integrates the spatial position in three dimensions, *SR(3)*, and the orientation space, *OS(3)*, resulting in $TS(3)=SR(3)\times OS(3)$. Positions and orientations in this space are commonly represented using homogeneous coordinates, a method that facilitates calculations and transformations in robotics (LaValle, 2006). Since rotation matrices can be non-intuitive for human interpretation, Euler angles are often employed as a more comprehensible alternative. Euler angles describe a series of rotations around the coordinate system's axes, requiring three angles to define any 3D space rotation—one for each axis (Craig, 2005). The pose of the end-effector in T-space is represented by a rotation matrix describing the end-effector's orientation and a translation vector indicating its position.

$$M = \begin{bmatrix} R_{3X3} & T_{3X1} \\ 0_{1X3} & 1 \end{bmatrix} \tag{8}$$

(2) **C-space**: This space represents the set of all possible configurations of the robot's joints or axis space. For standard industrial robots, which typically have six DOF, the C-space can be expressed as $C = R6$.

The relationship between the two spaces is articulated through two functions:

(1) **Forward Kinematics (FK)**: The robot joint angles is the input of this function while the end-effector position and orientation are the calculated outputs as $FK: C \rightarrow SR\ (3)$.

(2) **Inverse Kinematics (IK)**: The end-effector's position and orientation are the input of this function while and the robot's joints are computed for all possible configurations to achieve this pose, $IK: SR(3) \rightarrow C$. A typical 6-DOF industrial robot can have up to eight solutions.

An example illustrated in Figure 102 shows how one task point can be reached by two different robot configurations, demonstrating the practical application of these kinematic mappings in real-world scenarios.

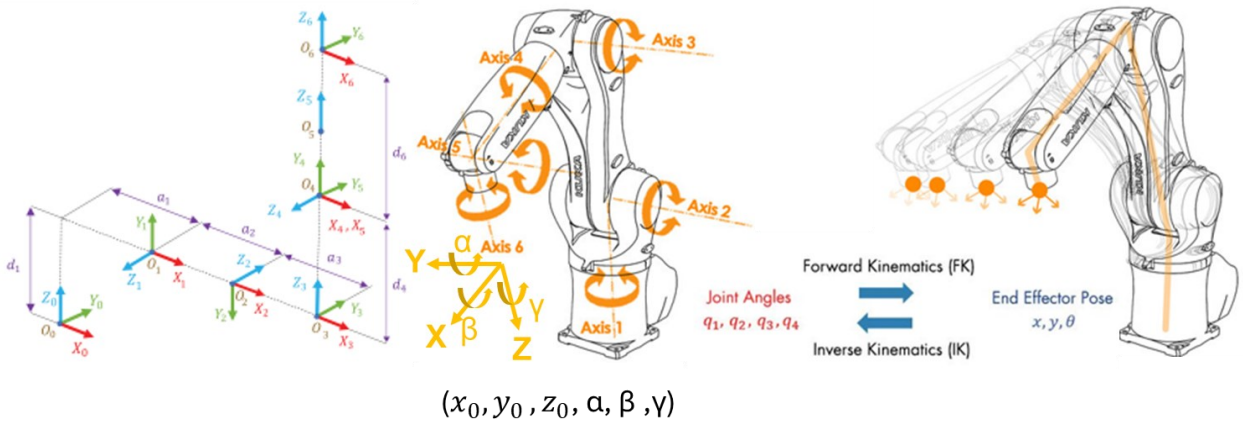


Figure 102. T-space Point Reached with Two Robot Configurations

While task sequencing in IR settings, robot programming can be executed online or offline, each method presenting unique benefits and limitations.

(1) **Online Programming:** This method involves directly teaching robots movements to replicate during production tasks such as pick and place. Programmers manually guide the robot, heavily relying on their skill and intuition to define movements, which can result in a process that lacks optimization and is time-intensive due to its reliance on manual input (Siciliano and Khatib, 2008; Pan et al., 2010).

(2) **Offline Programming:** This method uses simulations to create and refine robotic paths before real-world execution, allowing for pre-execution adjustments like collision checks to protect equipment. Despite the ability to test trajectories, optimization of task sequences in simulations is often limited. For example, simulation software like RobotWorks lacks complex sequencing capabilities, and DELMIA's adaptation for drilling uses a basic approach that often results in suboptimal sequences (Pan et al., 2010).

Despite advancements in simulation technology, task sequencing remains largely manual. This discussion will focus on sequencing approaches in offline programming, particularly exploring combinatorial problems foundational to task sequencing in robotics. These include specifying tasks either as points or areas, with outputs ranging from simple sequences to detailed paths including specific entry points for each task, as illustrated in Figure 103.

MRTA problems are often modeled as Mixed-Integer Nonlinear Programs (MINLP), which are effective for small sets of tasks but not scalable for larger ones, leading to the use of heuristic approaches. These heuristics, while not always optimal, are efficient at finding near-optimal solutions and can be categorized in two groups: tour-construction (e.g., Insertion Heuristic) and tour-improvement (e.g., 2-Opt, 3-Opt, Tabu Search) (Johnson et al., 1997). In robotics, these heuristics must also account for robot-specific elements like inverse kinematics and collision-free

planning. Effective methods for robotic task optimization include the PSO for TSP and GTSP, and various approximation algorithms for problems like the Traveling Salesman Problem with Neighborhoods (TSPN).

5.2.6 Planning Tasks in Robotics (Input/Output)

Input: Basic Tasks: This research tackles well-known combinatorial challenges, such as the TSP, which aims to identify the shortest route given that the robot passes only once through each point and revert to the initial point. This problem is known for its computational complexity (Montemanni et al., 2008). Asymmetric TSP (ATSP), where travel costs between points vary with direction, adds another layer of complexity (Applegate et al., 2007). Variants include the Sequential Ordering Problem (SOP), which imposes additional constraints such as visiting certain points before others, and the Shortest Sequence Problem (SSP), where the route doesn't need to return to the start. Extending SSP to sets of points leads to the Generalized TSP (GTSP), aiming to visit one point from each set in the shortest possible loop (Srivastava et al., 1969).

Output: Path: The Multi-Goal Path Planning Problem, a variation of the GTSP tailored for robotics, involves determining a minimal-cost, cyclic, collision-free path given that the robot passes only once through each point from each set of inverse kinematics solutions for a given T-space goal (Wurll et al., 1999).

Input: Advanced Tasks

Output: task entry points for each sequence: The Touring-a-sequence-of-Polygons Problem (TPP) is for tasks defined not just as points but as polygons. The objective to determine a minimal-cost cyclic tour that visits a predefined sequence of regions (Dror et al., 2003). Extensions of TSP like

the Close-Enough TSP (CETSP) and the more general TSPN involve visiting areas within certain proximities or regions, respectively. The most complex variation, the GTSPN, involves visiting at least one neighborhood from each cluster.

Output: Path: When the tour in the TSPN must adhere to a specific boundary, the problem is described as the Safari Route Problem (SRP). If the tour is restricted from traversing within predefined convex areas but is required to visit their borders only, it is termed as the Zookeeper Route Problem (ZRP). When obstacles are incorporated into the TSPN and the regions are represented as convex polygons, the problem is known as the Multi-Target Pathfinding for Goal Regions (MTPGR). Generally, the MTPGR is the most comprehensive and challenging variant to solve computationally.

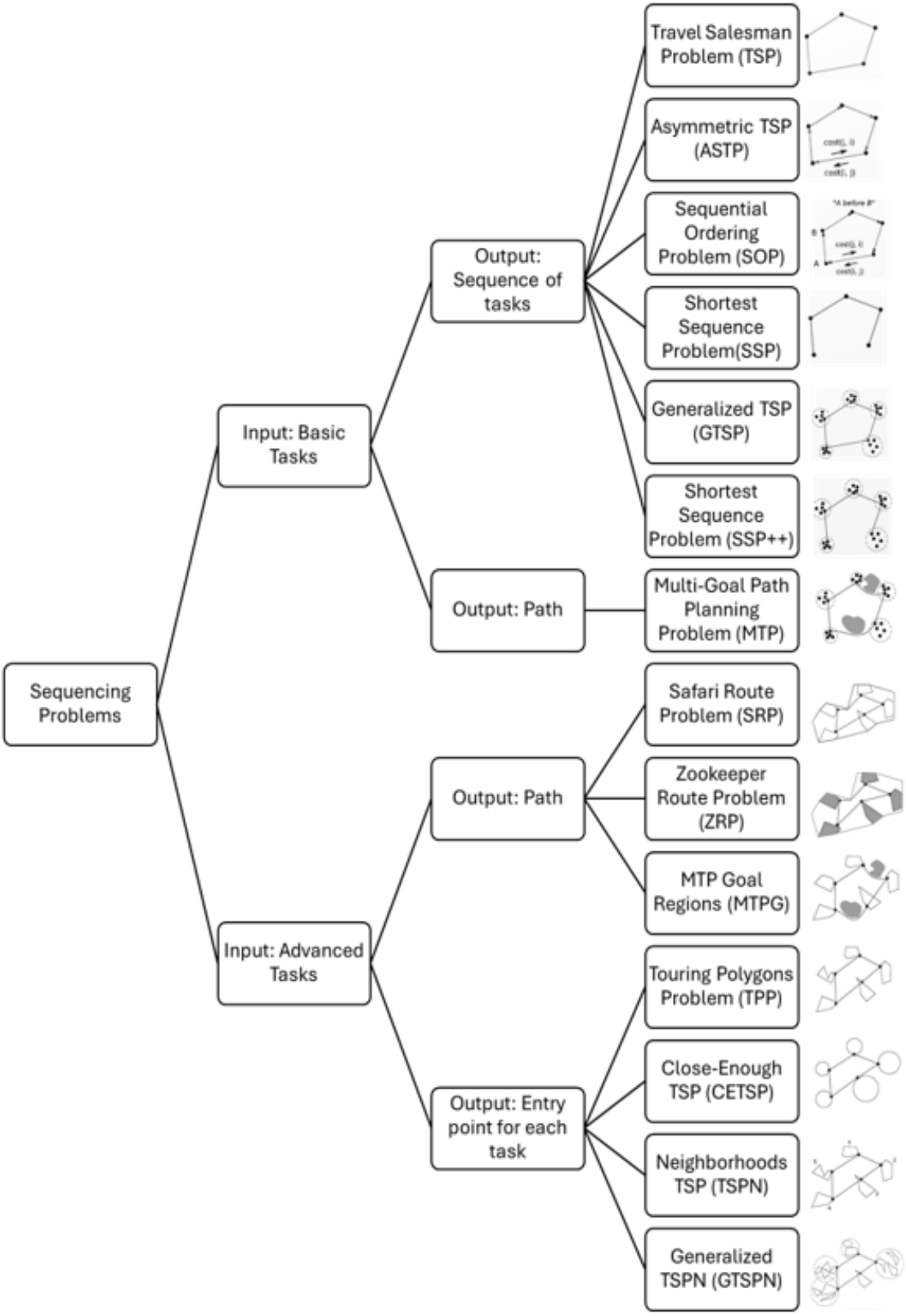


Figure 103. Categorization of Sequencing Problems

5.2.7 Dynamic Task Allocation Strategies

Multi-robot dynamic task allocation focuses on efficiently assigning tasks to robots while minimizing costs and meeting various constraints, often irregular and diverse in real-world applications (Sarkar et al., 2018a). Task allocation methods are classified into four main strategies, detailed in Figure 104. This study reviews these strategies, evaluating them across multiple dimensions including application-specific challenges, objective functions, coordination types, taxonomy, task reallocation capabilities, and methods for managing uncertainty. It also considers the number of tasks and robots, average times for task allocation and completion, and whether strategies have been validated through simulations or real-world testing. This comprehensive analysis helps identify the most suitable strategies for specific scenarios.

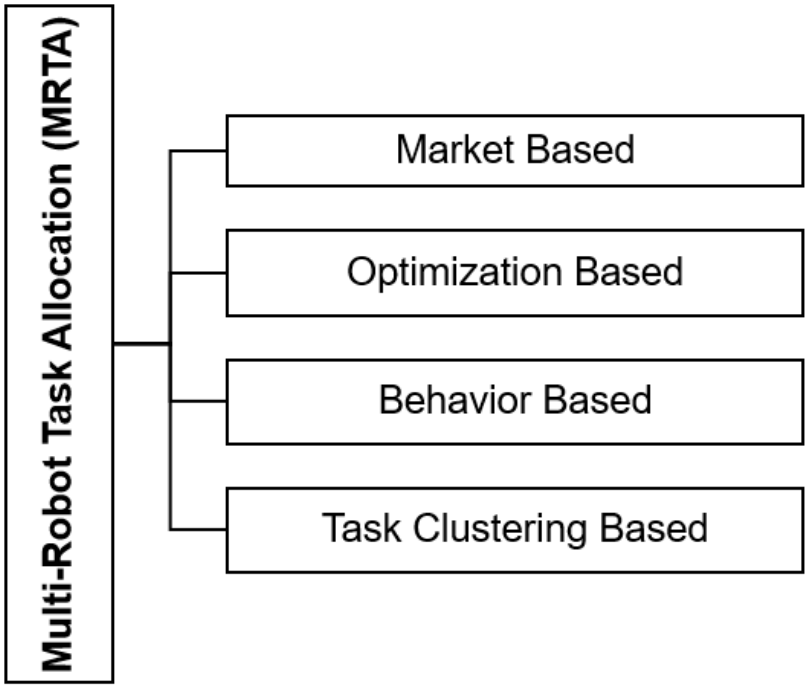


Figure 104. MRTA Classifications

Market-Based Task Allocation: Inspired from market trading principles, multi-robot systems involve an auctioneer robot that broadcasts tasks and collects bids from other robots based on their capability to execute the tasks efficiently (Schneider et al., 2017; Luo et al., 2015). Displayed in Figure 105, the auctioneer assigns tasks to the lowest bidders, optimizing cost efficiency within both centralized and distributed coordination setups. These strategies are divided into single-item and combinatorial auctioning, with the latter grouping tasks together to improve distribution efficiency (Otte et al., 2020). Advanced methods like the Consensus-Based Bundle Algorithm (CBBA) integrate auctions with consensus processes to ensure conflict-free task distribution, often yielding optimal or near-optimal solutions.

To address uncertainties, such as fluctuating task conditions and robot status, these strategies periodically reassess and reallocate tasks based on updated cost estimates and diagnostic data, improving reliability and product output (Turner, 2018; Talebpour and Martinoli, 2018). However, this can lead to increased computational demands and energy consumption. The auction process, while reducing travel distances for robots, depends heavily on robust communication networks, losing effectiveness in environments with unreliable communications.

Despite their theoretical benefits, these market-based strategies are predominantly tested in simulations rather than real-world settings, highlighting a gap in practical application knowledge. This discrepancy points to the need for more real-world trials to validate simulation-based findings.

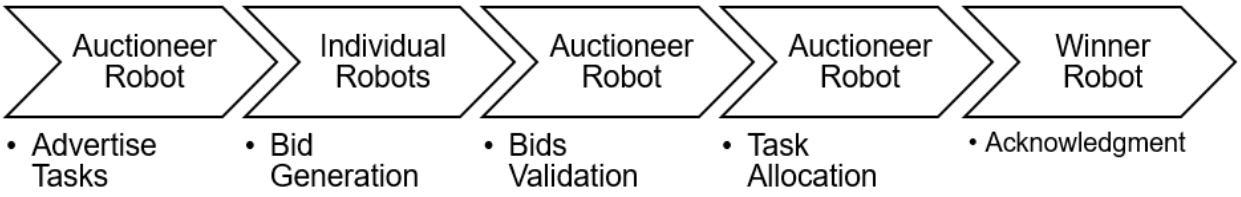


Figure 105. Market-based MRTA Process

Optimization-Based Task Allocation: Optimization-based strategies excel in task allocation for multiple robots, addressing the uncertainties and dynamics of real-world environments with efficient computational approaches (Badreldin et al., 2013; Li and Yang, 2018). These strategies, using evolutionary algorithms like GA, PSO, and ACO, handle complex problems often modeled as variants of the multiple traveling salesman problem (Arif and Haider, 2017). The main objectives of dynamic task allocation are to minimize completion times, travel distances, and energy consumption while maximizing task distribution and completion rates, typically tackled through single-objective optimization.

Despite their efficiency, these strategies can struggle with the flexibility needed to switch between multiple solutions and generally require significant computational resources (Shenoy and Anupama, 2017). Multi-objective optimization offers a solution, potentially enhancing task completion rates more effectively by simultaneously addressing multiple goals. However, designing suitable fitness functions for multi-objective optimization is challenging, especially when objectives conflict, necessitating careful weighting of each factor.

Multi-objective strategies often achieve higher task distribution rates than single-objective methods, providing better scalability but may suffer from limited adaptability to varying performance goals. There's also a notable research gap in comparing the effectiveness of these

strategies in simulated versus real-world settings, indicating a need for further investigation to standardize and adapt fitness factors for diverse task allocation scenarios.

Behaviour-Based Task Allocation: Behavior-based dynamic task allocation segments tasks into specific behaviors, allowing robots to act according to predefined criteria and priorities (dos Reis et al., 2021). This approach adapts flexibly to the situation at hand, incorporating mathematical models, heuristics, and optimization functions as needed. In applications like multi-robot exploration, it employs a two-tiered control architecture. High-level behaviors handle task identification and inter-robot communication, while low-level behaviors manage obstacle navigation and task switching (Chetty et al., 2011; Schillinger et al., 2018). Suitable for both centralized and distributed coordination, this method excels in environments with unreliable communications, enhancing system robustness against potential robot failures through dynamic task switching. Behavior-based approaches are ideal for uncertain, constantly changing environments seen in real-world multi-mobile robot (MMR) applications (Lee and Kim, 2019). Techniques like ALLIANCE optimize task scheduling to minimize completion times, while BLE adapts to new tasks dynamically. Additionally, Markov decision processes help manage dependencies and timing constraints in MRTA scenarios.

Clustered Task Allocation: It involves grouping similar or geographically close tasks into grouped clusters for assignment to robots, optimizing travel distances and resource use (Chen et al., 2018b; Sarkar et al., 2018b). This method is effective in scenarios like search and rescue or warehouse operations, where tasks are naturally proximate. Techniques such as Euclidean distance clustering, K-means, and fuzzy clustering are used to form these clusters (Ghassemi and Chowdhury, 2018). A challenge lies in determining the ideal number of tasks per cluster to balance efficiency and

effectiveness, as explored in studies by Mitiche et al. (2019). While clustering minimizes travel distances and computational complexity by reducing the number of tasks needing direct allocation, it's best for centralized systems and less effective in poor communication environments. The strategy also risks increasing the impact of robot failures on task completion, which can be countered by integrating task switching mechanisms that reallocate tasks to maintain continuity (Mitiche et al., 2019).

5.2.8 MRTA Related Work

MRTA problems utilize various optimization strategies, including Integer Linear Programming (ILP), heuristics, and metaheuristics, to address task allocation effectively:

- **Hungarian Algorithm (HA):** First introduced by H. W. Kuhn in 1955, HA is a method to solve the Linear Assignment Problem (LAP) optimally with O(n^3) complexity. Kasahara et al. (2020) applied HA to control mobile robots in warehouse environments, using a discrete-event system model to manage tasks dynamically. Giordani et al. (2010) adapted HA for decentralized environments, allowing robots to independently manage tasks without a centralized controller.
- **ILP and MILP Models:** These mathematical models optimize functions involving discrete and continuous variables. They have been applied to diverse scenarios like coordinating robots in unknown environments (Atay and Bayazit, 2006) and disaster response (Koes et al., 2006). However, their computational demands can be substantial, especially in large-scale and diverse robot environments.
- **Bio-inspired Methods:** Algorithms like GA, ACO, and PSO are popular in MRTA. For instance, Chopra et al. (2017) combined GA with A* for centralized task allocation and

path planning, while Saeedvand et al. (2019) integrated GA with constraint k-medoids for clustering tasks focusing on energy efficiency. Alitappeh and Jeddisaravi (2022) optimized robot cooperation and efficiency using GA.

- **Clustering Methods:** Employing clustering helps minimize travel distances and even out the workload among robots. Techniques like K-means are used to partition tasks into clusters, subsequently optimized with HA or GA for task distribution (Elango et al., 2010; Janati et al., 2017). Ghassemi and Chowdhury (2018) introduced a fuzzy clustering approach that allows tasks to belong to multiple clusters, enhancing flexibility in task allocation.

These methodologies highlight the complexity and multidisciplinary nature of MRTA solutions, each suited to specific operational contexts and challenges.

5.3 TSP Problem Modelling

In robotics, task sequencing complexities are often reduced by breaking down problems into simpler models, allowing for feasible optimization within realistic timeframes. This includes solving problems like the TSPN before addressing collision-free paths. The detailed factors affecting robotic task sequencing are:

- **Multiple Inverse Kinematics Solutions:** Robots can achieve the same target point through different configurations, increasing complexity but offering optimization possibilities. Optimal configurations minimize movement and avoid obstacles.
- **Obstacle Avoidance:** Simple point-to-point cost calculations become complex in cluttered environments requiring collision-free routes. Integrating collision-free planning into sequencing algorithms is a major challenge.

- **Robot Base and Workspace:** The robot's base location relative to task sites significantly affects sequencing. Changes in robot location may require complete re-optimization.
- **Precedence Diagram:** Certain tasks must follow a specific order, essential for optimizing sequencing.
- **Task Specification:** Tasks vary in complexity and are typically represented as points or configurations in task space. For instance, routing tasks, while seeming to correspond to specific configurations, can often be achieved through multiple setups, offering optimization potential. Some models extend to represent tasks as 2D-3D areas or closed contours.
- **Objective Function:** Optimization objectives focus on minimizing time for movements, path lengths in Cartesian or configuration space, or specific industry goals like material waste reduction. In kinematic-focused models, the objective may involve minimizing Cartesian path length.

This approach ensures adaptability and effective implementation across various operational environments, addressing specific robotic task sequencing complexities, as illustrated in Figure 106.

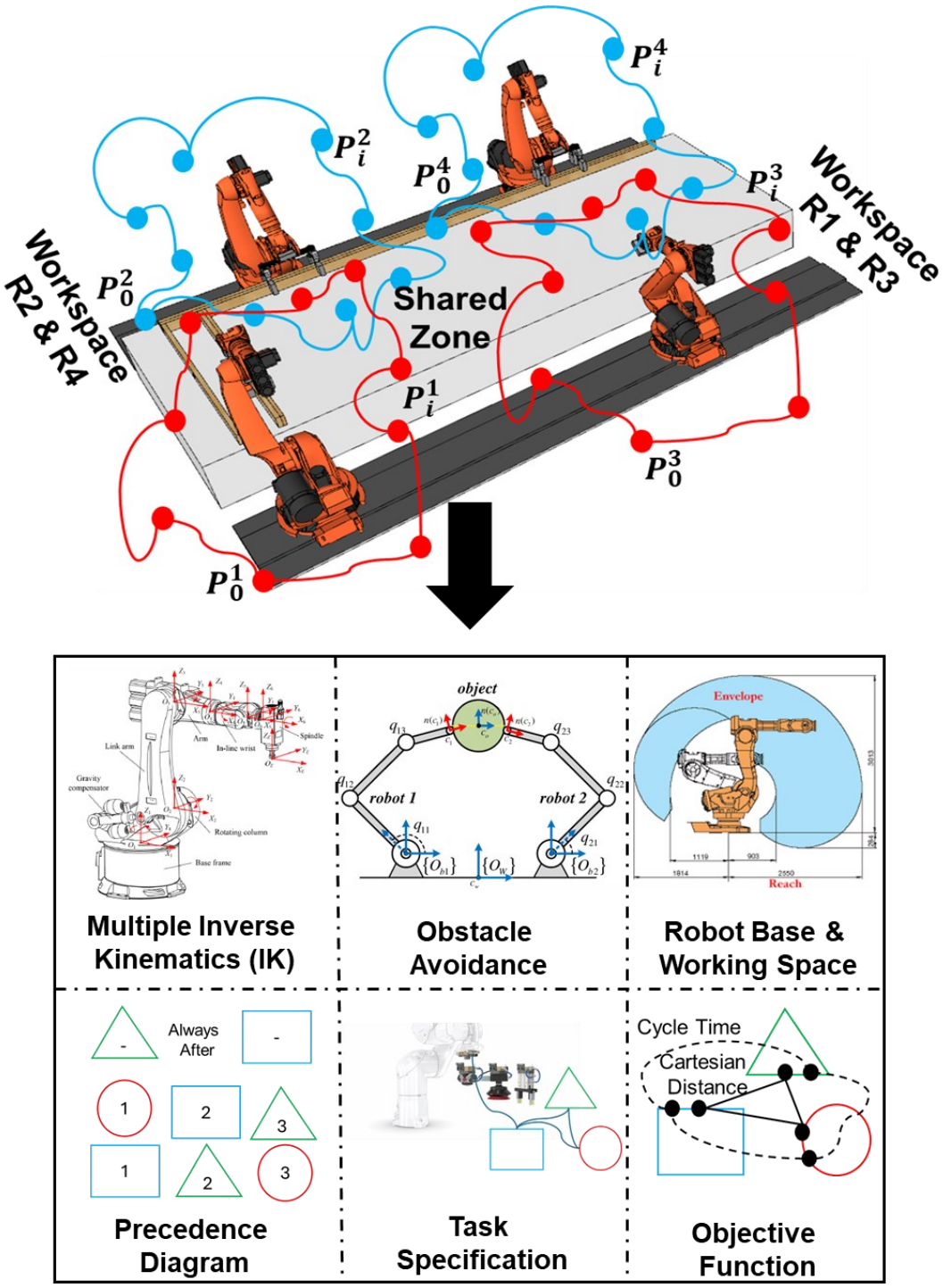


Figure 106. Robotic Task Sequencing Problem

5.4 Results and Discussions

This study reviews dynamic task allocation strategies for multi-robot systems (MMRs), noting a research gap between simulated validations and real-time experiments.

- **Market-Based Task Allocation**: Robots optimize task sequences for profitability. Traditional methods like sequential single-item and parallel auctioning rely on pre-defined task lists, but adaptive auctioning updates tasks dynamically, enhancing flexibility in changing environments. A key challenge is maintaining robust communication networks essential for effective allocation. Single-point auctioneer reliance poses a vulnerability, necessitating strategies to mitigate this risk.
- **Optimization-Based Task Allocation:** Utilizes various algorithms based on fitness functions. Current solutions lack robustness, highlighting the need for refined adaptive fitness functions to manage uncertainties better. Multi-objective optimization tends to outperform single-objective approaches, but developing effective multi-objective cost functions to balance competing objectives remains crucial.
- **Behavior-Based Task Allocation**: Offers adaptability to different constraints and uncertainties, improving robustness and scalability. The main challenge is managing multiple behaviors, which requires substantial computational resources. Recommended for uncertain scenarios.
- **Task Clustering-Based Allocation**: Effective in minimizing travel distances, especially in autonomous surveillance. Determining the optimal number of tasks per cluster is unresolved. Strategies to switch between clusters in response to uncertainties, like robot failures, are needed.

- **Hybrid Task Allocation Strategies**: Combining various allocation strategies presents opportunities for handling complex, dynamic environments more effectively by leveraging the strengths of each approach.

5.4.1 Research Directions and Performance Analysis

Further research is needed to identify and validate effective heuristic techniques for specific multi-robot problems and to assess the performance of various optimization algorithms. Table 20 outlines performance factors for MRTA problems, offering a framework for future studies.

- **Variables Influencing Complexity**: The complexity in MRTA problems increases with the number of robots, tasks, and the scale of the operational environment. More robots and tasks complicate the search for optimal solutions, especially when tasks are time-sensitive or spread across large areas.
- **Performance Metrics for Optimization Strategies**: Optimization strategies are evaluated based on their optimality, scalability, consistency, and fairness in task distribution. While achieving optimality is ideal, computational limits often require acceptance of sub-optimal solutions. Consistency ensures reliable performance, and fairness involves equitable task distribution according to each robot's capacity.
- **Numerical Complexity and Algorithm Performance**: As the size of the problem increases, so does the numerical complexity, challenging the scalability and efficiency of algorithms. It is crucial to examine whether the complexity of these algorithms is justified by the quality of their solutions.
- **Dynamic Task Allocation and Communication Challenges**: Adapting to dynamic environments that respond to real-time changes, unexpected tasks, and environmental

uncertainties is vital. The complexity of creating adaptable cost functions in such settings often leads researchers toward multi-objective optimization, which further complicates the solution process.

- **Communication Reliability:** Effective multi-robot operations heavily rely on reliable communication, which can be disrupted by external factors such as interference and network congestion. Innovative solutions are needed to mitigate these issues.

Table 20. Factors Influencing MRTA Problem Complexity

Performance Factors	**Market-based**	**Optimization-based**	**Behaviour-based**	**Task clustering-based**
No/Weak network	Broadcasting winner robot repetitively	Local network for proximity robots	Local network for proximity robots	Local network for proximity robots
Objective function	Single/multiple objective	Single/multiple objective	Single/multiple objective	Single/multiple objective
Coordination type	Centralized/distributed	Centralized/distributed	Centralized/distributed	Centralized/distributed
Reallocation task methods	Continuous auctioning and bidding	Continuous searching and allocation	Heuristics/ Bayesian Nash equilibrium	Difficult to reallocate within non-typical task clusters
Uncertainty handling techniques	Continuous auctioning and bidding	Difficult to manage uncertainties	Game theory, probabilistic, and/or predictive modelling	Difficult to manage uncertainty
Complex problem constraints	Difficult to execute auctions	Complex and difficult to solve due to multiple decision variables	Work in a hybrid manner	Work in a hybrid manner

Computational cost	Lower than optimization strategy	Higher than market-based strategy	Higher than optimization-based strategy	Lower than other methods

5.4.2 Overview of Research Trends and Directions

A comprehensive review of 71 articles, categorized by the Gerkey and Matarić MRTA taxonomy and application domains, reveals significant trends and research gaps in MRTA. Figure 107 shows that ST-SR configurations dominate the research, aligning with many practical applications. There's a growing interest in ST-MR-TA configurations, suggesting a shift towards exploring coalition formation among robots for complex tasks (Figure 108). This reflects the evolving focus of MRTA research towards enhancing real-world applicability and robustness.

In terms of optimization techniques, GA are the most commonly used, appearing in 32% of the studies (Figure 109). Clustering techniques and the HA each account for 16% of the research. While exact methods like BnB and HA offer optimal solutions under specific conditions, their practicality is limited by their inability to handle large or complex scenarios. Consequently, heuristic and meta-heuristic approaches like ACO, BCO, PSO, and SA are favored for their efficiency in delivering timely solutions, though they struggle with scale and real-time constraints.

Researchers often combine clustering with meta-heuristics to break down the MRTA problem into manageable parts: task assignment and task planning/scheduling. This approach, which assigns robots to clusters and then applies meta-heuristics for intra-cluster routing, reduces computational complexity but is unlikely to yield optimal results. For scheduling, local search methods or TSP algorithms like 2-opt are used to further simplify the problem.

The review also highlights the benefits of using a multi-robot system with individual depots for each robot to reduce travel distances and minimize inter-robot collisions. However, task allocation involving complex constraints like time-windows, hierarchical tasks, robot-dependent tasks, and scenarios with unknown tasks remains underexplored.

Future research is encouraged to develop strategies that enhance the dynamic adaptability, communication resilience, and overall efficiency of multi-robot operations in diverse and unpredictable environments as listed in Table 21. Exploring game-theoretical approaches for distributed task allocation in communication-compromised settings and integrating emerging technologies such as learning automata, deep learning, machine vision, and self-organizing map neural networks into dynamic task allocation strategies are potential directions.

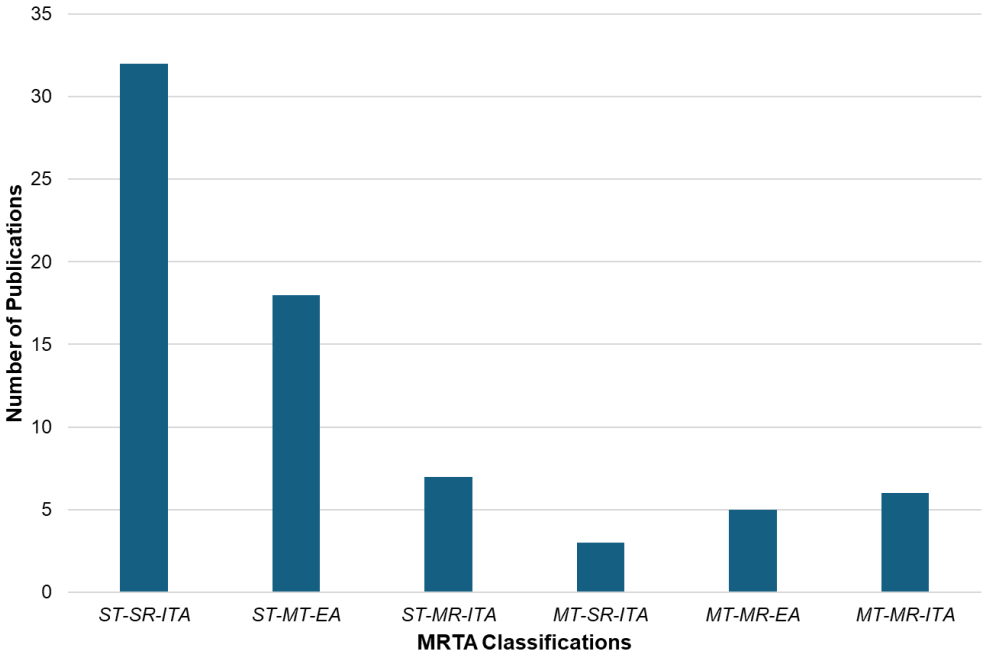


Figure 107. Number of Papers in Terms of MRTA Classifications

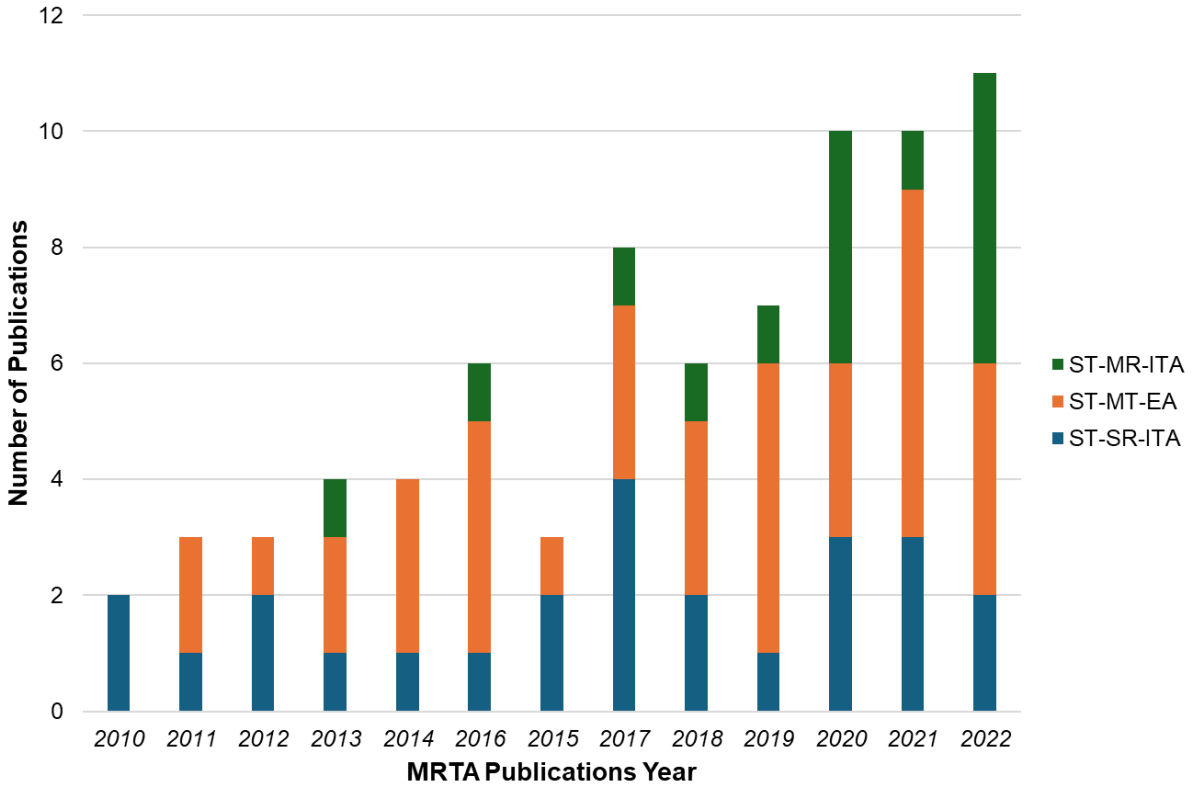


Figure 108. Number of Papers in Terms of Publication Years

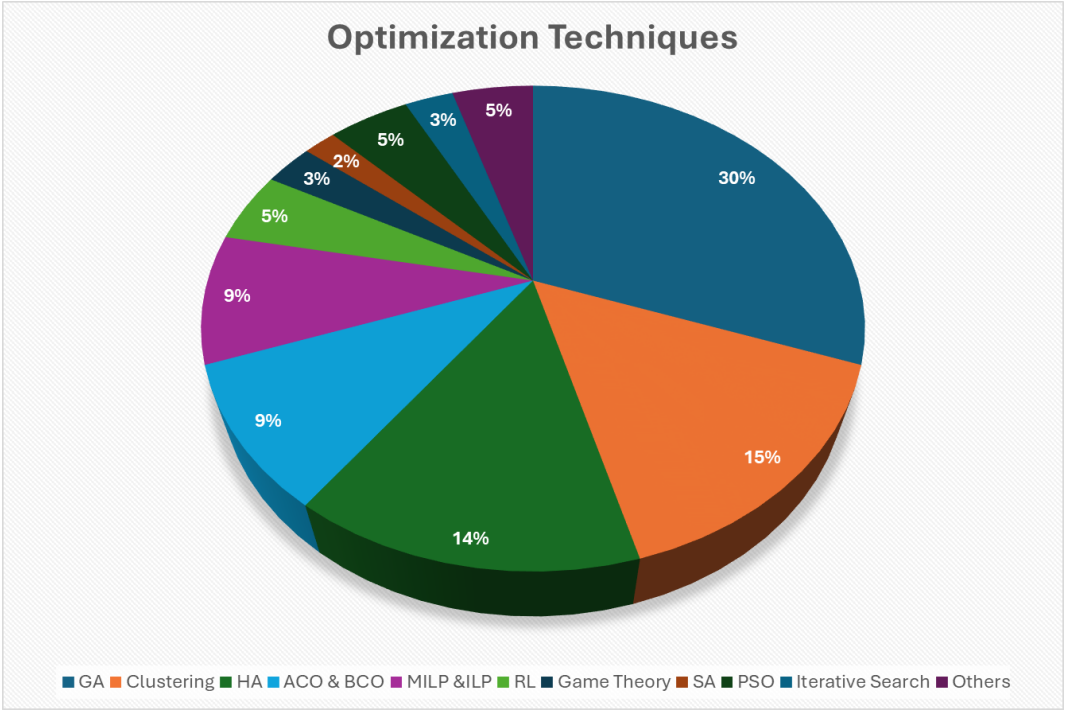


Figure 109. MRTA Optimization Techniques Allocation in Publications

Table 21. Future Research Directions

• Investigation and development of strategies for tightly coupled tasks	• Development of effective task rescheduling/reallocation mechanism
• Development of efficient failure-handling mechanisms	• Development of task allocation within a heterogeneous team
• Development of hybrid task allocation strategies	• Development of adaptive heuristic parameters for task allocation
• Development of energy-aware task allocation strategies for recharging robots to last longer	• Identification of ideal cluster size and procedure for task switching among task clusters
• Evaluation of task priorities	• Development of task allocation with no communication problem environment
• Coupling the performance evaluation metrics	• Development of uncertainty handling task allocation techniques
• Development of task allocation for strict time-constraint problems	• Real-time experimentation of task allocation problems

5.5 Conclusion

This research offers an in-depth review of MRTA problem, revealing a significant gap between simulation validations and real-world testing. It emphasizes the need for strategies that can handle weak or non-existent communication and improve robustness with efficient task switching mechanisms to manage uncertainties in dynamic environments.

Behavior-based dynamic task allocation techniques show promise in enhancing scalability and robustness but face challenges in environments where task allocation must be synchronized with dynamic path planning. The study underscores the necessity of developing robust strategies for these complex scenarios.

The review primarily discusses optimization-based approaches, noting a concentration on specific problem classes like ST-SR-TA, with GA being prevalent. However, it points out the numerical complexity of these algorithms and the lack of strategies accommodating real-time, dynamic task assignments. There's potential for improving outcomes by combining different approaches, addressing issues like collision avoidance, sensor malfunctions, and the kinematic constraints of robots which are often overlooked.

Future research directions include exploring systems with high DOF to utilize redundancy for collision avoidance, integrating time dimensions into task sequencing for dynamic environments like moving conveyors, and leveraging the ATSP to incorporate robot dynamics for better operational efficiency. The field also needs methods to handle sequence order constraints effectively and establish benchmarks for task sequencing that promote code reusability.

In summary, while existing research has developed efficient task allocation strategies, bridging the gap between theoretical models and practical applications remains crucial. Continued refinement of these strategies, integrating advanced technologies and robust algorithms, is essential for advancing industrial construction robotics in dynamic and complex environments.

Chapter 6 : Hybrid From Heuristics to Learning-Based Methods: Evolving 3D Motion Planning for Robotics

Abstract: 3D robot path and motion planning focus on creating optimal, collision-free routes within three-dimensional environments while adhering to kinematic constraints that include geometric, physical, and time-based limitations. This process is key in robotics, aiming to efficiently navigate through obstacles by prioritizing kinematic optimality and detailed environmental modelling without directly addressing dynamic forces. Effective movement strategies are essential for ensuring smooth, accurate trajectories, especially for robotic arms performing complex tasks. This research reviews foundational principles and recent advancements in robot 3D path planning algorithms, highlighting their broad applicability across various robotic platforms, including mobile and stationary industrial arms. While traditional methods like artificial potential fields, sampling-based approaches, and bio-inspired heuristics have addressed many motion planning challenges, they often struggle with computational intensity and slow convergence in dynamic, high-dimensional spaces, limiting real-time applications. Recent strides in learning-based motion planning methods have shown promise in navigating complex

environments more effectively. This review assesses learning-based strategies, including supervised, unsupervised, and reinforcement learning tailored for motion planning, which adapts through specific task-oriented rewards or learning from successful experiences. The research categorizes these methods derived from their exploration mechanisms and presents a new method for multi-fusion-based algorithms. This research provides a comprehensive examination of motion planning advancements for robotic manipulators, pinpointing current gaps, limitations, and future research opportunities. Algorithms are evaluated for their efficiency and practicality, providing a balanced view of their strengths and challenges. The study concludes that learning-based motion and path-planning methods are likely to remain at the forefront of research due to their effectiveness and reliance on specific knowledge and data sets. This ongoing evolution highlights the refinement and sophistication of robotic motion planning techniques, enhancing their capability to tackle complex tasks and environments with unprecedented precision and efficiency.

6.1 Introduction

AI significantly enhances robotics by enabling robots to collect, analyze, and use environmental data to make decisions, despite often dealing with unreliable information. Robots must autonomously perform actions to meet performance goals optimally while minimizing costs. Since environments can change unpredictably, robots adapt by replanning in real-time, considering errors, constraints, and feedback from sensors and users (Kala, 2016; Liu et al., 2015).

Autonomous robots often feature multiple interconnected parts, performing coordinated or independent movements (Halperin et al., 2017). A crucial aspect of these systems is motion planning, vital for enhancing robot functionality and decision-making capabilities. This review,

illustrated in Figure 110, explores motion planning's role within a hybrid AI-based pipeline, showcasing applications across various contexts (Tamizi et al., 2023; Mac et al., 2016).

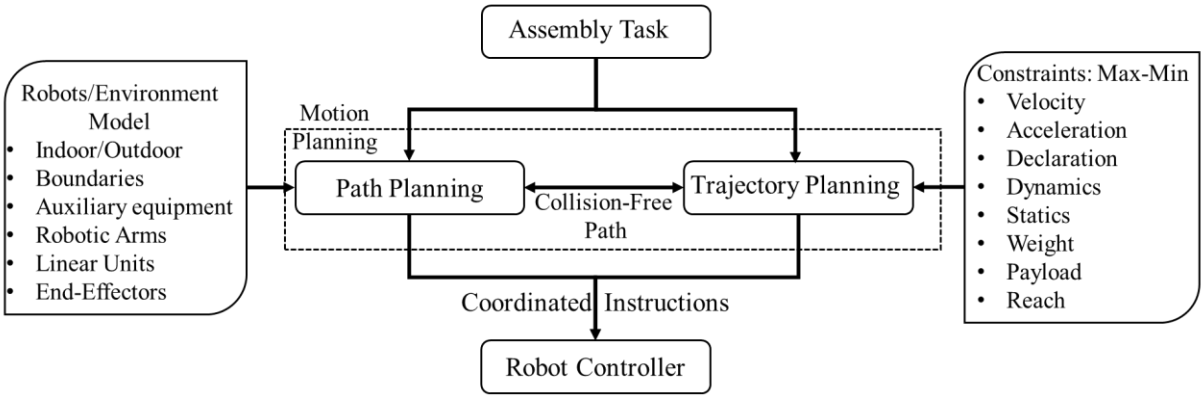


Figure 110. Hierarchy of a Robotic Motion Planning System

6.2 Literature Review

6.2.1 Path Planning vs. Trajectory Planning

Path and trajectory planning (illustrated in Figure 111) are distinct yet interconnected aspects of robotics. Path planning is concerned solely with creating a collision-free route based on environmental geometry, plotting a series of configurations from a start to a target point without considering the robot's motion dynamics. In contrast, trajectory planning adds the dimension of time, specifying when each configuration is reached, thereby integrating kinodynamic aspects such as joint velocities and accelerations.

Geometric constraints in path planning include avoiding obstacles and ensuring joint limits are not exceeded. Trajectory planning, however, also involves optimizing the physics of motion, such as velocities, accelerations, and the sequence of joint positions over time. The main objectives of trajectory planning are to minimize the robot's execution time, energy consumption, and jerk to improve efficiency, safety, and the smoothness of operations.

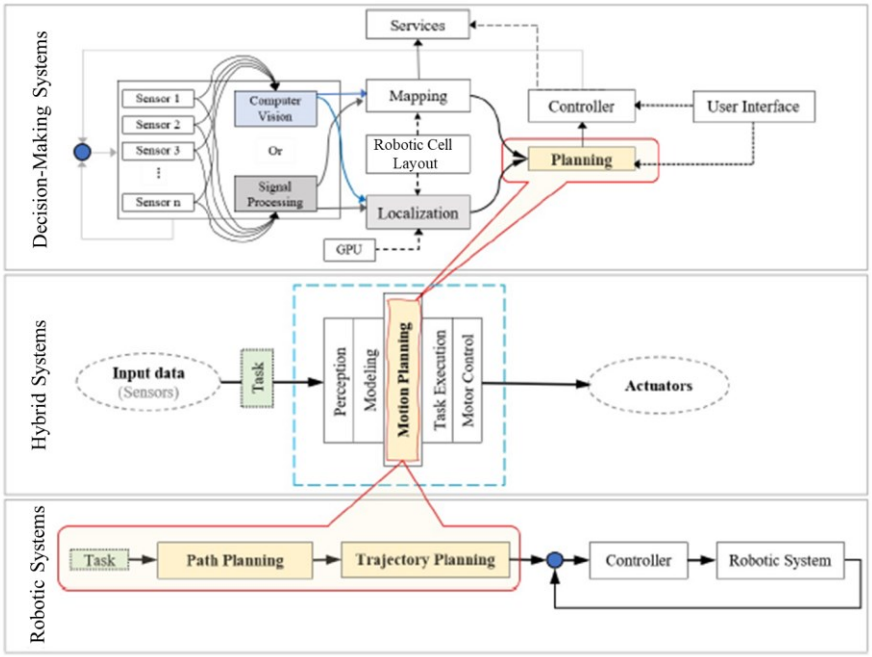


Figure 111. A Schematic Representation of the Position of Motion Planning

Motion planning is essential for autonomous systems navigating through 2D or 3D spaces filled with obstacles, using real-time data from sensors and image processing to update the robot's pose and adapt to dynamic environments. This process, complicated by the physical design of robotic manipulators, environmental uncertainties, and sensor reliability, relies on strategies like Dynamic Programming to avoid obstacles and ensure collision-free paths.

Path planning focuses on determining the most efficient routes based on environmental constraints and operates in the robot's joint or operational space, initially handling only kinematic aspects. This necessitates trajectory planning, which incorporates time to address dynamic factors like energy consumption, execution time, and acceleration impacts. Kinodynamic motion planning (KP) optimizes safety, smoothness, and energy efficiency, using algorithms that respect physical and temporal constraints to ensure feasibility.

However, integrating dynamics into motion planning is challenging due to the slow convergence of algorithms like Sampling-Based Motion Planning (SBMP), which aim for optimal but computationally intensive solutions. Motion planning algorithms are evaluated on their ability to navigate efficiently to a goal, emphasizing the need for robust local and global planning strategies to handle the complexities of dynamic environments as shown in Figure 112.

Path planning is crucial for deploying autonomous robots, guiding them from start to goal locations while avoiding obstacles. Key contributors like Choset (2005), LaValle (2001), and Latombe (1991) have extensively documented various algorithms, although detailed comparisons or focus on 3D path planning advancements are often lacking. A thorough examination of current 3D path planning methods is necessary.

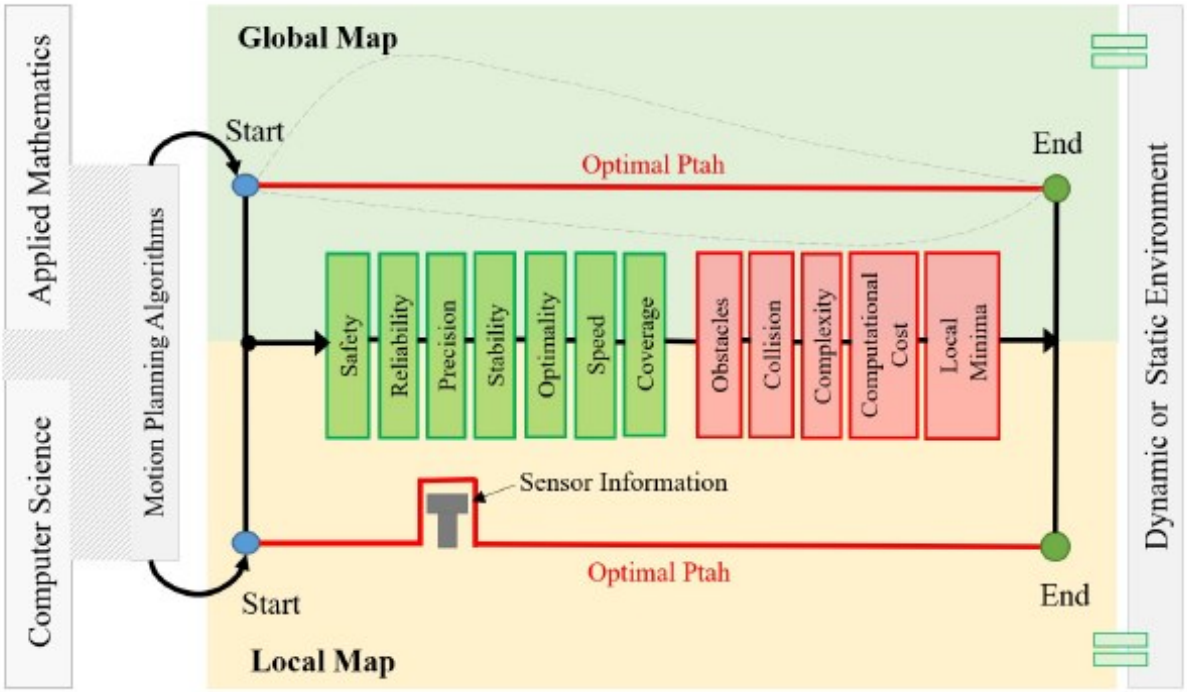


Figure 112. Criteria for Motion Planning Algorithms

In industrial settings, effective motion planning is essential for operational success. Traditional algorithms like A* and Rapidly Exploring Random Tree (RRT) address many challenges but face limitations such as scalability and sensitivity to sampling distributions. Emerging learning-based approaches, utilizing techniques from supervised to reinforcement learning, show promise in overcoming these limitations.

This research explores both traditional and learning-based motion planning, focusing on their applications in navigating robotic arms through complex tasks as shown in Figure 113. The study categorizes these methods, assessing their strengths and areas for improvement, and aims to optimize planning and execution processes in robotics, blending path and trajectory planning in practical applications.

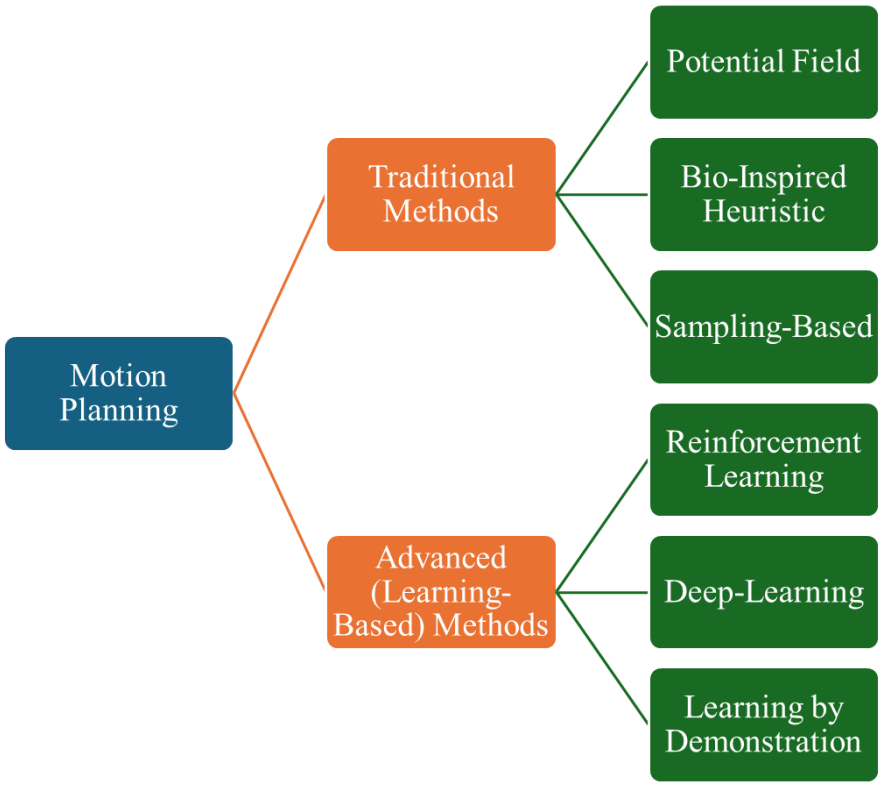


Figure 113. Different Types of Motion Planning Methods

Recent reviews indicate a focus on two-dimensional (2D) path planning, with limited exploration of three-dimensional (3D) strategies, treating elevation as static. Choset (2005) provided a thorough overview of path planning, focusing mainly on 2D, while LaValle et al. (2006) discussed sampling-based algorithms without extensive coverage of 3D planning. Galceran and Carreras (2013) contributed significantly by categorizing decomposition methods, enhancing understanding of 3D path planning but still focusing on sampling-based techniques. This highlights a need for more comprehensive research on 3D path planning to unlock the full potential of robotic navigation in complex environments. The literature suggests a growing interest in expanding 3D path planning research, as demonstrated in various studies and illustrated by generic obstacle avoidance procedures in Figure 114.

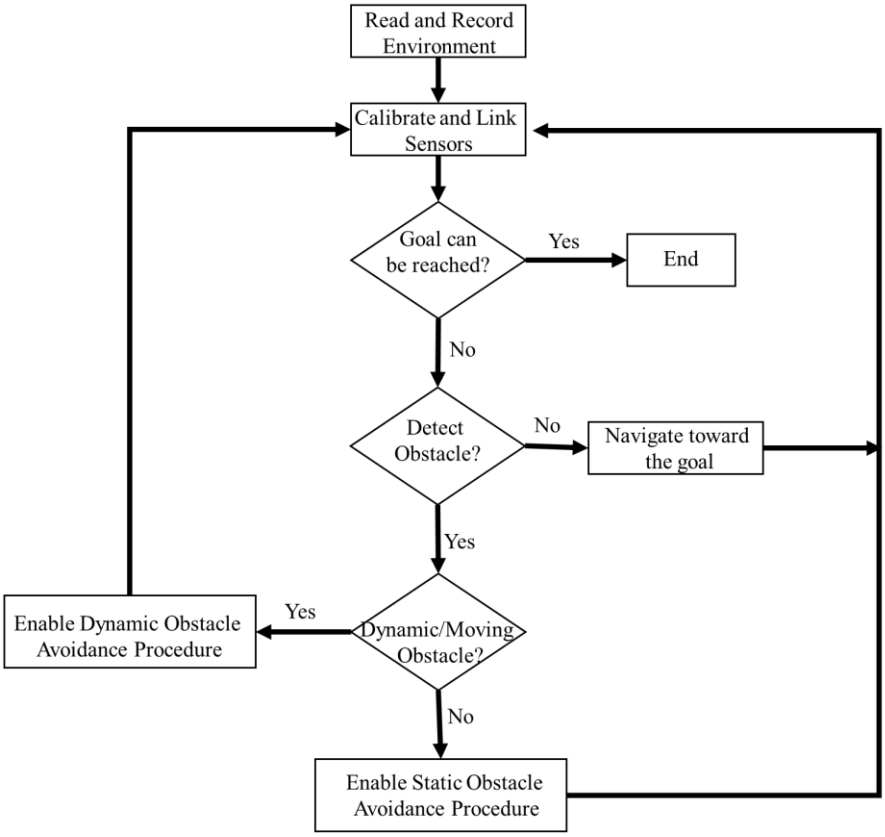


Figure 114. Generic Obstacle Avoidance Procedure

As robots navigate increasingly complex and unstructured environments, the need for advanced 3D path-planning algorithms has become critical, with 2D strategies proving insufficient. 3D path planning, an NP-hard problem, faces significant challenges from kinematic constraints and the need for precise environmental modeling. Key algorithms in this area include visibility graphs, which connect visible vertices of objects, and stochastic methods such as RRT and Probabilistic Road Map (PRM), alongside optimal search strategies like Dijkstra's, A*, and D* algorithms. The review highlights a two-step approach to 3D path planning: initial environment perception and modeling, followed by the application of path planning algorithms aimed at optimizing time and cost. This structured yet flexible methodology is discussed with an emphasis on efficiency, complexity, and adaptability to uncertain conditions. The review also introduces a new category in the taxonomy of 3D path planning algorithms called multifusion-based algorithms, reflecting the ongoing evolution in this field.

6.2.2 Fundamental and Properties of Motion Planning

This section explains the basic principles and essential ideas related to motion planning. A key concept introduced early in the study of motion planning is the configuration space (*C*-space), which serves as a foundational framework. *C*-space allows for the representation of every potential position and orientation of a robot as a distinct point within this space. To illustrate, consider a robotic arm equipped with *n* joints; its specific arrangement can be formulated by a point in *C*-space, as demonstrated in Equation 1.

$$q = [\theta_1 + \theta_2 + \cdots + \theta_n] \quad (1)$$

In this context, i represents the position of the i^{th} joint in the robotic arm. If this particular configuration exists within an area surrounded by obstacles and adheres to the constraints of the joints, it is positioned within the obstacle-free configuration space (C-free). Within the field of motion planning, both the initial configuration, q_{init}, and the desired target configuration, q_{targ}, are considered to be within the bounds of C-free.

The fundamental motion planning problem is formulated as follows, based on the given parameters:

- C: the configuration space,
- C_{obs}: the obstacle region within the configuration space,
- $C_{free} = C \setminus C_{obs}$: the obstacle-free region of the configuration space,
- $q(x_{init})$: the configuration q corresponding to the initial state x,
- $q(x_{goal})$: the configuration q corresponding to the goal state x.

The objective is to determine a duration T and a sequence of controls $u: [0, T] \rightarrow U$ such that $x(T) = x_{goal}$ and $q(x(t)) \in C_{free}$ for all $t \in [0, T]$. When expressed in integral notation, the formulation provides a comprehensive mathematical model for achieving a collision-free trajectory from the initial point to the target point within the specified time frame.

6.2.3 Motion Planning Properties

A motion planner is considered complete when it consistently finds a solution, if one exists, within a finite timeframe. Two main types of planners are resolution-complete and probabilistically complete. Resolution-complete planners use a discretized model of the configuration space (C-

space) to map out paths, whereas probabilistically complete planners increase the likelihood of finding a solution as more time is allowed for the computation.

Furthermore, motion planning is divided into single-query and multiple-query strategies. Multiple-query planners create a comprehensive model of C-space in advance, which is beneficial for static environments as it allows for quick trajectory generation. Conversely, single-query planners are suited to dynamic environments since they address each planning problem afresh, adapting to changes in C-space effectively. This makes them well-suited to situations where the environment frequently changes.

6.2.3.1 Collision Detection

In robotics, the configuration space (C-space) is divided into C-free, the area free of obstacles, and C-obstacles, areas occupied by obstacles. Efficient collision detection is crucial in motion planning, with distance measurement algorithms such as the Gilbert–Johnson–Keerthi (GJK) algorithm calculating the minimum distance between convex shapes for collision detection. Simpler methods use spheres to approximate objects, speeding up the process.

Collision detection can dominate up to 90% of motion planning time, prompting the integration of machine learning to improve efficiency. Support Vector Machines (SVM) and Gaussian Mixture Models (GMM) have enhanced collision boundary precision and speed in path generation, respectively. The K-nearest neighbor (KNN) technique has also been applied for static environment collision detection, while a recent innovation, Fastron, uses machine learning to segment C-space and efficiently predict collisions. This method requires a pre-segmented configuration space, assuming some simplifications are made by the operator.

6.2.3.2 Preliminary Concepts

Robots are engineered to operate autonomously, requiring minimal human supervision. Essential to their autonomy is path planning, a process that varies in definition but is generally described by Karaman and Frazzoli (2011) in a more standardized form. Robots typically navigate in a three-dimensional space (S^3), often called the robotic workspace *rw* which may contain hard and soft obstacles, denoted as rw_{oi} for the i^{th} obstacle. The area is clear of obstacles, $rw_{free} = rw \cup i\ rw_{oi}$, constitutes the safe zone for robot navigation. The starting point, x_{init}, and the destination, x_{goal}, both lie within rw_{free}, framing the path planning challenge as a triplet $(x_{init}, x_{goal}, w_{free})$.

Concept 1 (Path Planning): Path $\delta: [0, T] \rightarrow S^3$, starting at $x_{init}(\delta(0) = x_{init})$ and ending at $x_{goal}(\delta(T) = x_{goal})$, embodies path planning if there exists a continuous, uninterrupted process Φ that ensure $\delta(\tau) \in rw_{free})$ for every $\delta(\tau) \in rw_{free})$ for every $\tau \in [0, T]$.

Concept 2 (Optimized Path Planning): For a given path planning problem $(x_{init}, x_{goal}, w_{free})$ and a cost function $C: \sum \rightarrow S \geq 0$ (with $\sum$ representing all possible paths), if a path δ' satisfies the conditions of Definition 1 and minimizes the cost function $C'(\delta) = \min\{ C(\delta) | \delta \in \sum\}$, then $\delta'$is deemed the optimal path, and the process Φ' achieving this is considered the optimal path planning.

Concept 3 (Path Planning): Path planning involves identifying a continuous (though not necessarily smooth) curve within the configuration space that connects the starting point, x_{init}, to the destination, x_{goal}. This curve should meet the following criteria: (a) it is developed without considering the element of time, (b) it permits pauses at specific positions, and (c) it consists of various segments, each potentially representing a trajectory on its own.

Concept 4 (Trajectory Planning): Trajectory planning addresses navigating a path determined by a robot's path planning algorithm. A trajectory is a series of states indexed by time, often mathematically represented by a polynomial *X(t)*, with velocities and accelerations derivable through time differentiation. This process accounts for kinodynamic constraints, making them an integral part of trajectory planning.

Concept 5: Trajectory planning is characterized by time-continuous outputs and must adhere to control limitations, forming an integral component of the overarching path.

6.3 Motion Planning Pipelines: 3D Path Planning Algorithm Taxonomy

Motion planning, a key component of the broader Task and Motion Planning (TMP) challenge, involves handling conditions of partial observability. This process typically includes optimizing symbolic paths using a decision tree that expands with incoming data, and then refining the best policy via an optimized joint trajectory tree (Phiquepal et al., 2019; Dantam et al., 2018).

The field of 3D path planning has significantly evolved, introducing numerous algorithms for varying robots and environments. This research reviews and categorizes these algorithms, such as RRT, PRM, APF, and MIP. A proposed classification, shown in Figure 115, systematically classifies these strategies into five distinct groups, such as sampling-based algorithms which utilize Monte Carlo sampling to establish environmental connections. The unique characteristics and methodologies of each category are discussed in detail, providing a comprehensive overview of 3D path planning strategies.

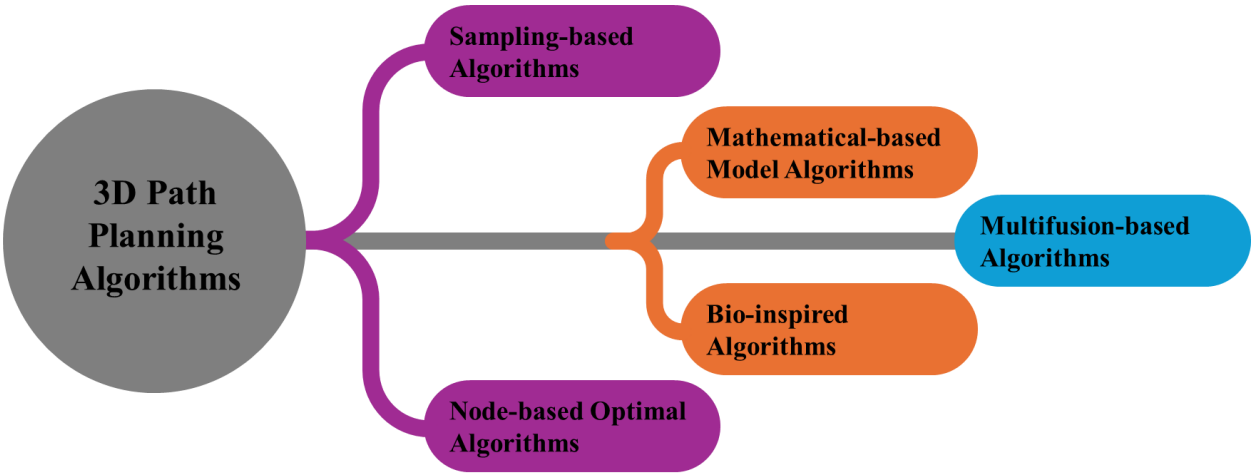


Figure 115. 3D Path Planning Taxonomy

6.3.1 Sampling-Based Algorithms (SBA)

SBA methodologies necessitate a pre-mapped mathematical representation of the workspace that includes obstacles and navigable areas. These methods typically dissect the environment into nodes or cells, mapping it comprehensively or using stochastic searches to pinpoint feasible paths. Examples include RRT and PRM, detailed in Figure 116, which sample the environment to explore paths. The review categorizes these algorithms into active and passive groups. Active algorithms like RRT and Dynamic Domain RRT (DDRRT) autonomously seek viable paths to the goal, whereas passive algorithms such as PRM generate a network of potential paths, requiring further analysis to select the optimal route. Additionally, 3D Voronoi diagrams and APF methods fall under SBA, using complete environmental data to avoid obstacles and manage local minima.

In the hierarchical structure of motion planning, SBA acts as an intermediary that delivers a viable, collision-free path after receiving start and goal configurations, shown in Figure 117. Positioned as a "black box" in this framework, it connects a high-level behavioral planner that defines goals with a low-level controller that executes the path. The function of sampling-based tree planners is further clarified in Algorithm 1.

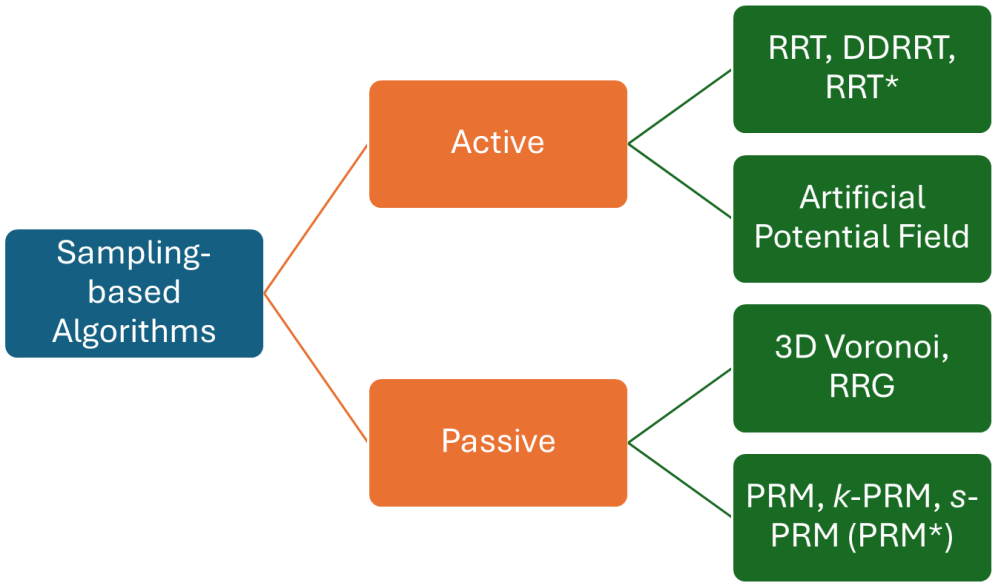


Figure 116. Sampling-Based Algorithms

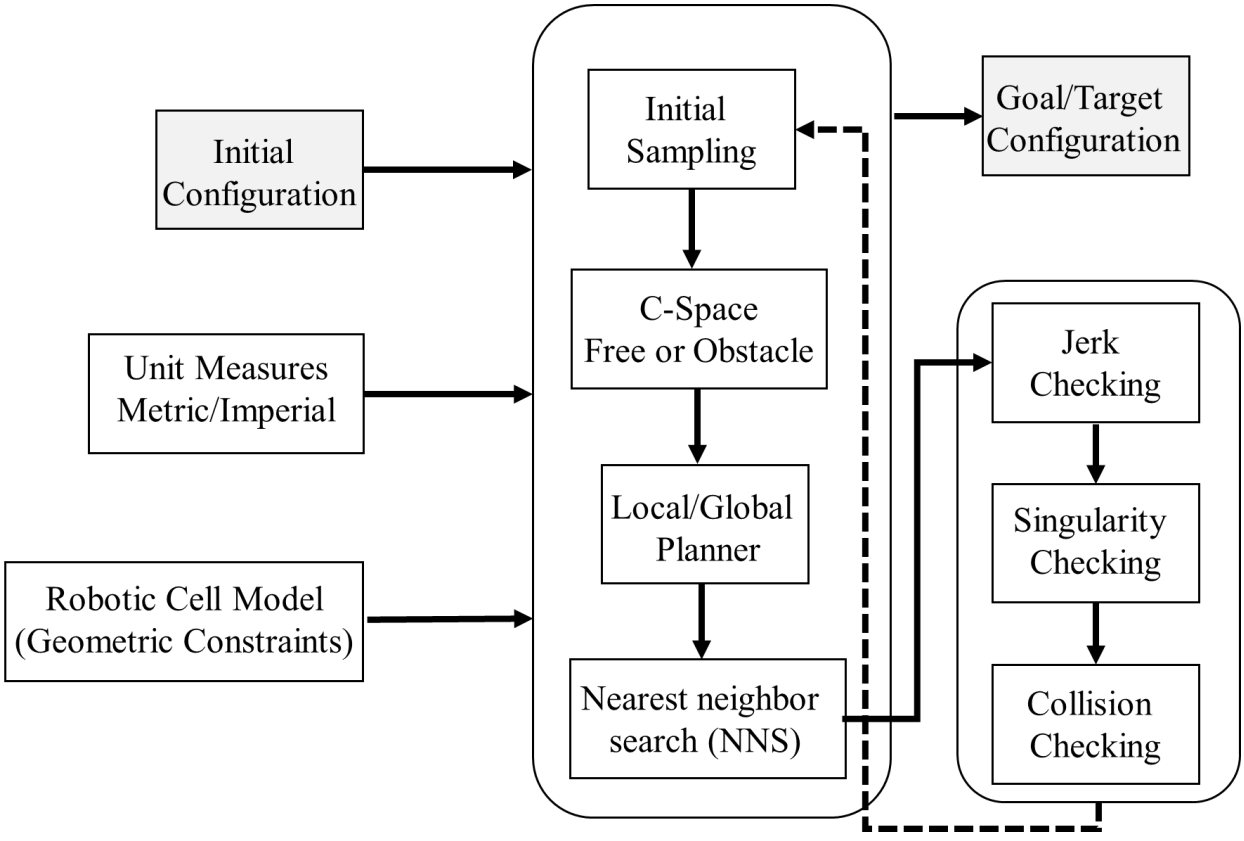


Figure 117. Generic Sampling-Based Planner

Algorithm 1: Sampling-Based Tree Planner

---

input ← path planning triplet $(x_{init}, x_{goal}, w_{free})$;

output ← constructed search tree (τ);

τ ← init.tree (x_{init});

while not stopping.criteria (τ, x_{goal}) **do**:

    x_{sel} ← select.node (τ);

    x_{new} ← extend.node (τ, x_{sel});

    If $x_{new} \neq$ null **then**:

        Update.tree (τ, x_{sel}, x_{new});

end

OUT ← $\{\tau\}$

6.3.1.1 Rapidly Exploring Random Trees (RRT)

introduced by LaValle in 1998, RRT is designed for path planning under various constraints such as holonomic, nonholonomic, and kinodynamic, making it suitable for complex multi-DOF systems. RRT is widely used in robotics, mainly in robots like the PR2. To improve its efficiency, Yershova et al. (2005) developed the DDRRT, which speeds up escaping local minima. Further enhancements by Karaman and Frazzoli (2010) introduced the Rapidly-exploring Random Graph (RRG) and RRT*, optimizing RRT's performance in achieving more optimal outcomes.

RRT effectively navigates the configuration space by generating a trajectory from the start to the goal. The process involves iteratively sampling new nodes and integrating them into the path if they successfully connect to the nearest existing node. In a 3D context, the method necessitates a 3D configuration space $X = C$, divided into an obstacle-loaded region $X_{obs} \subset X$ to be avoided and a navigable area $X_{free} \subset X$. The exploration by RRT populates a set *P* of path states or vertices.

The algorithm's implementation unfolds as follows:

(1) Initialization: Begin by placing the initial state $x_{init} \in X_{free}$ into P as the starting vertex. Then, sample a random state $x_{random} \in X_{free}$, demonstrating this step with $x_{random1}$ and $x_{random2}$.

(2) Nearest Neighbor: Identify the nearest state x_{near} to x_{random} within P, using a predefined metric (usually Euclidean), and consider x_{near} as $x_{random's}$ parent state.

(3) Control and Expansion: Given that x_{random} may exceed the robot's capacity, apply a control input under kinodynamic considerations within a cost function $\varphi = f(x, y, z)$. Evaluate the reachable state x_{new}, ensuring it falls within X_{free}. If so, add x_{new} to P; otherwise, disregard it. This step includes the decision to omit x_{new1} and repeat the process as shown in Figure 118.

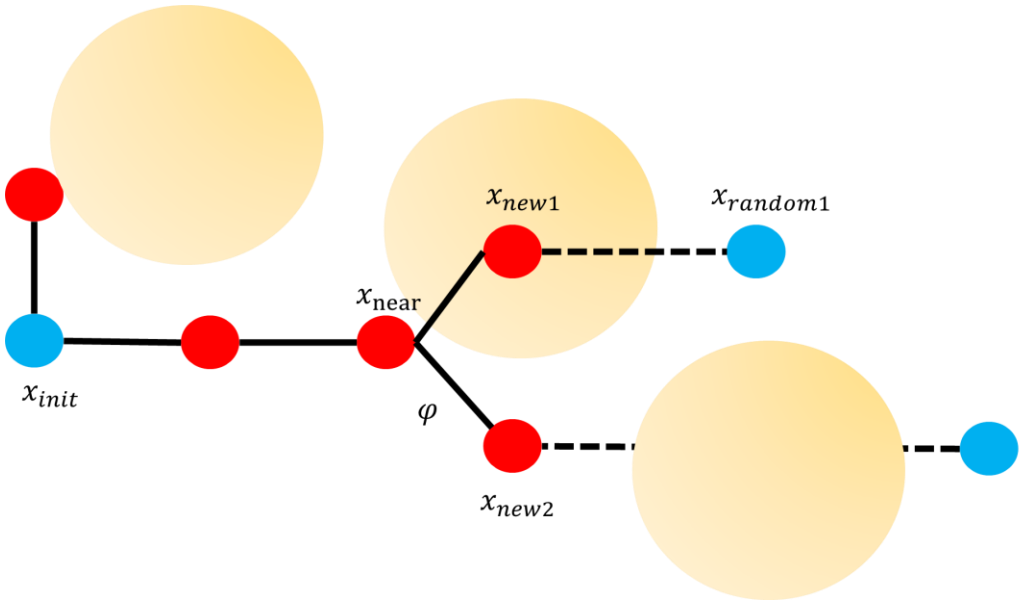


Figure 118. RRT Algorithm Procedure

Despite RRT's ability to chart a path to the goal, its reliance on Monte Carlo sampling tends to bias exploration towards already surveyed regions, leading to inefficiencies, especially in cluttered environments, and challenges in achieving optimality.

6.3.1.2 Dynamic Domain RRT (DDRT)

RRT method was improved by the development of DDRRT, which struggles with obstacle-rich areas that create local traps in the configuration space. RRT's limited use of local environmental data often leads to inefficient sampling and decreased time efficiency. Figure 120 visually compares these methods, highlighting DDRRT's enhanced ability to navigate around obstacles, indicated by the black arrow showing potential expansion directions.

The primary distinction between the two methods lies in the initialization prcoess. DDRRT is governed by an *n*-dimensional sphere, matching the environment's geometry, centred at x_{near} with a specific radius, *r*, to denote the reachable area. In this modified approach, a new sampling node x_{random} is selected if its distance from the nearest node x_{near} falls within radius *r*; otherwise, it is disregarded. The process attempts to connect x_{near} and x_{random}, adjusting radius *r* to infinity if the connection succeeds, or reverting it to *R*, upon failure.

DDRRT effectively mitigates the Voronoi bias issue prevalent in traditional RRT strategies, promoting quicker exploration across the configuration space. Nevertheless, similar to RRT, DDRRT lacks a mechanism for post-processing to smooth the resulting path, meaning the paths produced, while more efficiently reached, do not inherently approach optimality.

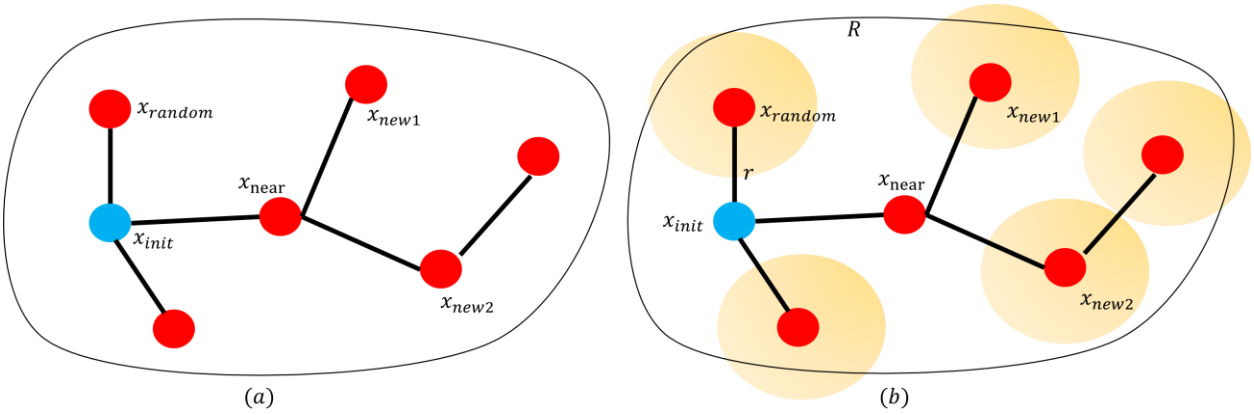


Figure 119. Enhanced DDRRT's

6.3.1.3 Rapidly Exploring Random Graph (RRG)

While the RRT method is effective in practical applications and guarantees success, it often overlooks the quality of its output, leading to non-asymptotically optimal solutions. To address this shortfall and achieve asymptotic optimality, the concept of the RRG was introduced. This approach utilizes a structure $k = (S, S_0, T, L)$ to depict the system's performance, where S represents the set of states, $S_0 \subseteq S$ the initial state set, $T \subseteq S \times S$ the transition relation, and L the labeling function that associates each state with a set of atomic propositions.

RRG method extends to all vertices identified by a specific function and interconnects them, creating a complex map that is highly likely to converge to an optimal path over time. However, like the PRM, RRG forms a dense network and cannot independently determine the most efficient path, necessitating additional algorithms to identify the optimal route.

6.3.1.4 RRT-Star (RRT*)

Karaman and Frazzoli (2005) introduced the RRT* algorithm as an extension of the RRG, maintaining the asymptotic optimality characteristic of RRG while specifically addressing differential constraints. RRT* enhances path efficiency by eliminating suboptimal connections in

a refinement process, thereby reducing the overall path cost. It incorporates a cost function, *cost (p)*, to quantify the expense of the path from x_{init} to any state *p* within the set P, setting the initial cost from x_{init} to zero.

Distinct from RRG, RRT* includes a post-processing phase that improves the solution quality, as described in its pseudocode presented in Algorithm 2.

Algorithm 2: RRT*

Create(x,y), Near$(G, x_{random}, |P|)$, ObstaFree(x,y);

Parent(x_{near}) returns the parent vertex of x_{near};

$P' = P; E' = E$

$x_{near} \leftarrow$ Near(G, x)

$x_{new} \leftarrow$ Create(x_{near}, x)

if ObstaFree(x_{near}, x_{new}):

    $P' = P' \cup x_{near}$;

    $E' = E' \cup \{(x_{new}, x_{near}), (x_{near}, x_{new})\}$;

    $X_{near} \leftarrow$ Near $(G, x_{new}, |P|)$;

    while all $x_{new} \in X_{near}$:

    if ObstaFree (x_{near}, x_{new}) and

    cost(x_{new}) > cost(x_{near}) +$c(Dist(x_{new}, x_{near}))$

        $x_{min} = x_{new}$;

        End;

    End;

$E' = E' \cup \{x_{new}, x_{near}\}$;

while all $x_{new} \in X_{near} \backslash \{x_{min}\}$;

if ObstaFree(x_{min}, x_{new}) and

cost(x_{near}) > cost(x_{new}) +$c(Dist(x_{new}, x_{near}))$;

    $x_{parent} \leftarrow$ Parent(x_{near});

    $E' = E' \backslash \{x_{parent}, x_{near}\}$;

    $E' = E' \cup \{x_{new}, x_{near}\}$;

    End;

Return $G' = (P', E')$

RRT* initially identifies the closest state alongside neighboring states X_{near}, incorporating the nearest state into the tree and the least costly connection. Subsequently, it aims to eliminate connections in favor of more efficient ones via the X_{new} state with relative to the cost. However, this method increases overall computation time and is not designed to produce multiple paths simultaneously.

Choudhury et al. (2013) enhanced the RRT and RRT* methodologies by selecting an alternative, second-best parent node when the nearest parent node is densely populated with child nodes. This adjustment allows for quicker online replanning, making the path adjustments more dynamic compared to the static nature of traditional RRT and RRT* approaches.

6.3.1.5 Probabilistic RoadMap (PRM) Series

Introduced by Kavraki et al. (1996), the PRM is a multiple-query path planning approach that constructs a roadmap through random sampling within the configuration space X, particularly focusing on the collision-free area, X_free. PRM connects sampled states using one of several strategies: connecting to the nearest k neighbors (K-PRM), connecting within a specified radius (δ-ball), or a combination of both to balance computational efficiency and collision avoidance. Once the roadmap is established, search algorithms like Dijkstra or A* are utilized to search and find the least-cost path from start to target.

Karaman and Frazzoli (2011) refined PRM into three variants to optimize vertex selection and connection strategies. K-PRM adjusts the k parameter to improve path smoothness but may bias towards denser areas. S-PRM varies the connection radius to address K-PRM's density bias, though it increases computational load. KS-PRM combines the advantages of both, enhancing directional connections and computational efficiency. PRM's efficiency in 3D environments was initially

demonstrated by Kavraki and Latombe (1994), but the method's drawback is the increased computational burden of collision checking as the exploration graph expands.

6.3.1.6 Voronoi

Shamos and Hoey (1975) introduced the Voronoi diagram to computational geometry, initially for sets of points in the Euclidean plane. This method has evolved to significantly aid path planning, creating uniform topological connections from edges to nearby obstacles. Luchnikov et al. (1999) extended its application to 3D systems, enhancing the method for complex spatial challenges. In 3D path planning, Voronoi diagrams start by establishing an initial point equidistant from surrounding obstacles, expanding through iterative analysis of Voronoi channels which define network boundaries.

The selection process within these diagrams involves calculating channels for object trios, using a radial edge data structure to manage the geometric and topological features of the diagram. While Voronoi diagrams facilitate the creation of both global and local graphs, they require supplementary algorithms like Dijkstra's, A*, or D* to optimize and identify the shortest path. The method unfolds in three stages: environmental sampling or structuring, constructing a 3D Voronoi graph, and deploying a search algorithm to determine the least costly path.

6.3.1.7 Artificial Potential Algorithms (APF)

Khatib (1990) introduced the potential field method, gaining widespread use due to its minimal computational requirements. This method assigns potential functions to differentiate between free space and obstacles, treating the robot like a particle influenced by attraction to goals and repulsion from obstacles, navigating along the resultant force gradient. However, this method can trap robots in local minima, leading to challenges in practical applications.

To counteract this, enhancements have been developed. Connolly et al. (1990) proposed a harmonic potential approach using Laplace's Equation to avoid local minima. Rimon and Koditschek (1992) introduced a navigation method based on a Morse function, which effectively avoids local minima by ensuring a single minimum exists at the goal. Additionally, Sundar and Shiller (1994) applied the Hamilton-Jacobi-Bellman principle to create an HJB function, further aiding in navigating away from local minima and refining the approach for more effective real-world application.

6.3.1.8 Summary of SBA

Sampling-based algorithms rely on initial estimates and may use methods like proximity collision detection or potential field adjustments to reduce dependence on detailed environmental models, enhancing adaptability across various settings. However, these adaptations can affect the algorithms' completeness and clarity in environmental mapping. Table 22 provides a comprehensive analysis of these algorithms' strengths and limitations.

Additional algorithms like the visibility graph (Flemming et al., 2011) and the corridor map method (Geraerts, 2010) also fall under sampling-based strategies. The visibility graph simplifies obstacle-based PRM by Amato et al. (1998), using cell decomposition similar to the octree-structured PRM by Yan et al. (2013). Both require a node-search algorithm to determine the optimal path, reflecting PRM and Voronoi diagrams' methodologies.

Table 22. Summary of SBA

SBA Type	Deficiencies	Improvements	Advantages
RRT	Only a single path is produced, and the solutions	Various enhancements have	It has a low time complexity and can

	are not optimal. It also considers only static threats.	been proposed to address these issues.	quickly search through space.
PRM	Like RRT, PRM only addresses static threats and does not guarantee an optimal path. Additionally, collision checking can be computationally expensive.	Some strategies have been introduced to make collision checking more efficient.	It is well-suited for navigating complex environments and is beneficial for scenarios that require re-planning.
Voronoi	The method might not represent the entire space completely and can sometimes fail to converge to a solution. It also addresses only static threats.	There are suggested methods for ensuring better convergence and completeness.	Implementation is straightforward for online use, and it can operate without intensive collision checking.
Artificial Potential	The approach can get stuck in local minima, failing to find the global minimum.	Multiple improvements have been proposed to help escape local minima and reach global solutions more reliably.	This method converges quickly to a solution.

6.3.2 Node Based Optimal Algorithms

This group of algorithms utilize weight information from nodes and arcs to calculate costs and find the optimal path through a pre-constructed network. These algorithms, often referred to as network

algorithms, rely on a graph composed of nodes or cells, leveraging pre-processed information to traverse the network efficiently and achieve optimal paths within their specific decomposition constraints. Figure 120 illustrates the core elements of these algorithms, including Lifelong Planning A* (LPA*) by Koenig and Likhachev (2002), which updates the A* algorithm to adapt to environmental changes, and D*-Lite (Koenig and Likhachev, 2005), tailored for environments with dynamic threats. Algorithms like Dijkstra's, A*, and D* fall under several categories like discrete optimal planning, roadmap algorithms, or search algorithms, reflecting their shared foundation in discrete optimization through grid decomposition.

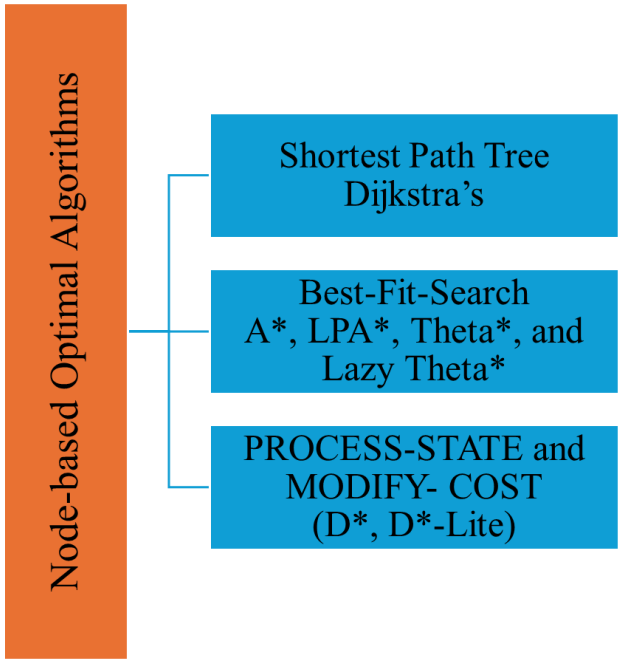


Figure 120. Node-Based Algorithms

6.3.2.1 Dijkstra's Algorithm

The shortest path in a weighted graph is calculated using a breadth-first search approach and dynamic programming principles. It requires constructing a 3D weighted graph for three-dimensional applications and performing an exhaustive search to find the minimum cumulative cost path. The algorithm, including its pseudocode, is described in Algorithm 3. Verscheure et al.

(2010) enhanced it by adding a constraint effective in navigating tubular structures, a concept initially proposed by Wan et al. (2002). Musliman et al. (2008) validated its practicality in a 3D GIS environment through experiments, confirming its effectiveness. The algorithm's performance heavily depends on the choice of the priority queue data structure, which significantly affects the time needed for computation.

6.3.2.2 A-Star (A*)

The A* algorithm, introduced by Hart et al. (1968), extends Dijkstra's algorithm by integrating a heuristic to reduce explored states. This heuristic accurately predicts costs to the goal, enabling faster and optimal convergence through a directed approach. A* is particularly effective in 3D environments, as shown by Amato et al. (1998) using the PRM method and by Niu and Zhuo (2008) with their "cell" and "region" concepts for better spatial representation. Further adaptations include LPA* by Koenig and Likhachev (2002), which dynamically adjusts to environmental changes, and conflict-direct A* by Williams and Ragno (2007), which excludes conflicting subspaces to expedite search.

Theta*, developed by Nash et al. (2007), and its comparison with A* by De Filippis et al. (2012), offers smoother pathfinding and reduced search efforts in 3D spaces. However, Theta* demands extensive time to evaluate unexpected neighbors, which led to the development of lazy Theta* that skips these evaluations with a line-of-sight check. Likhachev and Ferguson (2009) proposed reusing prior data and updating search details for faster responses to dynamic changes, although this increases computational requirements.

Algorithm 3: Dijkstra's algorithm

Q is the priority queue, x_{goal} is the target/goal;

I(x,u) return the cost to apply action u from x;

C' is the best cost-to-known so far;

U_x is the finite action space;

$x' = f(x,u) \in X$ is a state transition function;

$C(x) = \infty | (x \neq x_{init}); C(x_{init}) = 0;$

$Q.Insert(x_I);$

while Q not empty

$x \leftarrow$ Q. GetFirst();

End;

for all $u \in U(x)$;

$x' \leftarrow f(x,u)$;

if C(x) + I(x,u)<min$\{C(x'), C(x_{goal})\}$;

C'(x) = $C(x)$+ I(x,u);

End;

if $x' \neq x_{goal}$;

Q.Insert(x');

End;

End;

6.2.3.3 D-Star ($D*$)

D*, short for dynamic A*, was highlighted in DARPA's unmanned ground vehicle projects for its real-time navigation capabilities in dynamic environments (Stentz, 1995). It adjusts edge weights on-the-fly to avoid obstacles, calculating the shortest path from the robot's current position to the target. Similar to D*, it combines retrospective analysis and forward estimation to update arc cost functions, allowing it to adapt to environmental changes and find the most efficient route.

Smirnov (1997) determined the performance bounds of D*, addressing its potential for calculating unrealistic travel distances and the notable discrepancy between these bounds. Tovey et al. (2003) refined these bounds to more accurately reflect D*'s capabilities. Koenig and Likhachev (2005) further developed D* Lite, an adaptation of D* for changing goal locations, offering a more efficient approach for dynamic scenarios.

6.2.3.4 Summary of Node-Based Optimal Algorithms

Table 23 describes node-based optimal algorithms and their limitations due to the partial views of nodes and arcs in the configuration space, confining their effectiveness and rendering them suitable only for single-path searches, not for managing multiple paths for a fleet.

Regarding computational requirements, Dijkstra's algorithm has a complexity of $O(N^2)$, with n representing the total count of nodes. However, A* and D* algorithms have been optimized to reduce computational complexity, enhancing their suitability for real-time applications.

Table 23. Summary of node-based optimal algorithms

Node-Based Optimal Algorithm Type	**Deficiencies**	**Improvements**	**Advantages**
A*	Can have a significant computational load and sometimes produces paths that are not smooth. It also primarily considers static threats.	There have been advances aimed at reducing the computational requirements and improving the smoothness of the paths generated.	Offers a fast search capability and is suitable for real-time implementation.
D*	May estimate distances in a way that's not realistic for certain applications.	New versions improve the reality of distance estimations.	It searches quickly and is adept at handling dynamic

			changes within the environment.
Dijkstra	Suffers from high time complexity and only accounts for static threats.	Enhancements have been made to reduce the time complexity and extend its application beyond static scenarios.	It is straightforward to implement and versatile across various environments.

6.3.3 Mathematic Model Based Algorithms

This group of algorithms simplify robots to point models in grids, focusing on basic physical constraints but not capturing the full environmental or dynamic complexities. In contrast, mathematical model-based algorithms optimize by including kinodynamic constraints through polynomial representations, modeling environments as time-varying systems related to the robot's location.

These algorithms use linear and optimal control techniques to fully integrate environmental and dynamic constraints into the cost function for optimal solutions, as detailed in Figure 121. Key methods include NLP, which uses the flatness-based approach by Chamseddine et al. (2012) to linearize complex kinodynamic constraints. MILP by Yue et al. (2009) and Binary Integer Programming (BIP) by Masehian and Habibi (2007) demonstrate robust modeling for various applications, with BIP specifically suited to binary decision contexts.

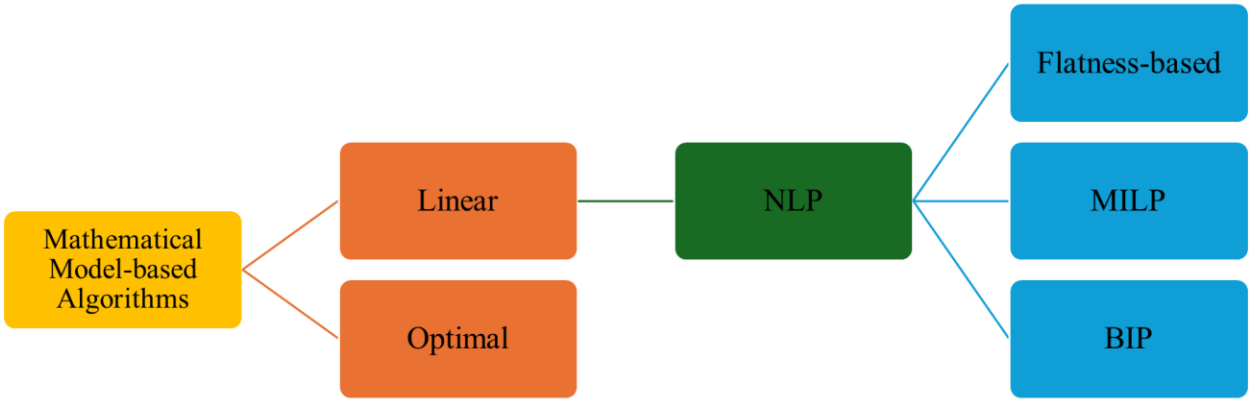


Figure 121. Mathematical Model-Based Algorithms

6.3.3.1 Linear Algorithms

Linear algorithms effectively represent environments and address kinematic and dynamic constraints, managing control disturbances and uncertainties well. MILP methods are notable for integrating binary and integer logic constraints, accurately reflecting real-world scenarios.

To support MILP applications in path planning, Bhattacharya (2006) introduced OPTRAGEN, an open-source MATLAB toolbox, tailored for MILP challenges. On another hand, Masehian and Habibi (2007) also utilized binary integer programming, applying binary integers to model path length variables for efficient path planning.

6.3.3.2 Optimal Control

The path planning challenge can be formulated within the framework of optimal control, which seeks to determine both the current state and control-oriented path by leveraging differential equations (Anderson et al., 2010). Optimal control is essentially an evolution of linear algorithms

to scenarios encompassing an infinite array of variables, offering a straightforward approach to incorporating uncertainty through linear chance constraints.

Figure 122 illustrates a generic optimization problem in flowchart form, explicitly incorporating both the initial (or current) state and the goal state within its constraints to guarantee a comprehensive solution.

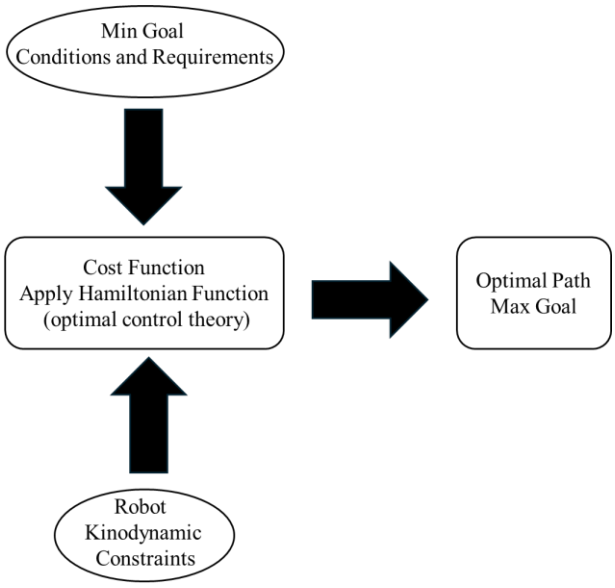


Figure 122. Generic Optimization Problem

6.3.3.3 Summary of Mathematic Model-Based Algorithms

These methods offer a comprehensive framework for describing states and surrounding variables in complex 3D environments. They can dynamically adjust by incorporating a wide range of constraints tailored to specific conditions. However, such algorithms often result in complex formulations, leading to significant computational demands. A solution to mitigate these challenges involves discrete decision-making within the optimization process, allowing for on-the-fly problem-solving through strategic environment modelling (Ma and Miller, 2006).

6.3.4 Bioinspired Algorithms

Path planning is crucial for autonomous robot operation, aiming to enable robots to perform tasks unsupervised in complex environments. Bioinspired algorithms, which mimic biological processes, simplify the challenge by avoiding detailed environmental modeling and using stochastic methods to find near-optimal paths. These methods effectively overcome the limitations of traditional mathematical models, especially in solving NP-hard problems with numerous variables and nonlinear objectives where typical approaches may fail due to local minima (Aghababa, 2012).

Bioinspired algorithms include Evolutionary Algorithms (EAs) and NNs, each offering different analytical approaches and levels of abstraction. Figure 123 illustrates popular bioinspired methods:

- GA (Holland, 1975): Uses population-based numerical optimization.
- MA (Moscato and Norman, 1992): Combines local search with population-based techniques for combinatorial optimization.
- PSO (Kennedy, 2010): Inspired by social behaviors of birds and fish for stochastic optimization.
- ACO (Dorigo et al., 1996): Models ant pheromone behavior to discover shortest paths.
- SFLA (Eusuff and Lansey, 2003): Merges MA and PSO techniques into a unified approach.

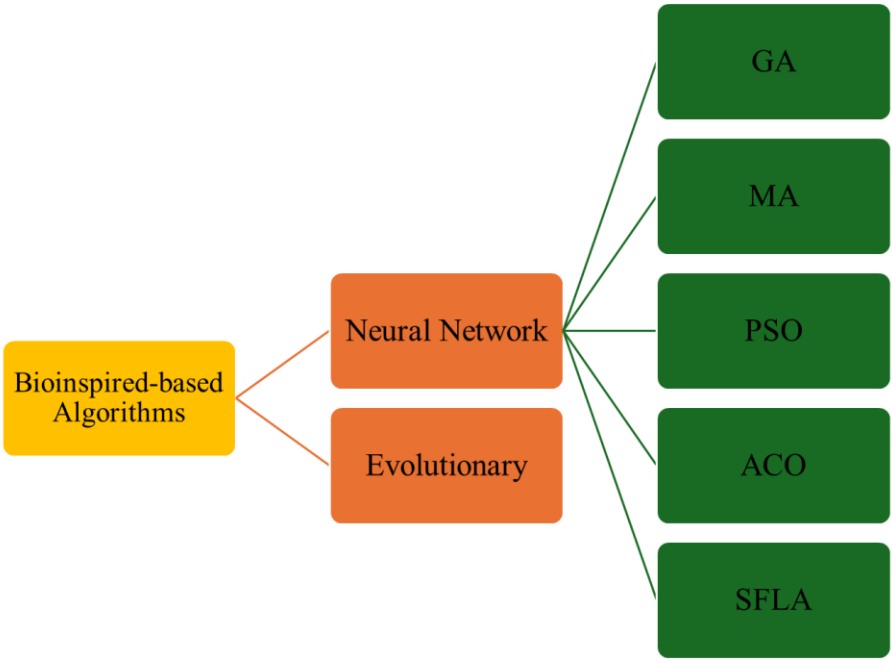


Figure 123. Bioinspired-Based Algorithms

6.3.4.1 Neural Network (NN)

Glasius et al. (1995) presented the NN for obstacle avoidance and navigation, have become a prominent method in path planning across various fields. NNs dynamically represent environments in a way that mimics neural processing, similarly to the APF approach, where unexplored areas globally attract the robot. Robot behavior in NNs is typically governed by the shunting equation, focusing on the neuron with the highest activity to guide movement.

Despite widespread use, NNs share a common drawback with other bioinspired algorithms: a lack of standardized methods, creating uncertainties in reliability and time efficiency. The Hopfield model attempts to mitigate these issues but is only suitable for specific problem types (Ahn et al., 2001).

6.3.4.2 Summary of Bioinspired Algorithms

Bioinspired algorithms mimic natural behaviors to create optimization rules and models. They operate through a structured, iterative process dependent on their initial setup and require significant computational resources in complex environments, as shown in Table 24.

EAs are a type of population-based algorithm involving stages like reproduction, mutation, recombination, and selection. While effective in multi-objective and NP-hard problems, they are prone to time-intensive operations and premature convergence. GA are well-known but slow; PSO is faster but parameter-sensitive. Hybrid methods can improve PSO but do not fully resolve its issues with early convergence (Tang et al., 2005). ACO avoids premature convergence and seeks optimal paths but is less effective in complex settings. SFLA mixes PSO's local search with solution "shuffling" to optimize exploration, dependent on specific settings (Rahimi-Vahed and Mirzaei, 2007). MA modify GA's approach by exchanging gene segments and integrating local learning, increasing computational load. NN handle environmental changes well but are highly dependent on their model configuration.

Table 24. Summary of Bioinspired Algorithms

Bioinspired Algorithms Type	**Deficiencies**	**Improvements**	**Advantages**
GA	Involves a high computational cost and can converge to suboptimal solutions prematurely.	Adjustments have been made to enhance efficiency and convergence quality.	Capable of solving complex problems that are NP-hard and accommodating multiple objectives.
ACO	This method also has a high computational cost.	Enhancements target improved efficiency.	Effective in dealing with problems involving multiple

			objectives and continuous spaces.
PSO	High computational cost and tendency for premature convergence.	Modified to improve its performance and handling of individual problems.	Operates faster than the GA and can handle problems with a small population size.
SFLA	It has a high computational cost and is sensitive to parameter settings.	Optimizations aim at improving efficiency and convergence.	More efficient than PSO in achieving faster global convergence.
MA	Characterized by high computational cost.	Improved to enhance path smoothness and reduce computational complexity.	More efficient than GA, particularly for path smoothness and computational efficiency.
NN	Computational cost is high as well the need for suitable rules and structures.	Various strategies have been proposed to improve the stability and efficiency.	Robust to sudden changes in the network, offering stability in dynamic environments.

6.3.5 Multifusion Based Algorithms

Fusion enhances 3D path planning by integrating algorithms layer-by-layer for optimal real-time performance and non-local optimality. APF algorithms risk entrapment in local minima without adjustments, and PRM alone cannot derive optimal paths. Consequently, multifusion-based algorithms merge techniques to overcome these limitations, particularly in environments with dynamic or static obstacles.

Traditional single-planning methods often fail to be cost-efficient, converge quickly, or be computationally effective. For example, PRM and Voronoi diagrams cannot independently generate optimal paths, potential field methods get stuck in local minima, node-based methods

need predefined layouts, mathematical models are time-consuming with NP-hard problems, and bioinspired algorithms depend heavily on their specific models. Integrating multiple strategies, therefore, helps achieve quick, globally optimal solutions.

Key studies include Yan et al. (2013) combining 3D PRM with A* for optimal pathfinding, Masehian and Amin-Naseri (2004) merging visibility graphs, Voronoi diagrams, and potential fields for a balance between shortest and obstacle-free paths. While Schøler et al. (2012) integrated visibility graphs with Dijkstra's, and Zhang et al. (2010) blending mathematical models with evolutionary algorithms to address NP-hard challenges and premature convergence in EAs.

This research classifies multifusion strategies into Embedded Multifusion Algorithms (EMA), which synergize various algorithms concurrently, and Ranked Multifusion Algorithms (RMA), which organize algorithms hierarchically.

Table 25 details representative algorithms from each category, highlighting their methodologies and applications.

Table 25. Summary of Multifusion-Based Algorithms

Subcategory Name	**Description and Typical Elements**
Embedded Multifusion Algorithms (EMA)	These algorithms consolidate multiple data sources or sensor inputs, exemplified by Recursive Dependency Resolving (RDR), Voronoi Partition-Based Potential Field Algorithms, and Neural Network-Based Potential Field Algorithms. This category also includes hybrid models inspired by biological system.

Ranked Multifusion Algorithms (RMA)	Algorithms in this subcategory enhance decision-making by ranking data inputs, employing node-based optimization like PRM, and using geodesic calculations in Visibility Graph-Based Algorithms. They also incorporate sampling techniques such as EA for optimization.

6.3.6 Supervised Learning Based Motion Planning

Recent research in supervised learning-based motion planning can be categorized into two methods: (i) methods aiming to replace the entire traditional motion planning pipeline (End-to-end Solutions) and (ii) methods designed to improve specific components of existing motion planning algorithms (Module Solutions). The first category develops systems for directly generating end-to-end collision-free paths or trajectories in a given configuration space. The second category enhances parts of the motion planning process, such as preprocessing, prediction, executing, and collision-checking, to optimize performance.

Reinforcement Learning (RL) combines optimal control with trial-and-error mechanisms from animal psychology (Nian et al., 2020) and includes value-based, policy-based, and actor-critic approaches. RL has been applied to motion planning by framing it as a Markov Decision Process (MDP), with some strategies providing comprehensive solutions (End-to-end Solutions) while others enhance existing frameworks by substituting specific components (Module Solutions).

In the context of learning-based approaches for motion planning, the typical framework is structured into several distinct modules, as shown in Figure 124 and outlined below:

- **Preprocessing Module** ($H: C \rightarrow C_{pro}$): This initial stage processes the current configuration space and sensor data, outputting a refined configuration space (). Its functions include simplifying the overall configuration space for enhanced search efficiency, transforming the configuration space into a more manageable form for planning, and representing obstacles with simplified, lower-dimensional data. This module is generally activated at the start of the motion planning process.
- **Prediction Module** ($P: U \times X \rightarrow C_{pro}$): Serving a purpose similar to the preprocessing module, the key distinction lies in its repeated application throughout the motion planning process. It prepares the configuration space for subsequent steps by refining or updating its parameters based on predictive models.
- **Executing Module** ($E: C_{pro} \times U \rightarrow X$): This component is responsible for selecting an appropriate action from the set of available actions *(U)* based on the processed configuration space (C_{pro}), resulting in the transition to a new state.
- **Collision-Checking Module** ($O: C_{pro} \times X \rightarrow \{True, False\}$): It evaluates if the new state will result in a collision within the obstacle regions of the configuration space, ensuring the path's feasibility.

In motion planning, while executing and collision-checking modules are essential, preprocessing and prediction modules are optional and adaptable based on the approach used. Deep learning enhances motion planning by enabling neural networks to replace these modules, acting as complex mapping functions. When neural networks are applied across the entire motion planning framework, creating an end-to-end system, they streamline the process through automated, integrated analyses.

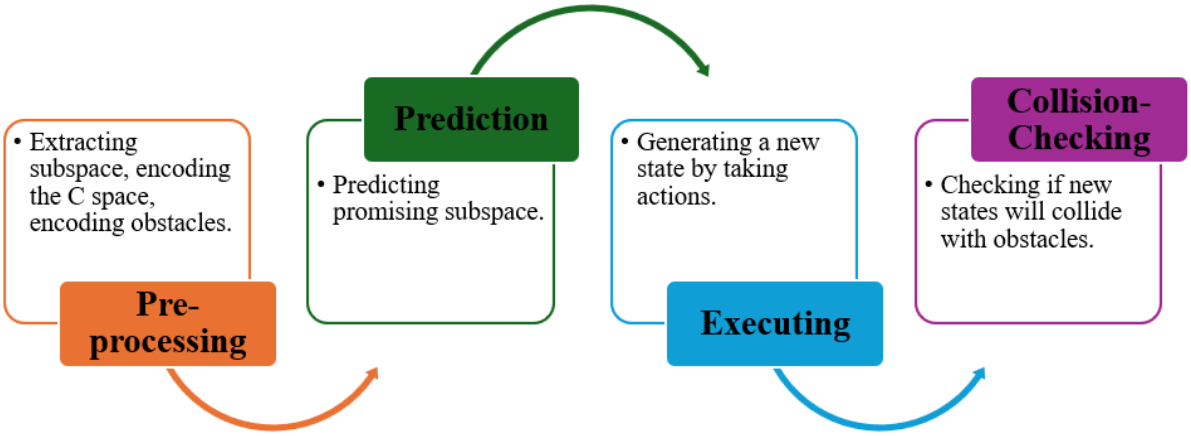


Figure 124. Framework of a Generic Motion-Planning Algorithm

6.3.6.1 Markov Decision Process Formulation

In RL scenarios, the task of motion planning is conceptualized as a MDP, structured as a tuple (S, A, P, R, γ) encompassing:

- ***S***: the state space, incorporating both the robot's configuration space and environmental observations (such as a 2D or 3D map or sensor data). Each state $s_t \in S$ adheres to the Markov property, meaning it contains all relevant information for decision-making at that point.
- ***A****:* the action space, comparable to the control space U mentioned previously, with each action $a_t \in A$ representing a potential maneuver or operation the robot can perform.
- ***P***: the state transition probability, indicating the likelihood of moving from the current state $s_t$to a subsequent state s_{t+1}upon executing an action a_t. This probability could be deterministic or stochastic, reflecting environmental uncertainties.
- ***R***: the reward function, which assigns a value or reward to actions taken in specific states to guide the robot towards desirable behavior. The reward $R(s_t, a_t)$ functions similarly to

a cost function in traditional motion planning, aiming to encode preferred actions within certain contexts.

- $\boldsymbol{\gamma}$: the discount factor, with $\gamma \in [0,1]$, used to modulate the future rewards' present value, emphasizing the importance of immediate rewards over distant ones.

RL algorithms aim to discover a policy $\pi(a|s)$ that optimizes action selection to maximize cumulative rewards over time, known as the return or accumulated reward. This return is often calculated as the sum of γ discounted future rewards, establishing a framework within value-based RL methods for quantifying and enhancing long-term reward outcomes through strategic action choices. In RL, the policy can be determined by selecting the action that offers the highest value. Within policy-based RL approaches, the policy is explicitly modeled using a neural network:

- $\pi_\theta(a_t|s_t)$, where the neural network, parameterized by θ, accepts the current state *s* as input and directly outputs the optimal action.

6.3.6.2 End-to-end algorithms

Recent advancements in data-driven, end-to-end motion planning for autonomous robotics primarily utilize machine learning models to process sensory inputs and generate navigational commands. Notable developments include:

- Pfeiffer et al. (2017): Introduced a method using Convolutional Neural Networks (CNNs) and Fully Connected layers to process laser data for navigation.
- Hamandi et al. (2019): Employed Deep Neural Networks (DNNs) combined with Long Short-Term Memory (LSTM) cells and ResNet architecture to interpret LiDAR data for speed and direction prediction.

- Bency et al. (2019): Developed OracleNet, a Recurrent Neural Network (RNN)-based algorithm for determining collision-free paths in static environments.
- Kurutach et al. (2018): Used a Generative Adversarial Network (GAN)-inspired causal InfoGAN model for generating feasible observation sequences for motion planning.
- Qureshi et al. (2019): Presented Motion-Planning Networks, a DNN-based iterative algorithm for environmental point cloud processing, demonstrating rapid and efficient trajectory generation for a seven-degree-of-freedom Baxter robot.
- Ichter and Pavone (2019): Introduced a learned space adaptation of traditional sampling-based motion planning techniques for increased efficiency.
- Huh et al. (2021): Combined supervised learning with a cost-to-go function in their c2g-HOF network, enhancing high-dimensional motion planning through continuous cost function training and gradient-based path optimization.

These innovative approaches leverage various neural network architectures to improve the efficiency and effectiveness of motion planning in robotics.

6.3.6.3 Module replacement algorithms

Recent developments in motion planning have introduced hybrid learning-based algorithms that combine deep learning technologies with traditional motion planning frameworks to enhance classical strategies. Techniques like the Conditional Variational AutoEncoder (CVAE), CNN, and GAN are used to preprocess configuration spaces, providing a refined starting point for developing traditional motion planning strategies (Ichter et al., 2020).

A significant development is the Deep Sampling-based Motion Planner (DeepSMP) by Qureshi and Yip (2018). DeepSMP merges deep neural networks with sampling-based planning

techniques, using an Autoencoder to process environmental data from point clouds. This information is integrated with initial and goal configurations into a Dropout-based, stochastic, deep feedforward neural network. The network efficiently generates sampling nodes for use in sampling-based methods like RRT, improving the speed and reliability of finding viable, collision-free paths. DeepSMP's enhanced convergence rate demonstrates the potential of hybrid algorithms to advance motion planning.

6.3.7 Unsupervised Learning Based Motion Planning

While supervised learning-based motion planning algorithms are common, the potential of unsupervised learning in this area is just beginning to be explored. Sarker et al. (2019) developed PROM-Net, an innovative motion prediction network that predicts robot movements from raw video footage without supervised input, notable for its efficiency and suitability for devices with limited processing capabilities. Meanwhile, Yang et al. (2020) created Plan2vec, an unsupervised learning algorithm for path planning that constructs a weighted graph from near-neighbor distances and uses path-integral principles to derive a local metric for global embedding. Plan2vec has demonstrated effectiveness in reducing planning costs and enhancing reactive planning capabilities, highlighting the advantages of unsupervised learning in motion planning.

6.3.7.1 Learning by demonstration (LbD)

RL typically requires extensive interaction with an environment, which may not be feasible in all practical scenarios. LbD provides an effective alternative, particularly in high-dimensional motion planning spaces, by leveraging expertise from a proficient demonstrator (Argall et al., 2009). For example, in robotic arm motion planning, observing a human hand can offer instructional examples for learning (Mukherjee et al., 2022).

GMM and Task Parametrized Gaussian Mixture Models (TP-GMM) are prominent in capturing and encoding human trajectory data, effectively managing uncertainties with probability theory (Calinon and Billard, 2007; Ghahramani, 2015). Duque et al. (2019) applied TP-GMM in an LbD strategy for robotic assembly, involving three phases: collecting demonstrator-provided positional data, training the TP-GMM model for each assembly subtask, and synthesizing tailored trajectories based on learned models and object positions.

6.4 Analysis and Conclusion

This research reviews literature on 3D path planning, defining terms and assessing various algorithms and their application in robotics. It categorizes these into five groups: SBA, node-based optimal, mathematical model-based, bioinspired, and multifusion-based algorithms. Each category is evaluated on its classification, mechanisms, environmental suitability, and application rationale.

Key findings include:

- Sampling-Based Algorithms: These use an initial guess strategy to escape local minima and can actively identify optimal paths, making them suitable for real-time applications in both static and dynamic environments.
- Node-Based Optimal Algorithms: Operating on a grid, these algorithms assign weights using node and arc information, effective with a pre-established graph but limited by environmental decomposition.
- Mathematical Model-Based Algorithms: Offer detailed mathematical workspace representations and prioritize safety and reliability, though they require significant computational power.
- Bioinspired Algorithms: Use heuristic principles to handle complex constraints and NP-

hard challenges through iterative optimization. Suitable for offline use, their mutation process can delay computation.

- Multifusion-Based Algorithms: Combine various methods for global optimization and cost minimization, designed for real-time use with robust environmental adaptability.

The research suggests that future advancements should focus on real-time planning, improved information representation, and complex environment modeling. It recommends integrating multipath planning with environmental modeling and considering control uncertainty.

6.4.1 Scope and Challenges

In dynamic environment motion planning, heuristic methods have proven to be more effective than traditional algorithms, particularly with the complexities of NP-hard problems (Raja P, 2012). Despite advancements, there's significant room for improvement in online motion planning algorithms to handle moving obstacles and improve route quality. Key challenges include:

(1) Relative Velocity Impact: The velocity vector of an obstacle relative to the robot is critical and should be a focus in new motion planning strategies.

(2) Inclusion of Kinematic Constraints: Many methods overlook real-world kinematic constraints that affect robot steering, which should be incorporated to bridge theoretical models with practical applications.

(3) Real-Time Obstacle Recognition and Path Optimization: Efficient navigation requires immediate recognition of moving obstacles and dynamic generation of optimal paths.

(4) Solution Guarantees: Heuristic methods may not always find a solution; integrating multiple heuristics could enhance reliability.

(5) Dynamic Environment Complexity: The changing nature of dynamic environments due

to obstacle movements calls for advanced hybrid meta-heuristic methods.

(6) Additional Challenges: Complexities such as multiple optimization goals, coordination among multiple robots, and uncertainties in obstacle detection and path control require innovative solutions in mobile robotics.

Addressing these challenges could lead to more robust and efficient motion planning systems capable of operating effectively in dynamic settings.

6.4.2 Conclusion

This research examines learning-based robot motion planning algorithms, covering both classical and learning-enhanced strategies involving supervised and unsupervised learning. A key limitation of supervised learning in motion planning is its optimal performance only in environments similar to the training data, leading to poorer results in unfamiliar or complex settings. Additionally, these algorithms often struggle with navigating large-scale maps, highlighting the need for more adaptable solutions across different environmental scales.

Another challenge is the difficulty in obtaining comprehensive datasets for supervised learning, while RL methods require substantial interaction with environments, leading to inefficiencies. A promising research direction is to integrate the strengths of supervised and RL-based methods to overcome their individual limitations.

Motion planning continues to face an expanding range of challenges, such as collision avoidance, path optimization, computation efficiency, adaptability to changing environments, balancing multiple criteria, and managing dynamic constraints. These issues emphasize the necessity for more sophisticated and effective algorithmic approaches in motion planning.

Chapter 7 : Enhanced Simulated Motion Planning Framework for Robotic Assembly in Offsite Construction

Abstract: This research presents an advanced algorithm that can enhance the trajectory planning process in offsite construction, specifically for assembling prefabricated building components using robotic automation. The algorithm integrates geometrical multi-level collision-check method with the rapidly exploring random Tree-Star (RRT*) algorithm, which improves its capacity to plan trajectories and prevent collisions. This approach significantly improves collision avoidance, trajectory smoothing, and efficiency compared to traditional human-operated methods. The research focuses on the development of a motion planning algorithm that accurately generates kinematic parameters for robots assembling prefabricated components. The prototype's performance was tested using through simulations and in-lab experiments, demonstrating efficiency in executing a logical assembly sequence, ensuring reachability to assembly coordinates, and dynamically avoiding obstacles. The algorithm's effectiveness is underscored by its successful application in assembling offsite wall and floor panels, a task traditionally prone to collision challenges. The results indicate a success rate of over 80% in finding collision-free paths in various simulated environments. Furthermore, the research presents the Robotic Motion Planner (RMP), a cutting-edge tool designed to optimize robotic movements for more efficient assembly processes. The RMP breaks down robot trajectories into discrete motions, considering the constraints of the surroundings and the robot's capabilities. This involves a comprehensive algorithm that incorporates inverse kinematics, collision detection, and singularity avoidance, tailored to the specific needs of large and complex building components. The developed algorithm uses AI to autonomously optimize paths generated from the Assembly Sequence Planner (ASP). It includes a user interface for fine-tuning auto-generated paths, allowing adjustments in

manipulation time, path constraints, and workspace specifications. The tool visually represents the simulation of robotic motion planning, providing an intuitive understanding of the robot's joint configurations and movements. Additionally, it records and visualizes the flexion and extension angles of the robot's joints, offering detailed data streams for further refinement of the planning results. A pivotal feature of the system is its ability to connect with real robots, directly exporting optimized movements as commands, thereby bridging the gap between simulation and practical application. In conclusion, this research presents a comprehensive, AI-driven solution to the challenges of trajectory planning in assembling prefabricated building components. Its innovative approach in integrating geometry-based collision checks with advanced algorithms marks a significant leap forward in robotic automation for offsite construction.

7.1 Introduction

The expanding field of OSC, particularly prevalent in North America in recent years, has been a game-changer in the construction industry. This method, characterized by transferring a significant portion of on-site construction into a controlled indoor environment, has been praised for its ability to boost productivity, reduce costs, and improve safety. However, it necessitates sizable factory footprint and involves extensive manual labor. To address these challenges, innovative approaches such as automation and robotics have been integrated into the prefabrication process, aiming to reduce labor requirements and enhance design flexibility in building construction. The emergence of industrial robot arms in prefabrication, benefiting from their decreasing cost and expanding capabilities, represents a significant trend in this evolution.

Key to the robotization of conventional construction tasks is the concept of flexible automation, underpinned by sophisticated trajectory planning. As highlighted in the literature, trajectory

planning remains a core issue in robotics (King et al., 2014). This planning, crucial for enabling autonomous robotic platforms to navigate effectively and adjust their movements in real-time, has gained considerable attention for construction tasks. Research contributions in this field include the use of the Bucket Brigade algorithm for shortest trajectory identification in building block assembly (Terada & Murata, 2008), the Lin-Kernighan algorithm for toolpath optimization in concrete contour crafting (Davtalab et al., 2018), and the Bidirectional Evolutionary Structural Optimisation algorithm for optimizing clay structure infill patterns and establishing nozzle trajectories in additive manufacturing (Kontovourkis et al., 2020). Moreover, developments like Ding et al.'s (2020) deep convolutional neural network for brick pattern identification and King et al.'s (2014) digital workflow for robotic tile placement using Rhinoceros and ABB RobotStudio, further emphasize the growing sophistication in this field.

Despite these advancements, there remains a lack of mature solutions specifically tailored for the robotic assembly of lightweight structures in OSC manufacturing facilities. This research aims to fill this gap by developing a collision-free trajectory planning method for the robotic assembly of OSC prefabricated components like walls and floors.

The IR arm, with its advanced controls, mechanical systems, and high DOF manipulators, is ideally suited for performing precise and repetitive tasks such as component assembly. In the context of prefabrication, finding collision-free paths for assembling a large number of building components is a critical challenge. Unlike more uniform manufacturing processes, prefabricated and modular home building involves thousands of distinct components, necessitating a path-finding method that can accommodate this diversity. Additionally, the involvement of workers in

tasks like material preparation and quality inspections necessitates a path-finding approach that not only identifies collision-free paths but also facilitates human-robot interaction planning.

The objective of this research is to develop a collision-free path-finding method specifically for robotic OSC residential prefabrication. Traditional path-finding methods fall short in addressing the unique demands of this process, which include accommodating the diverse specifications of building components. This study introduces a novel path-finding approach employing game engine technology and a sampling-based algorithm. This method not only ensures the robot arm navigates collision-free paths but also visualizes these paths in a virtual environment, paving the way for broader applications in the field of OSC.

7.2 Literature Review

The integration of IR arms in the OSC prefabrication process represents a rapidly growing trend in recent construction practices. An important challenge in this integration is the development of collision-free paths for the robot arm, which must handle a diverse array of building components. This issue, though extensively explored in the area of computer science and mechanical engineering, still lacks a dedicated path-finding solution specifically tailored for robotic OSC prefabrication.

The manufacturing industry, with its emphasis on mass production, typically employs pre-planned assembly processes that are repeatedly applied to identical products. In contrast, the field of OSC is characterized by continuous changes in product design, including variations in size and shape (Iturralde and Bock, 2018). This dynamic nature of OSC necessitates a refinement approach to RMP, one that not only acknowledges the unique specifications of modular construction but also leverages the full capacity of the IR arm.

To address these complexities, this research is structured into three parts. The first part discusses the specifications of the OSC construction approach, examining the unique challenges and requirements that distinguish it from traditional construction methods. The second part explores the applications of robotics within the area of OSC, highlighting how robotic technology can be adapted and optimized for this specific context. The final part focuses on robotic path planning methods, reviewing existing strategies and identifying the adaptations necessary for their effective application in the modular home prefabrication process.

This comprehensive review aims to bridge the gap between the established theoretical foundations of robotics and the practical requirements of OSC. By synthesizing insights from these distinct yet interconnected domains, the research seeks to pave the way for innovative solutions in robotic path planning that are specifically designed for the challenges of OSC prefabrication.

7.2.1 Specifications of Modular Home Construction

1. **Handling of Numerous Building Components**: OSC homes are constructed using factory-built panels or modules, a method that shifts a significant portion of construction work to a controlled factory environment (Carlson, 2015). In this setting, up to 90% of the building process occurs off-site, with the entire structure being broken down into manufacturable pieces like floor panels, wall panels, and roof trusses (Kamali and Hewage, 2016). This approach often involves the management and assembly of thousands of different components within the factory, necessitating precious scheduling and planning. For example, a two-storey wood-framed home with a basement might require assembling over 50 different wall panels, encompassing more than 900 individual components (e.g., wood studs, sheathing boards).

2. **Production of Large Building Components**: OSC must also contend with the production of extremely large components, driven by transportation limitations. Prefabrication factories, drawing from industrialized production lines, aim to produce building components that maximize the use of space in shipping trailers to minimize transportation costs (Smith, 2010; Salama et al., 2017). These components are often tailored to fit standard trailer sizes, such as 53' x 102" and 48' x 96', necessitating a factory layout that can accommodate such large-scale production (Schoenborn et al., 2020).

3. **High Level of Human-Machine Interaction**: Despite the shift to a controlled environment, OSC still heavily relies on manual labor. On average, each module in the U.S. requires about 208 labor hours, including various stages from subassembly to shipping (Mullens, 2011). This blend of human and machine work raises potential safety concerns. To mitigate these risks, frameworks and tools like 3D visualization of the workspace have been proposed to enhance safety and efficiency in the factory (Golabchi et al., 2015).

7.2.2 Advantages and Challenges of OSC

The OSC method offers significant advantages over conventional construction, including higher efficiency, lower costs, and reduced uncertainties due to environmental factors. McKinsey's 2019 report highlighted that OSC could reduce construction costs by up to 20% (Bertram et al., 2019). By operating in a controllable environment, the method accommodates delays caused by weather, protects components from environmental damage, and safeguards workers from unpredictable onsite hazards (Gibb, 1999).

However, these benefits come with the challenge of adapting the production process to the unique requirements of OSC. The need to manage a vast array of components, produce large-scale pieces,

and plan effective human-machine interactions requires innovative approaches in factory layout, production planning, and safety management. As the industry evolves, addressing these specifications will be crucial for maximizing the potential of OSC in the modern building landscape.

The integration of automation and robotics into prefabricated OSC, particularly in the area of modular home construction, is an emerging and transformative trend. Despite the cost and uncertainty reductions offered by OSC, it faces challenges such as limited design versatility, the need for large factory spaces, and reliance on manual labor. To address these limitations, there has been a significant push towards incorporating advanced technologies, particularly robotics, which are becoming more affordable and user-friendly (Bock, 2015).

7.2.3 Advancements in Robotics and Automation for OSC

(1) **Robotic Systems for Prefabrication**: Over the past two decades, several robotic systems have been proposed and implemented in OSC. The ROCCO project introduced robotic systems for erecting prefabricated walls (Gambao et al., 2000), and the FutureHome project transformed a gantry crane into a robotic system for on-site modular building assembly (Balaguer et al., 2002). Bock (2007) proposed a multifunctional unit for the flexible production of concrete panels, and SAM100, developed by Construction Robotics, has been used to automate the bricklaying process, demonstrating the diversity of applications in robotic technology.

(2) **Automation in Building Component Manufacturing (BCM)**: Bock and Linner (2015) extensively reviewed automation and robotics technologies used in BCM, highlighting

state-of-the-art systems for materials such as ceramic, concrete, wood, and steel. Their work emphasizes the broadness of automation applications in the prefabrication process.

(3) **Autonomous Planning and Supporting Systems**: The field has also seen the development of systems aimed at enhancing production planning and control. Altaf et al. (2018) integrated RFID, data mining, and simulation for panelized home manufacturing. Martinez et al. (2019) developed a vision-based system for steel frame assembly inspection, and Kasperzyk et al. (2017) proposed a Robotic Prefabrication System (RPS) capable of disassembling and reconstructing structures for design changes.

(4) **IR Arms in OSC**: The declining cost and increasing capabilities of IR arms have stimulated their adoption in modular construction. These arms, with high DOFs can perform more complex tasks than machines with fewer DOFs. Eversmann et al. (2017) used IR for prefabricating truss structures, Dörfler et al. (2019) developed a robotic system for rebar forming, and Huang et al. (2018) created a method for robotic spatial extrusion of 3D trusses using the RRT* algorithm. Iturralde & Bock (2018) integrated these technologies for building renovation, while the Randek ZeroLabor (2021) Robotic System automated the assembly of timber panels in OSC homes.

7.3 Challenges and Innovations in Robotic Path Planning

Despite these advancements, adapting preplanned robot motions to real and changing environments remains a significant challenge. Iturralde et al. (2019) explored methods for object recognition and robot pose adjustment during assembly, comparing techniques like OpenCV with ArUco markers and digital theodolites. This highlights the ongoing need to refine robotic systems to account for deviations and displacements during the assembly process. The integration of IR

arms into OSC processes, particularly in prefabricated and modular OSC sectors, has highlighted the critical challenge of collision-free path planning. While their application is gaining attention, specific research on collision-free path-finding within this context is relatively scarce, despite motion planning being a well-established field in robotics (Latombe, 1991).

It is important to note that the shift towards automation and robotics in prefabricated and modular OSC is marked by significant technological advancements and innovative applications. These developments are not only enhancing the OSC efficiency and versatility but also reshaping the landscape of the construction industry by addressing its traditional limitations.

7.3.1 Key Concepts and Challenges in Robotic Path Planning

1. **Configuration Space (C-space)**: The concept of C-space is fundamental in robotics for addressing the path-finding problem. However, identifying obstacle-free paths for manipulators in C-space is recognized as a PSPACE hard problem, indicating significant computational complexity (Reif, 1979).

2. **Sampling-Based Algorithms**: To manage this complexity, sampling-based algorithms have been developed, offering more efficient, approximate methods for path planning (Ichter et al., 2018). These algorithms simplify the task by probabilistically sampling the C-space to find feasible paths.

3. **Trajectory Planning**: The focus of trajectory planning is to determine an optimal geometrical feasible path for a robotic arm, from a start to a goal position, ensuring collision avoidance and adherence to kinematic constraints (Xiao et al., 2021). Optimal trajectories are those that minimize sharp turns and ensure smoothness.

4. **Approaches to Trajectory Planning**: Vagale et al. (2021) and Souissi et al. (2013) differentiate between classical and advanced approaches to trajectory planning. The classical approach is suitable for predictable environments with static obstacles, while the advanced approach addresses dynamic, unpredictable environments requiring real-time trajectory modifications.

7.3.2 Specific Algorithms and Tools

The field of robotic path planning features several classical approaches, each with its unique methodologies and applications. These approaches include, the cell decomposition, the optimization-based, the probabilistic roadmap, the circular fields, and the potential field as outlined by Souissi et al. (2013).

1. **Cell Decomposition Approach**: This technique involves dividing the robot's configuration space into distinct, non-overlapping cells, thereby creating a connectivity graph to model the connected relationships among these cells (Souissi et al., 2013). Algorithms like Dijkstra's, Bucket Brigade, and Lin-Kernighan Heuristic are then applied to navigate through this graph and formulate the robot's trajectory (Saxena et al., 2020). Terada and Murata (2008), as well as Davtalab et al. (2018), have successfully applied the Bucket Brigade and Lin-Kernighan algorithms, respectively, in construction-related trajectory planning.

2. **Probabilistic Roadmap Approach**: This approach consists of two phases: roadmap construction and querying (Kavraki et al., 1996). During construction, a navigation mesh is created in the configuration space, with vertices randomly placed in free space and linked to their closest neighbors without intersecting any obstacles (Vagale et al., 2021). The

query phase involves searching this mesh using computational geometry, like visibility graphs for shortest paths and Voronoi diagrams for maximum clearance trajectories (Souissi et al., 2013).

a) **PRM**: This method involves random sampling of configuration-free nodes in C-space to create a roadmap. Algorithms like Dijkstra's or A* are then used to find optimal paths on this roadmap (Kavraki et al., 1996).

b) **RRT**: The RRT algorithm grows a tree from the robot's initial configuration, randomly generating configuration-free vertices in C-space (Lavalle, 1998). This approach has been expanded with variations like PRM* and RRT* to improve path-finding efficiency.

3. **Potential Field Approach**: Commonly used in static obstacle environments, this approach employs potential functions, such as electrostatic potential, to navigate the robot. In this context, the goal configuration creates attractive forces, while each obstacle generates repulsive forces, directing the robot toward the goal and preventing collisions (Borenstein and Koren, 1989). Recent adaptations by Fan et al. (2020) and Wang et al. (2019) have attempted to extend this method for dynamic obstacles, though issues like local minima traps still pose challenges (Wang et al., 2019).

4. **Optimization-Based Algorithms**: These algorithms are grounded in the formulation of cost functions that define various parameters for an ideal trajectory, such as minimizing trajectory completion time and adhering to obstacle constraints (Petrović et al., 2019). Key examples include:

a) **CHOMP (Covariant Hamiltonian Optimisation for Motion Planning)**: Created by Ratliff et al. (2009) and enhanced in later versions, CHOMP employs a signed distance field for obstacle detection and covariant gradient descent to minimize cost functions.

b) **STOMP (Stochastic Trajectory Optimisation for Motion Planning)**: Introduced by Kalakrishnan et al. (2011), STOMP begins with an initial stochastic trajectory and optimizes it by sampling various trajectories and combining them to minimize cost functions.

c) **TrajOpt**: Proposed by Schulman et al. (2014), TrajOpt treats trajectory planning as sequential quadratic programming to iteratively converge cost functions to a minimum.

5. **Circular Fields Method**: Drawing from electromagnetism principles, this method views the robot as a charged particle moving through an electromagnetic field, where reactive forces are applied for trajectory planning (Ataka et al., 2018). However, it relies heavily on prior knowledge of obstacle positions and tends to lack path length optimality (Becker et al., 2021). This method requires pre-labeled obstacle coordinates, limiting its flexibility in dynamic environments.

6. **Robotics Software Tools**: Platforms like MoveIt, built in the ROS environment, provide libraries with various planners to aid users in applying path planning methods for collision avoidance. MoveIt is an open-source platform that enables users to prototype designs and develop commercial applications for robots (MoveIt, 2024).

Algorithms like RRT* are primarily valued for their ability to operate in real-time within unpredictable environments featuring dynamic obstacles, without needing prior knowledge of the environment. This capability renders them highly adaptable and suitable for scenarios where conditions are in constant flux. While optimization-based algorithms are adept at calculating cost-efficient trajectories with precision, they can be computationally demanding and susceptible to local minima. On the other hand, sampling-based algorithms like RRT* offer real-time adaptability and a higher likelihood of reaching global optimality, making them particularly effective in dynamic and unpredictable settings.

The choice of algorithm in robotic path planning depends greatly on the specific requirements and constraints of the scenario, balancing the need for precise path optimization against the ability to adapt to dynamic obstacles. This ongoing evolution in the field is driven by the need to develop tailored algorithms and tools for navigating robots, such as IR arms, through complex environments in prefabricated and modular OSC. The continuous advancement of these technologies is crucial for enhancing both the efficiency and safety of robotic applications in the OSC process, highlighting the need for continued innovation and adaptation in robotic path planning.

7..3.3 Current Limitations

The frequent use of algorithms like C-space in robotic applications for motion planning, including in contexts like panelized and modular home OSC, faces three significant limitations that hinder their effectiveness.

1. **Complexity in High-Dimensional Spaces**: The application of C-space becomes exponentially complex as the dimensionality increases (Halperin et al., 2017). In OSC,

where robots often exceed six DOFs, the resulting high-dimensional C-space (more than six dimensions) becomes difficult for engineers to automatically understand and visualize. This complexity poses a challenge in planning and executing human-robot interactions during the robotic prefabrication process, as noted by Vähä et al. (2013).

2. **Computational Expense in Dynamic Environments**: Constructing a C-space in a high-dimensional and highly dynamic environment, as is common in OSC industry, demands substantial computational resources (Henrich and Qin, 1996). Given that the IR arm is required to assemble thousands of different components, and the environment continually changes with each new component, traditional path planning methods become extensively resource intensive.

3. **Difficulty in Handling Varied Components**: Adapting C-space for different objects attached to the robot's end-effector is a time-consuming and computationally demanding task. The varied nature of components, such as wall panels with studs of different lengths (ranging from 4 ft to 80 ft) and sheathing boards of assorted sizes, necessitates a path-finding method capable of adjusting to a wide array of target objects.

In light of these challenges, especially the need for a solution that can handle unpredictable environments with dynamic obstacles and seek global optimality to reduce energy consumption of the robotic manipulator, this research has chosen to adopt the RRT* algorithm. RRT* is selected for its ability to develop a trajectory planning approach that is more suited to the complex and variable requirements of robotic assembly in OSC. This choice reflects an understanding of the specific demands of OSC and the limitations of existing path-planning methods in this context.

7.3.4 Problem Definition and Research Objectives

Traditional wood wall panel fabrication, commonly used in building construction, involves a series of steps to create the structural and non-structural elements of a wall. Here's an overview of the typical process:

1. **Design and Planning**: The first step involves designing the wall panel according to architectural and engineering specifications. This includes determining the dimensions, layout, and the placement of structural elements like tracks, studs, headers, and sheathing. The design is often done using CAD software, which helps in precise planning and optimization of materials.

2. **Material Selection and Preparation**: Materials such as lumber (for studs, tracks, and headers), sheathing materials (like plywood or OSB - Oriented Strand Board), and fasteners (nails, screws) are selected based on the design requirements. The lumber is typically pre-cut to the required lengths, and any special treatments, like weatherproofing or insulation, are also prepared.

3. **Assembling the Frame**: The frame of the wall panel is assembled on a flat surface, usually beginning with the top and bottom tracks (horizontal members of the frame) and then adding the vertical studs. The studs are spaced according to building codes and design specifications, often 16 or 24 inches on-center. Special framing components like window and door headers, corner studs, and cripple studs are also installed during this stage.

4. **Attaching Sheathing**: Once the frame is assembled, sheathing is attached to one side of the frame. Sheathing adds rigidity to the wall and provides a base for exterior finishes. It

is typically nailed or screwed to the studs and plates. The sheathing is often applied in large sheets, which are cut to fit the dimensions of the panel.

5. **Installing Insulation and Vapor Barriers**: In some cases, insulation may be added between the studs of the wall panel at this stage, especially if the panels are being prepared for immediate installation in a building. Similarly, vapor barriers might be installed to prevent moisture penetration.

6. **Quality Checks and Panel Finishing**: After the sheathing is attached (and insulation and vapor barriers are installed, if applicable), the panel undergoes quality checks. This includes ensuring that the panel is square, the studs are properly aligned, and the sheathing is securely fastened. Any necessary adjustments are made.

7. **Transportation to Site**: Once the wall panels are fabricated, they are transported to the construction site. The panels are often labeled or numbered to assist with the correct placement according to the building's layout.

8. **Installation at Construction Site**: At the site, the panels are lifted into place and secured to the foundation and to each other. Additional structural connections are made, and utilities like electrical and plumbing are installed.

This method of wall panel fabrication allows for a quick and efficient building process, as panels can be constructed off-site or on-site in a controlled manner, reducing construction time and often improving the quality compared to traditional stick framing methods.

This research is dedicated to developing a sophisticated collision avoidance method for IR arms utilized in the OSC process, specifically focusing on the assembly of prefabricated multi-wall

panels. These panels, comprising various components like windows and doors, present complex multi-query trajectory planning problems. The approach combines several innovative techniques:

1. **Spatial Analysis of Multi-Wall Panel BIM Models**: The first step involves analyzing BIM models of multi-wall panels to determine the assembly coordinates of prefabricated components. This process also includes sequencing the incorporation of these components into the overall assembly plan, ensuring an efficient and logical order of construction.

2. **Trajectory Planning Using RRT Algorithm*:** The trajectories for assembling each component are defined using a refined version of the RRT* algorithm. Unlike previous studies in construction that often represent robot end-tips in a particle setting for trajectory planning (Davtalab et al., 2018; Kontovourkis et al., 2020; Ding et al., 2019), this approach acknowledges that prefabricated components are not only particles but rigid bodies with significant volume. This distinction is crucial to prevent collisions between the robot and surrounding objects during assembly. For instance, when a robotic manipulator moves a wood stud, considering the stud's volume is essential to avoid collisions with other parts of the structure. The trajectories of each wall panel component are carefully checked for collisions with other elements in the assembly environment.

3. **Adapting to Offsite Prefabrication Specifications**: The proposed path-finding method is tailored to accommodate IR arms, particularly those with at least 6 DOFs, commonly used in OSC prefabrication. It is designed to adapt to various sizes and types of building components, like wood studs and sheets. Furthermore, the method visualizes both the collision-free path and the path-finding process, aiding engineers in planning subsequent human-robot interactions.

The ultimate goal of this research is to establish a path-finding method that not only meets the unique demands of OSC prefabrication but also facilitates smooth and safe human-robot collaboration. This method is expected to be versatile enough to handle a variety of IR arms and diverse building components, ensuring efficient and collision-free assembly in the prefabrication process.

7.4 RMP Research Methodology

This study is centered on developing a collision avoidance strategy specifically engineered for the prefabrication of modular and panelized homes using robotics. The core aim is to describe paths with no obstructions for an IR arm within the context of prefabrication activities. The approach is tailored to align with the unique demands of the OSC prefabrication workflow, unfolding across three principal stages as shown in Figure 125.

In the outset stage, the strategy employs a virtual environment, powered by a game engine, to facilitate the navigation of the IR arm through a Cartesian coordinate setup, thereby precluding the necessity to describe the C-space of the operational setting. The fundamental task at this juncture is to construct a digital twin that accurately mirrors the robotic arm, the manufacturing landscape, and the relevant materials. Furthermore, an innovative collision configuration technique, formed across four levels of detail during robots' active and in-active construction activities, is introduced to refine the collision detection process within the game engine framework.

Progressing to the intermediate stage, an array of robotic configurations is systematically sampled, taking into consideration aspects such as the dimensions of the components to be handled, the distance between their starting and intended positions, and the carrying capacity and reach of the IR arm.

The final stage involves the application of a path-finding algorithm to ascertain routes that are free from collisions, leveraging the previously sampled configurations of the robot. This method is structured within a tiered system comprising assembly-level, manipulator-level, and joint-level planning, each tier addressing distinct facets of the task at hand, with further detailed provided in the sections that follow.

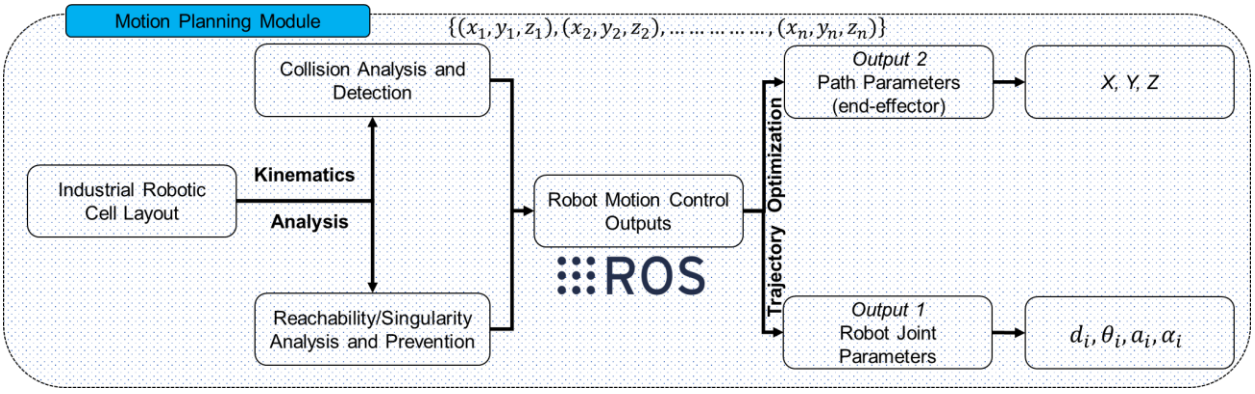


Figure 125. RMP Research Methodology

7.4.1 Manipulator-Level Planning

In this section, we examine the configuration details of two robotic manipulators, focusing on the KUKA KR IONTEC 2500/50 and KUKA KR QUANTEC PA 3200/120 models. This particular manipulator boasts an extensive operational range of 2500 mm and 3200 mm, respectively, and a payload capacity of 50 kg and 120 kg, respectively, making them suitable for a variety of alternating tasks. Both IR arms are equipped with seven joints, as illustrated in Figure 126, with Joint 1 being a prismatic joint featuring ABB linear units KL 4000 that allows for linear movement along its axis, enhancing the manipulator's workspace flexibility. The remaining Joints 2 to 7 are considered as revolute joints generating rotational movement around their axes. With reference to KUKA (2023), the kinematic limits for each joint can be extracted as listed in Table 26.

Table 26. IR Arms Model Specifications

IR Arm Model	KUKA KR IONTEC	KUKA KR QUANTEC PA
Max Payload	61 kg	135 kg
Rated Payload	50 kg	120 kg
Maximum Reach	2501 mm	3195 mm
Motion Range/Speed - Joint 1	KL4000 (14 m)	KL4000 (14 m)
Motion Range/Speed - Joint 2	±185˚ (-3.23 to +3.23 rad)	±185˚ (-3.23 to +3.23 rad)
Motion Range/Speed - Joint 3	-175°/60° (-3.05 to +1.04 rad)	-140°/-5° (-2.44 to +0.08 rad)
Motion Range/Speed - Joint 4	-120°/170°(-2.09 to +2.96 rad)	0°/155°(0 to +2.70 rad)
Motion Range/Speed - Joint 5	±180° (-3.14 to +3.14 rad)	±180° (-3.14 to +3.14 rad)
Motion Range/Speed - Joint 6	±125° (-2.18 to +2.18 rad)	±125° (-2.18 to +2.18 rad)
Motion Range/Speed - Joint 7	±350° (-6.10 to +6.10 rad)	±350° (-6.10 to +6.10 rad)

Enhancing the functionality of the manipulator, a SCHUNK end-effector combo equipped with a magnetic gripper and two-fingers gripper is attached to its structure through a tool change flange and swivel head. Utilizing magnetic gripper suction, this gripper is adept at handling construction materials, as detailed by Huang et al. (2021) (refer to Figure 126). Designed for rigorous use, the

gripper can support loads up to 120 kg, making it capable of managing substantial components like the long-edge beam of an LGS stud or track, which weighs 40.

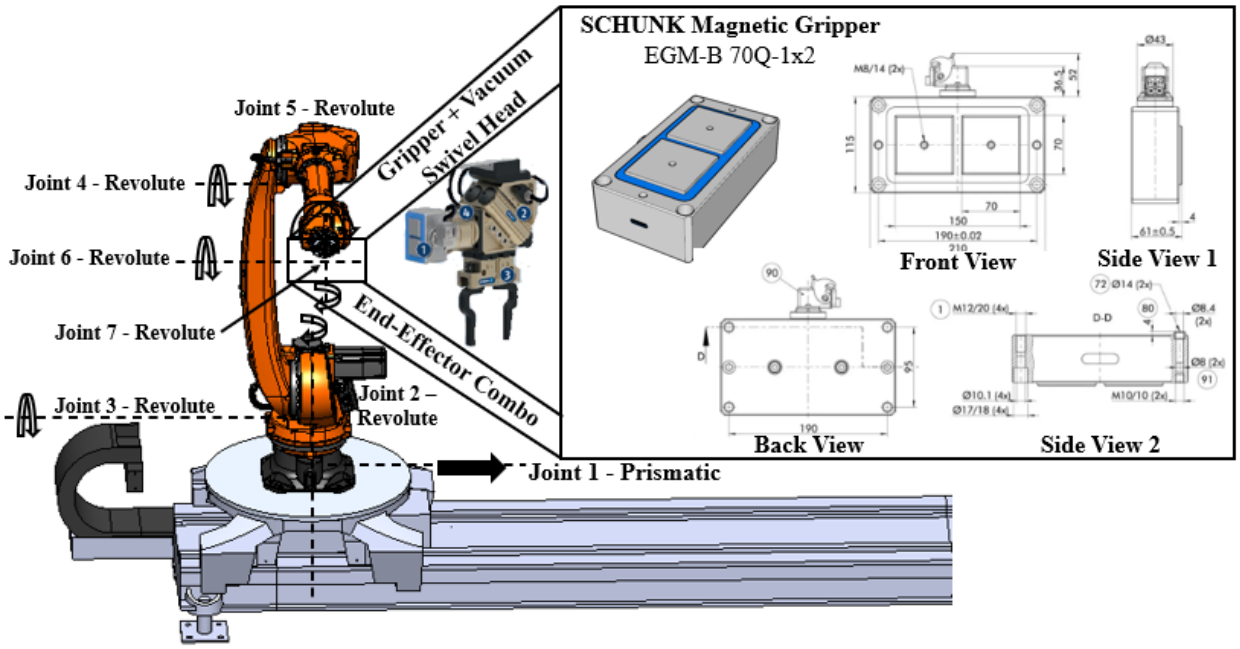


Figure 126. KUKA KR QUANTEC PA 3200 Robotic Manipulator Combo with FK/IK of Joints

The configurations of the robotic cell are shown in Figure 127, where two different robots are mounted on a linear motion rail. Nonetheless, it has been identified that the manipulator's reach is insufficient for the complete assembly of an LGS panelized wall panel, with certain areas lying outside its operational span, as shown in Figure 128. To overcome this spatial limitation, a cooperative assembly strategy involving four KUKA KR robotic manipulators positioned on either side of the wall panel is proposed. This quad-manipulator setup ensures full coverage of the assembly space, effectively overcoming the challenges presented by the dimensions of the wall panel, material picking area, and tool changing zone.

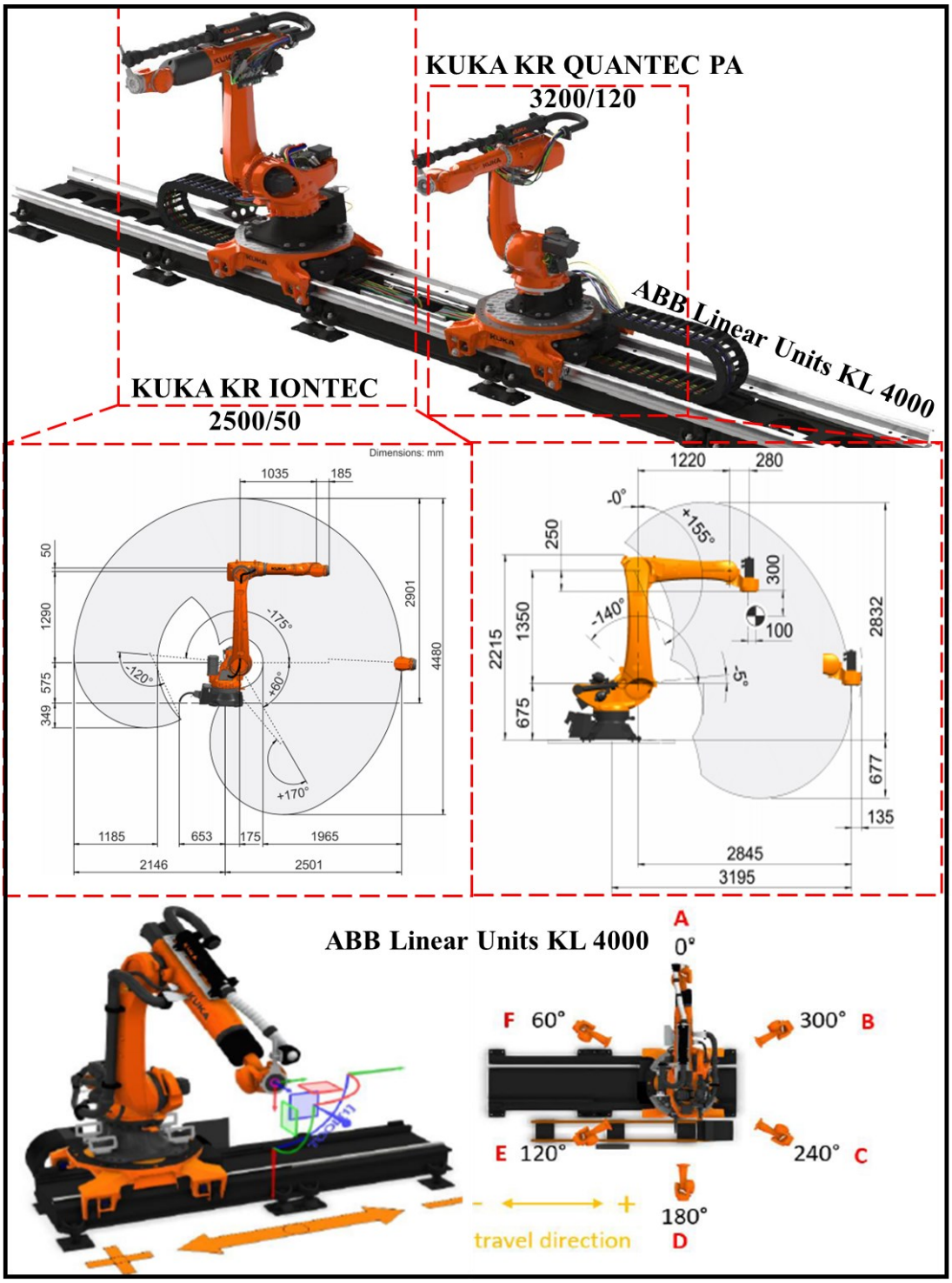


Figure 127. Robotic Configurations

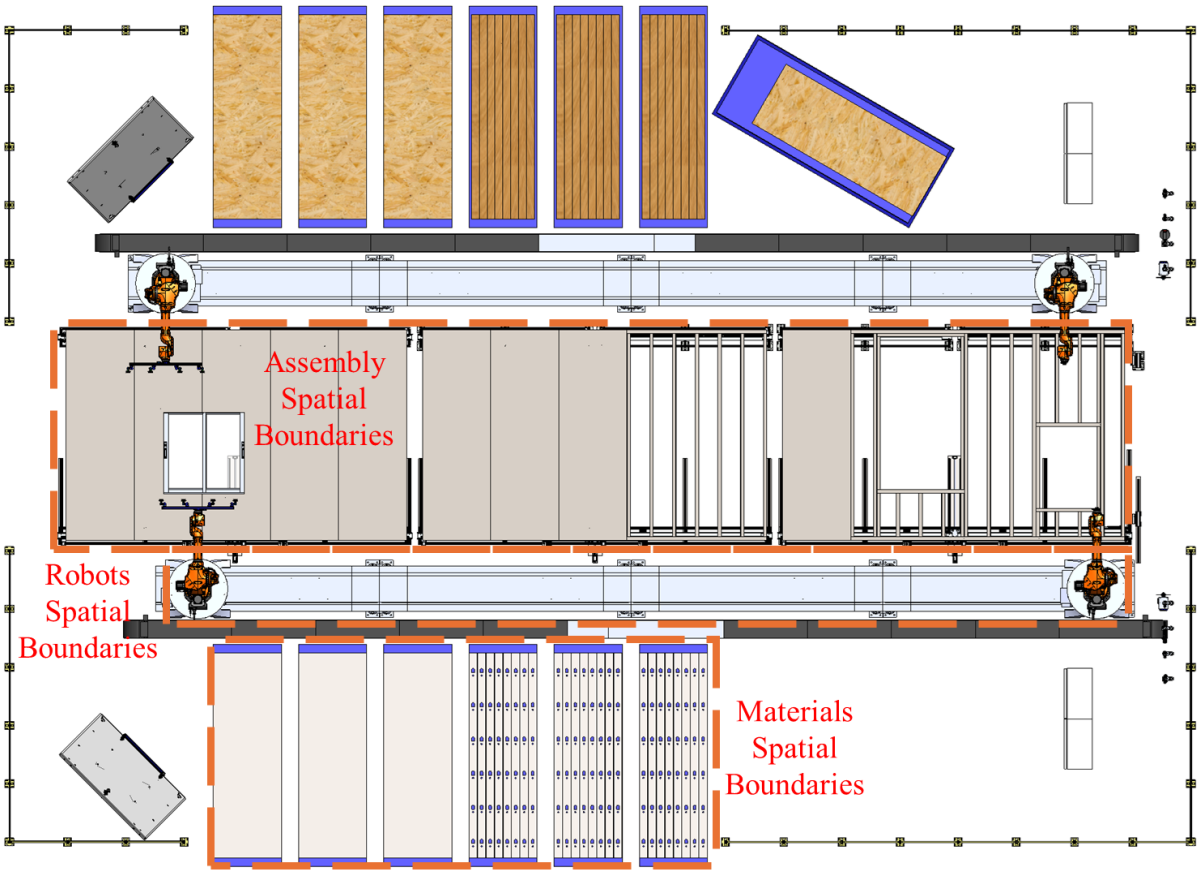


Figure 128. Spatial Boundaries for Robots, Assembly, and Materials

In the area of joint-level trajectory formulation, this method focuses on crafting precise articulation movements for the IR arm to facilitate the execution of a variety of tasks. Specifically, during the assembly of a wall panel, two positions are identified where collision avoidance becomes imminent:

1. **Position 1**: When lifting a pre-constructed element from the material storage zone (as shown in Figure 129), the second joint of the arm commences a counter-clockwise rotation, leading the end-effector (marked as point A) to move from the material storage to the assembly area. To transition the element to its assembly point, the arm undergoes a linear and rotational movement along its x-axis and y-axis. This action could potentially position

the element in a state (shown as the A-B) where it nears the assembly table boundary, raising a collision threat. To proactively counter this, the trajectory planning algorithm retracts the end-effector from its starting location at a distance labelled as d_{AB} to a more secure location indicated as $d_{A'B}$. Post-retraction, the end-effector at position A' is situated within a safe passage positioned between the material picking zone and the assembly table. Following this, a clockwise rotation aligns the element along the x-axis, enabling further x-axis translation while taking into account any frontal collision risks with the linear rail.

2. **Position 2**: As the end-effector A aligns with the y-axis coordinate E, the assembly advances with a movement towards E along the x-axis. This action guides the end-effector and the attached construction component into the panel's interior, positioning the component in a state that might clash with the panel's internal vertical studs. To overcome potential collisions, the trajectory planning strategy implements a clockwise rotation of the end-effector to arrange the component along the z-axis, effectively dodging any vertical stud interference. The end-effector A then proceeds with movements within the O_{xy} plane to achieve the component's placement at the intended E coordinate. The main challenge here lies in orchestrating the robot's maneuvers, focusing on both the end-effector's location and its orientation, with the ultimate aim of confining the component's geometry within an unobstructed area between the panel's upper and lower tracks as shown in Figure 130.

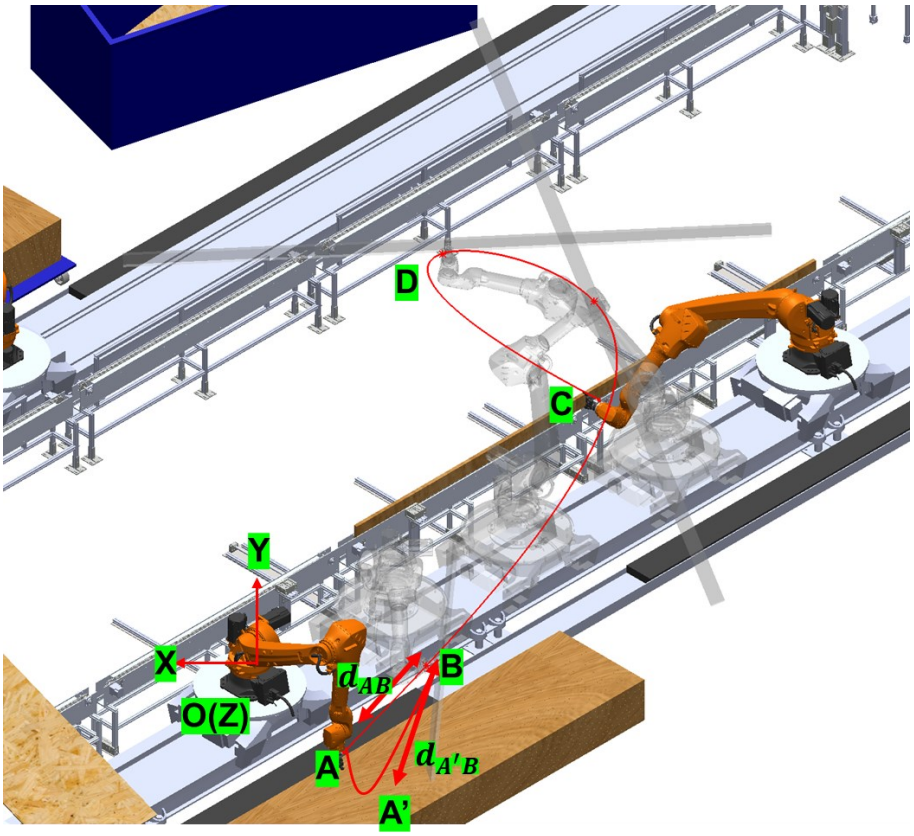


Figure 129. State #1 Collision Avoidance

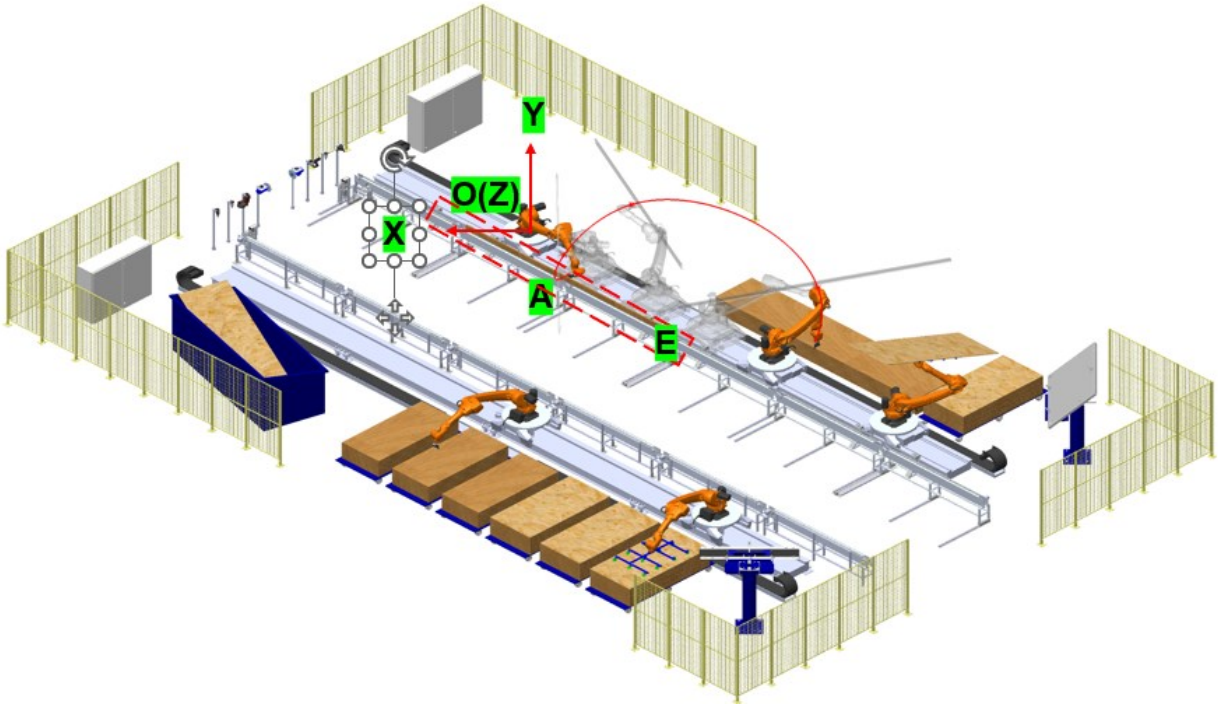


Figure 130. State #2 Collision Avoidance

7.4.2 RRT* Algorithm Formulation

To address the RRT* algorithm's challenges with collision detection involving rigid body constraints in specific states, an advanced modification has been integrated, involving the application of Minkowski Difference calculations. This technique is employed to ascertain the intersection of two distinct shapes, labeled A and B, by utilizing the symmetric difference function ($A \ominus B = \{a - b | a \in A, b \in B\}$), where 'a' and 'b' signify arbitrary points within the geometric representations of shapes A and B, respectively (Yang et al., 2021). The detection of common points suggests the occurrence of a collision, indicated by the existence of a '0' within the set A⊖B. The foundational principles of this algorithm are explained in Algorithm 1. Let *TC* represent the configuration task space of trajectory planning, with TC_{obs} denoting the space occupied by obstacles. The collision-free domain, termed TC_{free}, is derived as $TC_{free} = TC - TC_{obs}$. The StartTree function initializes an empty set, S, which is subsequently populated with vertices to outline a trajectory for the manipulator's endpoint. The Insert function introduces new node as the initial state $TC_{start}$of each pre-manufactured building element denoted as (*ele*), intended for assembly, as the foundational vertex of V, encompassing the component's spatial and orientational attributes, including those of the endpoint. The Sampling function is tasked with generating a random state TC_{rand} within TC_{free}, adhering to a uniform distribution, implying an equal probability for TC_{rand} to occupy any given location within TC_{free}. The Expand function then promotes the expansion of the trajectory tree V, primarily towards the randomly generated state TC_{rand}. Upon identification of the state $TC_{nearest}$ within V that is closest to TC_{rand} in TC_{free}, the Expand function is invoked to create a new state TC_{new} along the path from $TC_{nearest}$ to TC_{rand}, incorporating a time increment Δt in the state evolution of the manipulator. The Close function

compiles a set of states, TC_{close}, emerging from V and located within a spherical vicinity centered around TC_{new}, with a radius 'r'. The SelectMin function evaluates the cost of the trajectory from TC_{start} to TC_{new}, considering each state in TC_{close}, and identifies the state TC_{min} with the minimum cost.

The IF function critically evaluates whether the manipulated element, *'ele'*, and various entities within the environment intersect along the edge connecting 'TC_{new} and TC_{min} states, aiming to verify if *'ele'* is situated within TC_{free}. This determination relies on the collaborative operation of three algorithmic modules within the enhanced RRT* algorithm, functioning as follows:

1. **Dimensional Module**: This module is tasked with cataloging and organizing the geometric models of objects within the construction environment, distinguishing between 'active colliders' (the currently manipulated component *'ele'*) and 'passive colliders' (other environmental entities like the robotic arm's segments, terrain, and previously assembled components). 'Active colliders' actively engage in collision detection, whereas 'passive colliders' remain static, awaiting potential collisions as shown in Figure 131.

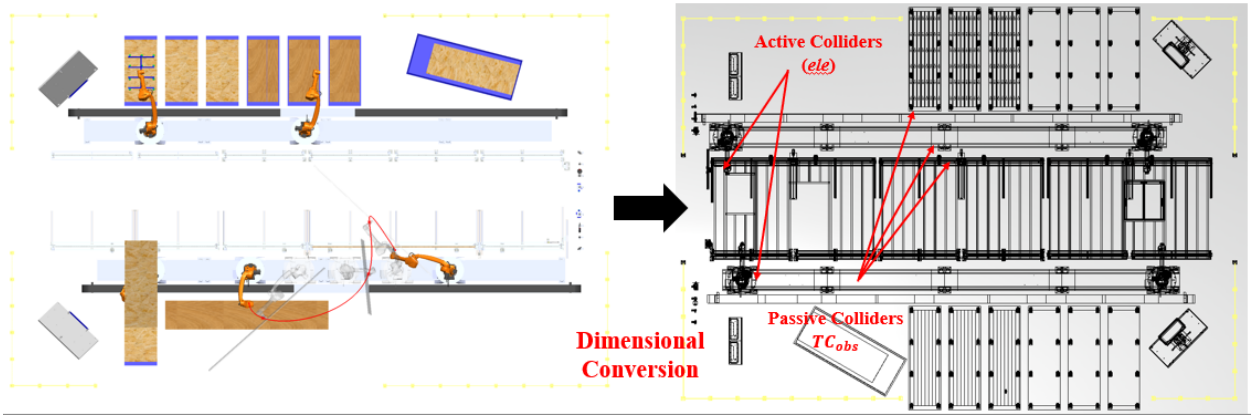


Figure 131. Dimensional Conversion with Active and Passive Colliders Representation.

2. **Representation Module:** Post classification, this module models the objects as polyhedral entities composed of polygonal faces, indexing each face's vertices and normals, and translating them into the global coordinate system. This yields algebraic representations for both active and passive colliders, facilitating subsequent analyses. The IfcIndexedPolygonalFace is a compact representation of a planar face that is part of a face set. The vertices of this polygonal planar face are specified by three or more Cartesian points, which are defined by indices pointing to an IfcCartesianPointList3D, either directly or through the PnIndex, if available in the IfcPolygonalFaceSet. The IfcPolygonalFaceSet is a tessellated face set where all faces are bounded by polygons. These planar faces are formed by implicit polylines defined by three or more Cartesian points. Each planar face is represented by an instance of IfcIndexedPolygonalFace or, in the case of faces with inner loops, by IfcIndexedPolygonalFaceWithVoids. As illustrated in Figure 132, depending on the value of the attribute Closed, the instance of IfcPolygonalFaceSet represents:

    a. if TRUE, a boundary representation (or B-rep);
    b. if FALSE, a face-based surface representation.

The attribute Faces refers to a list of IfcIndexedPolygonalFace elements, each having a one-based CoordIndex that provides three or more indices. If the face has inner loops, the IfcIndexedPolygonalFaceWithVoids contains a two-dimensional, one-based list, where:

    a. the first dimension addresses the list of inner loops;
    b. the second dimension provides three or more indices, each representing a vertex of the planar polygon for the inner loop as shown in Figure 135.

Depending on the presence of PnIndex, the indices either directly reference the IfcCartesianPointList3D specified by the Coordinates in the supertype IfcTessellatedFaceSet, or they refer to the PnIndex, where the integer values at that position indicate the location of the coordinate values within the IfcCartesianPointList3D, as shown in Figure 133. The IfcFacetedFaceSet is defined based on the indexedFaceSet in ISO/IEC 19775. Figure 134 provides an example of this concept.

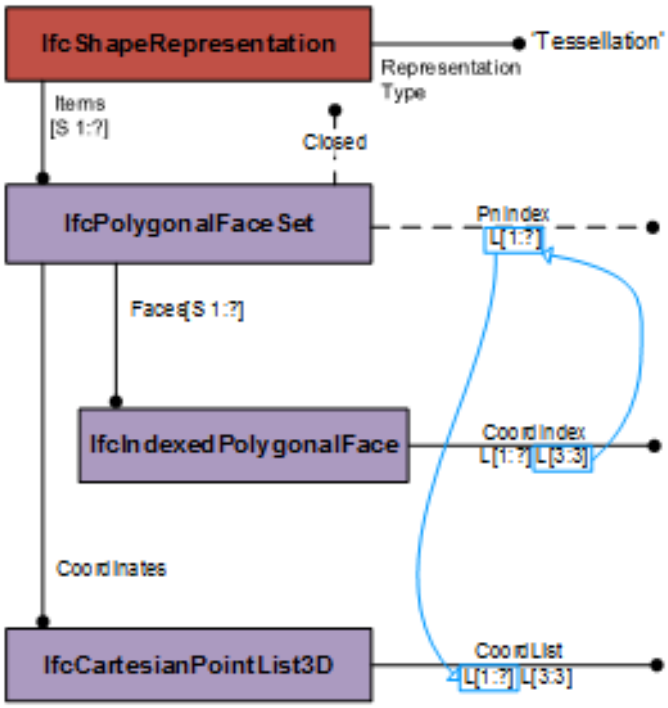


Figure 132. Indices for Coordinates with PnIndex

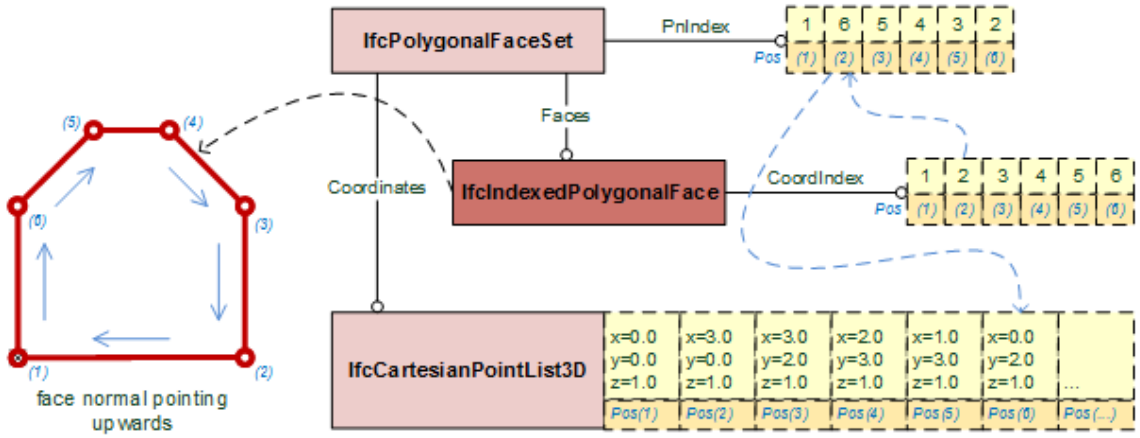


Figure 133. Polygonal Face Geometry (Indices Referencing a Point List)

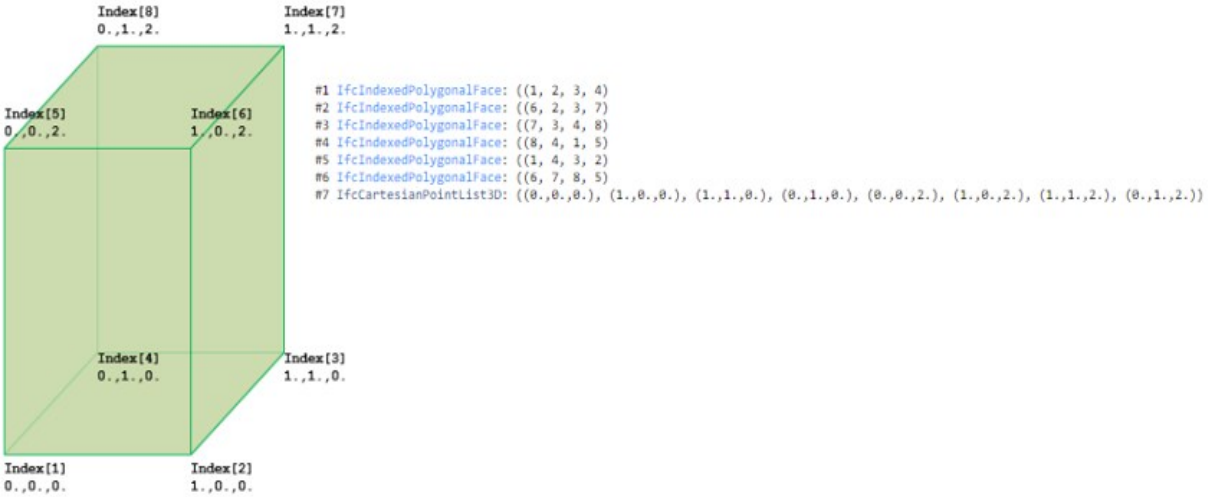


Figure 134. Stud Polygonal Face Set Geometry with IfcFacetedFaceSet Representation

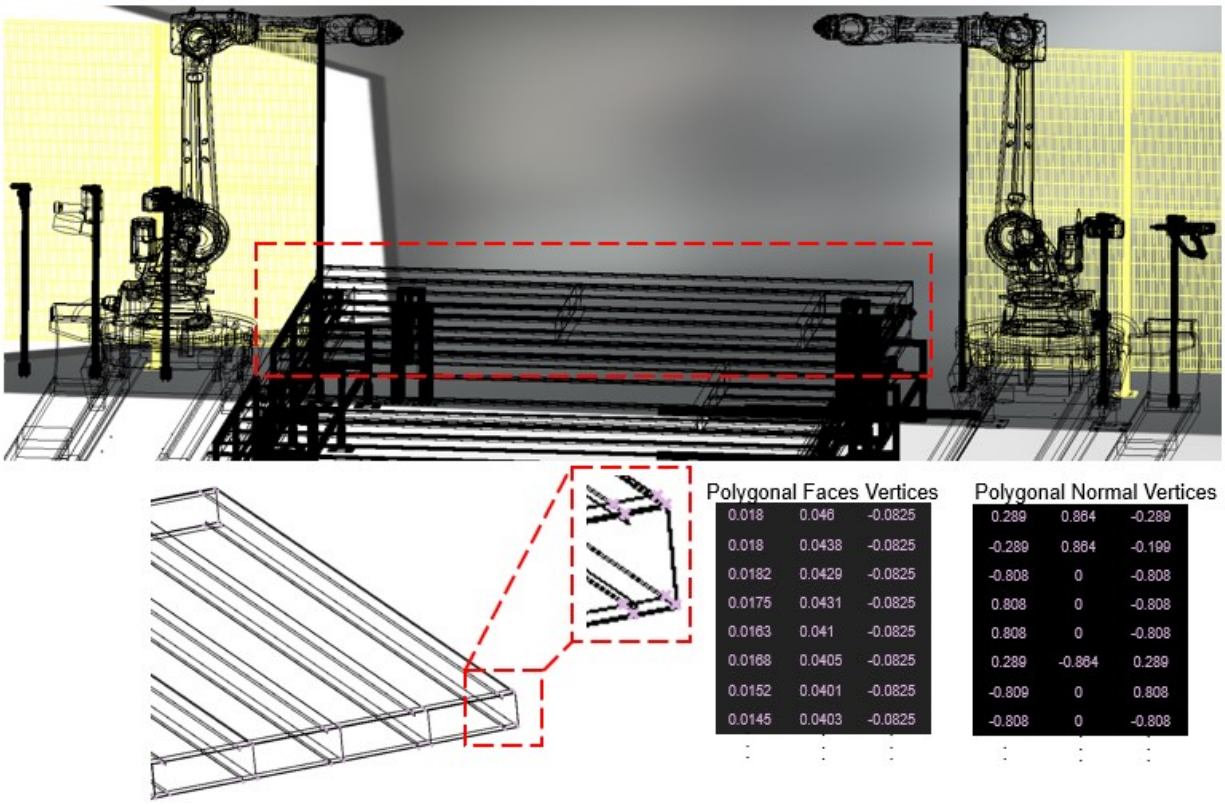


Figure 135. Constructing Polygonal Faces: Normals, Vertex Normals, and Algebraic Collider Representation

3. **Minkowski Difference Module**: Tasked with identifying shared points between active and passive colliders, this module applies the symmetric difference function on their algebraic representations. A zero outcome from the Minkowski Difference calculation $ele \ominus TC_{obs}$ indicates collision by revealing common points. The Minkowski Difference module is responsible for assessing whether there are shared points between the active and passive colliders. The algebraic representations of these colliders are modelled using the symmetric difference function. Specifically, it checks for common points by evaluating the symmetric difference ($ele \ominus TC_{obs} = (\{a - b \mid a \in ele, b \in TC_{obs}\})$. If the result of the Minkowski Difference ($ele \ominus TC_{obs}$) equals zero, it signifies the presence of shared points, indicating a collision event.

```
public List<Vector2D> MinkowskiDifference(Polygon other)
{
  List<Vector2D> vList = new List<Vector2D>();

  foreach (Vector2D vT in this.vertices)
  {
     foreach (Vector2D vO in other.vertices)
     {
        vList.Add((vT+this.position) - (vO+other.position));
     }
  }
  return vList;
}

public static Polygon ConvexHull(List<Vector2D> vList)
{
   List<Vector2D> S = vList;
   List<Vector2D> P = new List<Vector2D>();
   Vector2D endpoint;

   Vector2D pointOnHull = LeftMostPoint(S);
   int i = 0;
   do
   {
      P.Add(pointOnHull);
      endpoint = S[0];
      for (int j = 1; j < S.Count; j++)
      {
        if (endpoint == pointOnHull || RelPosition(P[i], endpoint, S[j]) == -1)
        {
             endpoint = S[j];
        }
      }
      i++;
      pointOnHull = endpoint;
   } while (endpoint != P[0]);
   return new Polygon(P);
}
```

Figure 136. Minkowski Difference with Concave Polygons Partial Code

Upon the fulfilment of the conditions set forth by the IF clause, the following steps are executed where the Parent function assignsTC_{new} the role of the parent node for TC_{min} after that the line function incorporates the segment $TC_{new} \leftarrow TC_{min}$into the trajectory framework F.

To augment the fluidity of the trajectory, the Kinodynamic Relocating mechanism is activated. This process is aimed at rectifying instances of sudden directional shifts and velocity alterations within the plotted trajectory. It accomplishes this by distributing intermediary points within the unobstructed domain TC_{free} positioned in proximity to the trajectory's segments. Subsequently, it refines these segments by formulating a parametric pathway that interconnects all intermediary points and adjacent states within the trajectory structure *F*.

It is imperative to acknowledge that the refined RRT* algorithm is designed to function in a repetitive manner. Following the creation of an assembly path for a construction element, the spatial configuration of said element at its designated terminal location, represented as ele. Geometry. TC_{goal} is merged into TC_{obs} . This integration aids in the trajectory formulation for subsequent construction elements. Finally, the InverseKinematics routine calculates the articulation movements $[Joint\ 1(m), Joint\ 2(m), \ldots . Joint\ 7\ (m)]$that adhere to the kinematic restrictions imposed on each articulation within the robotic arm. These movements facilitate the success of the planned end-effector positions over the time motion sequence *F(m)*.

During the assembly planning stage, the positions for constructing individual architectural elements are established in advance. However, it is crucial to recognize a significant variation: the perception of obstacles within the work area is predominantly maintained by human operators monitoring the system, rather than by the automated robotic unit. This distinction stems from the incremental emergence of obstructions within the workspace as assembly activities progress. For example, a structural component that has been assembled earlier, like a stud, might become an obstacle for the installation of subsequent elements, such as a window component. As a result, the configuration of obstructive elements within the robot's operational spatial boundaries is subject

to continuous and unforeseen modifications over time. To navigate this variability, Algorithm 1 is designed to function in a continuous online mode, incorporating essential features such as:

1. **Continuous Geometric Model Update**: Algorithm 1 undertakes a repetitive procedure to constantly evaluate and catalogue the geometric configurations of emerging obstructions in real time.
2. **Kinematic Parameter Synthesis**: It computes the requisite kinematic variables to facilitate the robotic arms in performing assembly operations without encountering collisions.
3. **Immediate Dissemination of Kinematic Information**: The algorithm swiftly distributes the calculated kinematic data to the robotic arms, providing immediate guidance for their movements.

Additionally, Algorithm 1 employs a consistent reference point for the location of the assembly table, material storage, and tool changing station, which acts as the predefined starting condition for each building element within the path planning phase.

During the assembly planning stage, the positions for constructing individual architectural elements are established in advance. However, it is crucial to recognize a significant variation: the perception of obstacles within the work area is predominantly maintained by human operators monitoring the system, rather than by the automated robotic unit. This distinction stems from the incremental emergence of obstructions within the workspace as assembly activities progress. For example, a structural component that has been assembled earlier, like a stud, might become an obstacle for the installation of subsequent elements, such as a window component.

Algorithm 1: Enhanced RRT*

F← StartTree();

F← Insert(TC_{start})

J ←Ø;

$f = 0$;

for i=1 →N do

  $TC_{rand} \leftarrow$ Sample (TC_{free}, i);

  $TC_{close} \leftarrow$ Closest (F, TC_{rand});

  $TC_{new} \leftarrow$ Expand $(TC_{close}, TC_{rand}, \Delta_m)$;

  $TC_{close} \leftarrow$ Close (F, TC_{new}, r);

  $TC_{min} \leftarrow$ SelectMin $(TC_{new}, TC_{close}, TC_{start})$;

  if $0 \notin ele \ominus TC_{obs} = (\{a - b \mid a \in ele, b \in TC_{obs}\}, TC_{new} \leftarrow TC_{min})$ then

    F ← Insert(TC_{new}, TC_{min}, F);

    Merge (TC_{new}, TC_{min});

    F ← Line(c, TC_{min});

    Kindoynamic Relocating (F, TC_{new}, TC_{min});

  end if;

  if $|| TC_{new} - TC_{min} ||$ < d then

    $TC_{obs} \leftarrow$ ele. Geometry. TC_{goal};

    Break;

  end if;

end for

return F;

do

  $joint_1(m), joint_2(m), joint_3(m), joint_4(m), joint_5(m), joint_6(m), joint_7(m) \leftarrow$ F(t). InverseKinematics;

  $J \leftarrow [joint_1(m), joint_2(m), joint_3(m), joint_4(m), joint_5(m), joint_6(m), joint_7(m)]$;

  $m = m + \Delta_m$;

while $|| F(m) - TC_{goal} ||$ > eps

end do-whole

return J;

As a result, the configuration of obstructive elements within the robot's operational spatial boundaries is subject to continuous and unforeseen modifications over time. To navigate this variability, Algorithm 1 is designed to function in a continuous online mode, incorporating essential features such as:

4. **Continuous Geometric Model Update**: Algorithm 1 undertakes a repetitive procedure to constantly evaluate and catalogue the geometric configurations of emerging obstructions in real time.

5. **Kinematic Parameter Synthesis**: It computes the requisite kinematic variables to facilitate the robotic arms in performing assembly operations without encountering collisions.

6. **Immediate Dissemination of Kinematic Information**: The algorithm swiftly distributes the calculated kinematic data to the robotic arms, providing immediate guidance for their movements.

Additionally, Algorithm 1 employs a consistent reference point for the location of the assembly table, material storage, and tool changing station, which acts as the predefined starting condition for each building element within the path planning phase.

7.4.3 Random Configuration Sampling Stage

In the subsequent stage, the aim is to stochastically synthesize robotic configurations within the simulated environment, utilizing the robot arm's point-to-point (PTP) planning capability to randomly establish configurations along a predefined trajectory. The PTP planner excels in identifying the most efficient route to a designated terminal position by evaluating the angular differences and joint velocities between the initial and concluding positions. However, it is crucial to acknowledge that the paths deduced by the PTP planner might not be free of collision risks due to its primary focus on kinematic parameters rather than interactive environmental elements.

To refine this approach, the method introduced here employs a targeted sub-global sampling technique, which is informed by several critical factors including the dimensions of the object in focus, the spatial distance separating the initial and final positions of the object, and the variance in configurations from start to finish. The initial step involves segmenting the original PTP trajectory into N_{Traj} distinct intervals, each characterized by a uniform configuration trajectory expressed as$\frac{L^{(n)}}{N_{Traj}} x\ L^{(n)}$. The term $L^{(n)}$ signifies the total configuration length bridging the initial and destination configurations, with its calculation presented in Equation (1).

$$L^{(n)} = \sqrt{(\theta_{start-1} - \theta_{goal-1})^2 + (\theta_{start-2} - \theta_{goal-2})^2 + \cdots + (\theta_{start-n} - \theta_{goal-n})^2} \quad (1)$$

In this context, n represents the number of DOFs associated with the IR arm. The determination of the start and goal configurations leverages an IK solver, a tool commonly embedded within commercial robotic simulation software. This solver is adept at calculating the requisite joint angles by examining the spatial positioning of the end-effector within the cartesian coordinate framework.

Following this, for each point, labeled as N_P, a collection of configurations is randomly synthesized, each lying within a specified configuration radius denoted by $r^{(n)}$. The formulation and operational principles are detailed in Equation (2).

$$r^{(n)} \leq \begin{cases} \frac{r}{R^{(3)}} L^{(n)}, if\ \frac{r}{R^{(3)}} L^{(n)} \leq L_{max}^{(n)} \\ L_{max}^{(n)}, otherwise \end{cases} \quad (2)$$

In this framework, the variable *r* denotes the maximal dimension of the object being manipulated, proportional to the length of a wood stud. Within the domain of construction framing, it is common

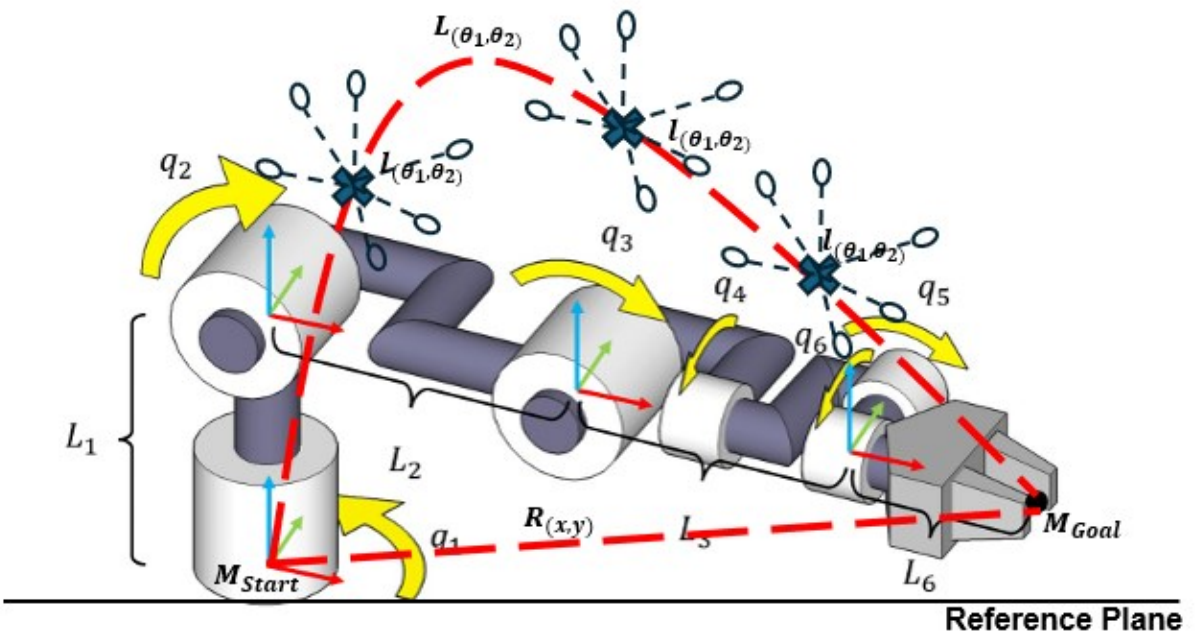
for wood elements to share uniform cross-sectional profiles (for instance, 2”x4” or “2x6” or 2”x8”), yet their longitudinal extents can vary significantly. Accordingly, the proposed methodology could accommodate the variability in stud lengths by generating random configurations that reflect the length of the stud in question.

The notation $R^{(3)}$ quantifies the trajectory length traversed by the object in transit from its starting point to its goal endpoint, a value derivable from Equation (3). Conversely, $L_{max}^{(n)}$ is established as a predefined limit that regulates the maximal displacement permissible for the robotic arm's movement. The expression $\frac{r}{R^{(3)}}$ underscores the reciprocal influence of the object's length and the displacement between the starting and goal points on the search domain's dimensions. Specifically, an elongated object might necessitate a larger search perimeter to ascertain pathways free of obstructions, and conversely, a condensed distance between the start and goal points might also increase the search area requirements.

$$R^{(3)} = \sqrt{(x_{start} - x_{goal})^2 + (y_{start} - y_{goal})^2 + (z_{start} - z_{goal})^2} \quad (3)$$

Upon the completion of the random generation of configurations, these configurations are systematically arranged and grouped within a matrix, referred to as *M*, as explained by Equation (4). This matrix *M* is then forwarded to the path-searching segment of the process. The primary aim within this phase is to pinpoint the most direct path free of obstructions, utilizing the configurations sampled, which are embedded within matrix *M*.

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1(N_p+1)} \\ \vdots & \vdots & \ddots & \vdots \\ M_{(N_{Traj}-1)1} & M_{(N_{Traj}-1)2} & \cdots & M_{(N_{Traj}-1)(N_p+1)} \end{bmatrix} \quad (4)$$



Δ *is refered to as nodes on the original PTP trajectory*

O *is referred to as the randomly generated configurations for each node, where*

$$l_{(\theta_1,\theta_2)} \leq \frac{r}{R_{(x,y)}} \times L_{(\theta_1,\theta_2)}$$

Figure 137. Example of the proposed sampling method with a 3-DOF robot arm in a 2D space

Figure 137 showcases the configuration sampling method applied to a robot arm possessing a 2-DOF operating within a 2D framework. Where, M_{start} and M_{goal} are referred to as the robot arm's starting and goal configurations, respectively. The configuration distance, expressed as $L_{(\theta_1,\theta_2)}$ is calculated by evaluating the angular deviations Δ_{θ_1} and Δ_{θ_2} between the start and goal configurations. Furthermore, $R_{(x,y)}$ quantifies the Euclidean distance bridging the start and goal locations within the 2D plane. With N_{Traj} set to three, the methodology entails the generation of two intermediary waypoints along the plotted PTP trajectory. Given N_{Traj} is six, the process involves the random generation of six configurations proximal to each waypoint, with each configuration maintaining a distance less than $l_{(\theta_1,\theta_2)}$ from the waypoint. As a result, this particular

example results in the aggregation of 27 configurations into a matrix, which subsequently undergoes analysis during the path-searching stage.

7.4.4 Synthesizing Kinematic Parameters with Integrated Obstacle Avoidance

Within the scope of this research, a prototype robotic system was engineered for the assembly of OSC prefabricated components, integrating Dynamo for initial data provisioning and the ROS for operational execution. Dynamo was instrumental in the preprocessing phase, facilitating the extraction of pivotal assembly data from the architectural Revit model of a prefabricated house by utilizing the RAP. A specialized RAP algorithm, developed within Dynamo's environment, was tasked with pinpointing the precise assembly coordinates for the building elements, which it then utilized to deduce a logical sequence for the assembly process based on spatial coordinates.

This sequence, essential for structuring the robotic assembly strategy, was then fed into ROS, where a sophisticated motion planning algorithm took over. This algorithm processed the assembly sequence to ascertain the requisite kinematic configurations, encompassing both the articulation points and the trajectory paths essential for the robotic entities to efficiently assemble the given wall panel. To ensure a smooth exchange of information between the RAP component in Dynamo and the RMP framework in ROS, an interfacing protocol was established, optimizing the coordination between data input and robotic execution phases.

The kinematic formulation is typical within the domain of robotic trajectory optimization, acting as the foundational computational mechanism for deducing the required articulation parameters that facilitate the generation of a predetermined end-effector attached to a manipulator, as explicated by Nahangi et al. (2015). Coordinate frames were assigned to the articulations for the derivation of the kinematic schema relevant to the TCP utilized in this study. These coordinate

frames, as illustrated in Figure 138, are instrumental in formulating the motion dynamics inherent to each articulation.

It is also critical to acknowledge that the allocation of these coordinate frames adhered to the universally recognized right-hand rule for coordinate system orientation, a methodology that has been rigorously detailed in preceding studies by Zhang et al. (2019) and Renuka et al. (2021). This convention ensures consistency and standardization in the spatial orientation and analysis of joint movements within the field of robotics.

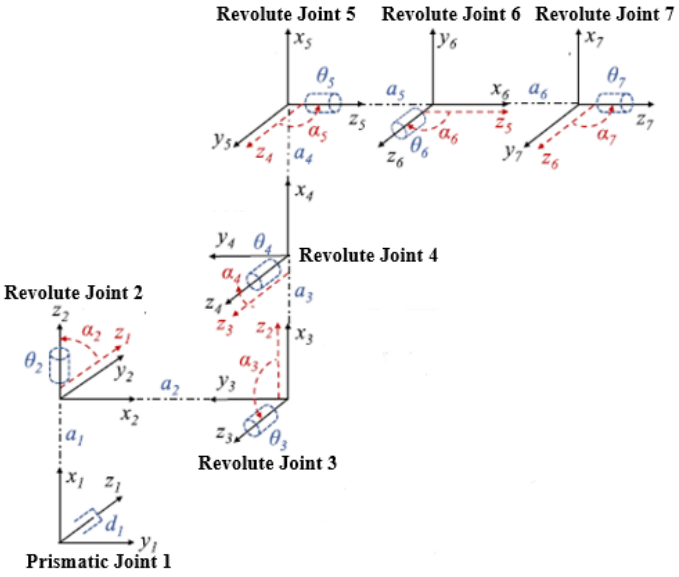


Figure 138. Robotic TCP Reference Frames

The kinematic relationship, specifically the pose association describing the spatial orientation and locational disposition between two adjacent articulations, is defined via the implementation of frame-anchored Denavit-Hartenberg (D-H) convention as explained by Alam et al. (2020). The D-H schematic expresses a set of transformational coefficients, d_i(translation distance along z-axis of the ith joint), θ_i (rotation angle around z-axis of the ith joint), a_i(link length along x-axis between

the i[th] and i[th]+1 joints), and α_i (angler twist around z-axis between the i[th] -1 and i[th] joints) as shown in Figure 138.

Relevantly, in the context of a 7-DOF manipulator, the variables d_i and θ_i exhibit variability concurring with joint dynamics, whereas a_i and α_i remain unchanging, reflecting the manipulator's structural geometry (e.g., link dimensions) as shown in Table 27. The sequential combination of transformations of the variables across a sequence of frames leads to the kinematic equation *T*, derivable through the product integration of the homogeneous transformation matrices corresponding to d_i, θ_i, a_i, and α_i as represented in Equation (5).

$$T = \prod_{i=1}^{7} \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & \cos(\theta_i)a_i \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & \sin(\theta_i)a_i \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

Table 27. D–H Parameters of 7 DOF Robotic TCP

i[th] Frame (Joint)	$\boldsymbol{d_i}$ **(m)**	$\boldsymbol{a_i}$ **(m)**	$\boldsymbol{\theta_i}$ **(rad)**	$\boldsymbol{\alpha_i}$ **(rad)**
# 1 (Prismatic)	d_1	$a_1 = 2.433$	0	0
# 2 (Revolute)	0	$a_2 = 0.458$	θ_2	$\alpha_2 = 2.570$
# 3 (Revolute)	0	$a_3 = 2.470$	θ_3	$\alpha_3 = 2.570$
# 4 (Revolute)	0	$a_4 = 0.215$	θ_4	$\alpha_4 = 2.570$
# 5 (Revolute)	0	$a_5 = 2.420$	θ_5	$\alpha_5 = 2.570$
# 6 (Revolute)	0	$a_6 = 0.408$	θ_6	$\alpha_6 = 2.570$
# 7 (Revolute)	0	0	θ_7	$\alpha_7 = 2.570$

The revised enhanced RRT* algorithm is shown in Algorithm 2 and explained as follows:

1. Initially, the function Sample trajectory configuration TC_{free} which produces a stochastic state TC_{rand} within the collision-free domain TC_{free}, utilizing a uniform probability distribution. This implies that $TC_{rand}the$ potential location has an equitable likelihood of being anywhere within TC_{free}.
2. Subsequently, an assessment is made between TC_{rand} and the extant states within the node set V to identify the state TC_{close} that is closest to TC_{rand}.
3. The Steer(TC_{close}, TC_{rand}, Δ_m) function then creates a new state TC_{new}, positioned nearer to TC_{close} by employing a steering mechanism to bridge TC_{rand} and TC_{close}.
4. The CollisionFree(TC_{close}, TC_{new}, TC_{obs}) function dictate the absence of any obstructions along the direct trajectory connecting TC_{new} and TC_{close} within the obstacle space TC_{obs}.
5. If the path is clear of obstacles, the Near(V, TC_{new}, r) function aggregates a subset of states within a spherical region centered at TC_{new} with radius r.
6. The Line(TC_{close}, TC_{new}, s) function establishes a linear connection between TC_{new} and TC_{close}, where the line's length is set by the parameter step size s.
7. The Parent(TC_{new}, TC_{close}, F) function designates the state from TC_{close} with the minimal cost-to-go as the parent state TC_{min}.
8. Subsequently, TC_{new} is incorporated into the node set V, and the new linkage between $TC_{min}$and TC_{new} is added to the edge set E.
9. The Rewire(G, TC_{close}, TC_{new}) function dynamically adjusts the connections between $TC_{new}$and states in TC_{close} to minimize the overall path cost and optimize the trajectory.

10. Ultimately, the algorithm iterates to construct the global graph G(V, E), encapsulating the optimized path θ = [TC_1,⋯ TC_n] after n iterations.
11. RRT* continuously samples random nodes TC_{rand} within TC_{free} and reassesses the global graph G(V, E) by evaluating the prospective cost-to-go for each node $TC \in$ V situated near the newly sampled node TC_{new}.

Algorithm 2

V← TC_{start} , TC_{goal};
E← Ø;
for i=1,2,...., N do
    TC_{rand} ← Sample (TC_{free}, N);
    TC_{close} ← Closest (V, TC_{rand});
    TC_{new} ← Steer $(TC_{close}, TC_{rand}, \Delta_m)$;
    if CollisionFree $(TC_{new}, TC_{close}, TC_{obs})$ *then*
        TC_{close} ← Close (V, TC_{new}, r);
        F ← Line $(TC_{new}, TC_{close}, s)$;
        (TC_{new}, TC_{min}) ← Parent(TC_{new}, TC_{min}, F);
        V ← V ∪ $\{TC_{new}\}$;
        E ← E ∪ $\{TC_{new}, TC_{min}\}$;
        G ← Rewire (G, TC_{new}, TC_{min});
        end;
end
return G(V, E);

7.5 Implementations and Testing of the Prototype

The developed path-finding algorithm was deployed within the developed RoboSimX robotic simulation platform. RoboSimX is engineered to facilitate a comprehensive virtual framework for the design, simulation, and management of robotic operations, integrating advanced FK and IK solvers tailored for robotic manipulators produced by leading global manufacturers.

7.5.1 Modelling of a Robotic Manufacturing Cell in the Virtual Environment

The initial step involves the creation of a digital counterpart for the robotic manufacturing cell within a game engine-driven simulation. This method diverges from traditional approaches by not constructing the Configuration Space (C-space) for the real-world setup. Instead, it utilizes a virtual representation of an IR arm to explore and outline paths clear of obstructions in a simulated environment. As part of the digital twin setup, various components of the robotic manufacturing cell, such as the assembly tables for panel fabrication, conveyance systems like belts for material movement, raw material picking areas, tool changing stations, and even the factory floor, are modelled in three dimensions within the virtual landscape. Figure 139 showcases a digital representation of a robotic manufacturing cell, featuring virtual models of multiple IR arms with different attached gripping and machining tools, an assembly table with conveyor belts for the movement of finished fabricated panels, raw materials (studs and sheets) supply carts, tool exchanging zone, waste bin, electrical control panels, and the factory's base and safety fence terrain. Moreover, the simulation incorporates FK/IK algorithms for the robotic arms embedded within the game engine to aid in the computational derivation of robotic movement paths all integrated in Figure 140.

Upon the creation of the digital twin, it becomes crucial to implement collision elements across all 3D entities within the virtual domain. Within the area of a game engine, collision plays a pivotal role in defining the spatial extents of 3D models. The intersection of collision elements from different objects signals a collision event to the game engine. Game engines typically facilitate the automatic generation of meshed collision elements for imported 3D assets, yet defaulting to auto-generated mesh collision for every object might lead to significant computational strain. The game

engine conducts collision detection by examining collision element intersections on a per-frame basis, which means complex environments with irregularly shaped objects (such as cone-shaped entities) that possess multiple collision elements demand an exhaustive inspection of each frame.

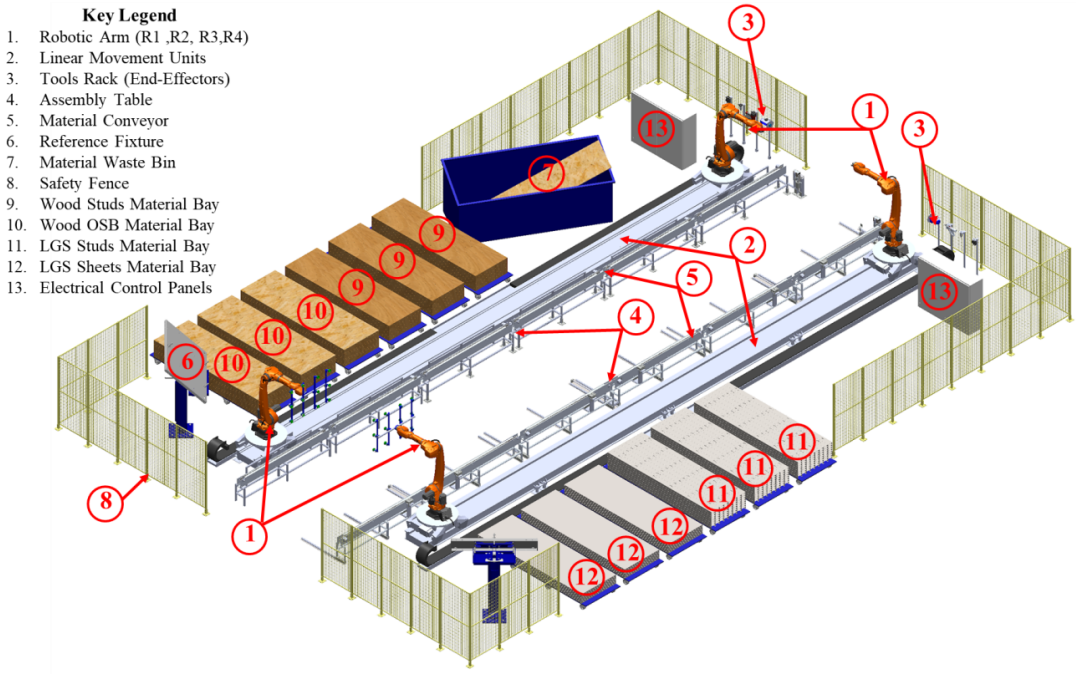


Figure 139. Example of a Robotic Manufacturing Cell in the RoboSimX Virtual Environment

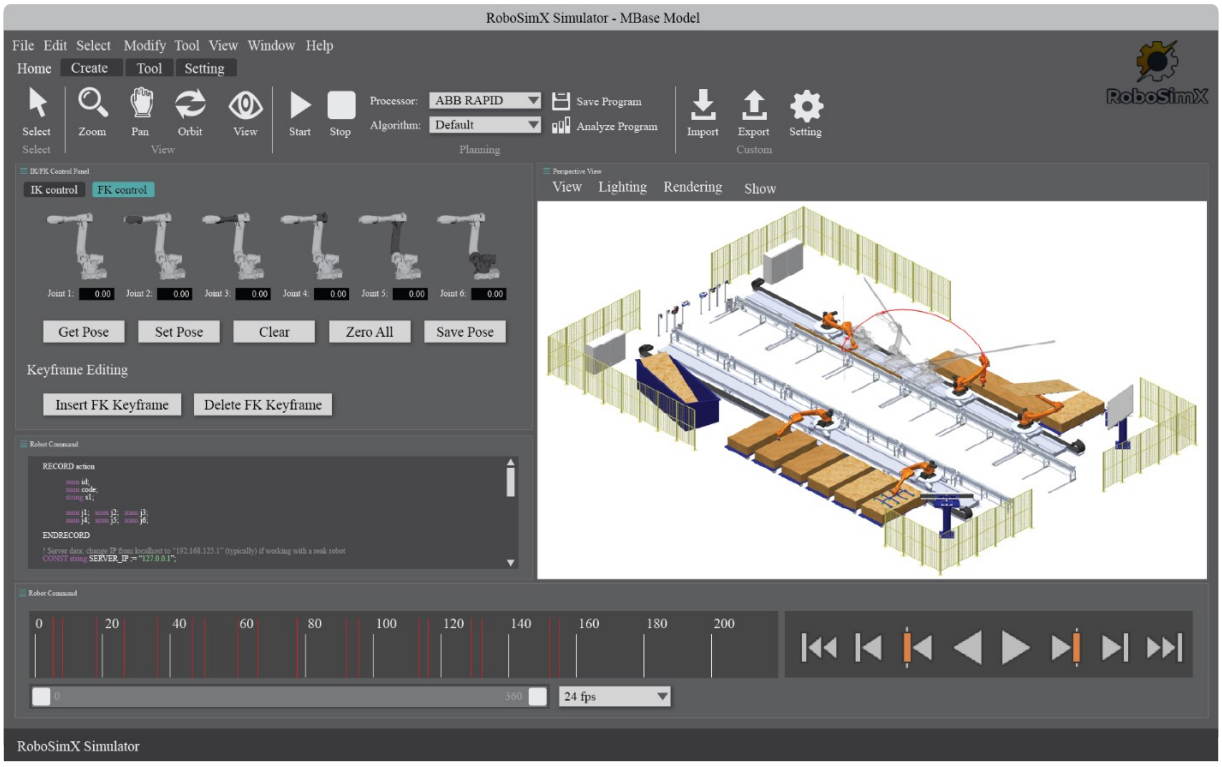


Figure 140. RoboSimX Simulater User Interface

7.5.2 Configurations sampling with gripped object

During the configuration sampling stage, the process necessitates manual specification of three critical parameters: the quantity of discrete segments, the node count per segment, and the ceiling on configuration spacing. The selection of the discrete segment count and node density is important, as an excessive number of configurations can significantly extend the computational duration required for pathfinding. Furthermore, the complexity of the operational context dictates an increased allocation of discrete segments, enhancing the robotic manipulator's ability to navigate a collision-averse trajectory as shown in Figure 141.

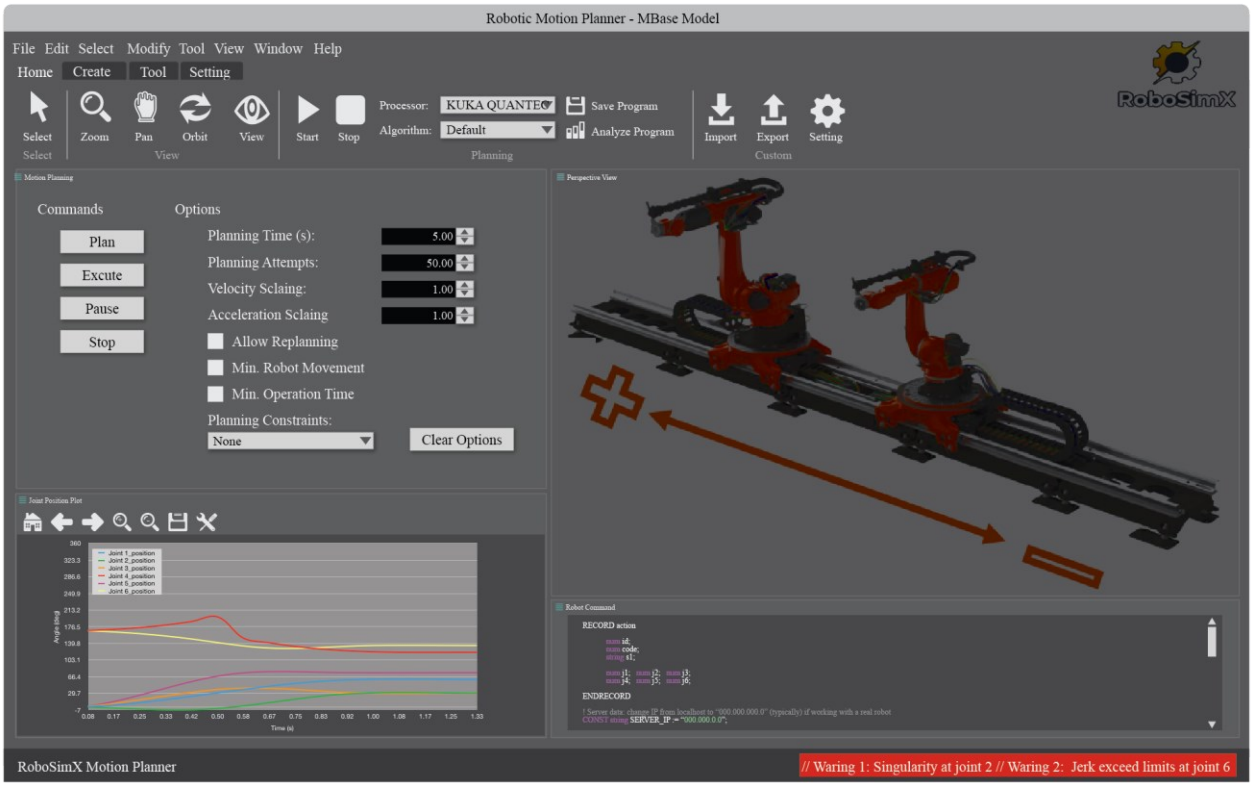


Figure 141. RoboSimX Motion Planner User Interface

Figure 142 presents a visual representation of a sampling scenario within the RoboSimX simulation framework, showcasing an instance where the sampling parameters were set to five discrete segments, with each segment comprising ten nodes, and a maximal configuration interval set at 180 degrees, allocated as 180 degrees per articulation point.

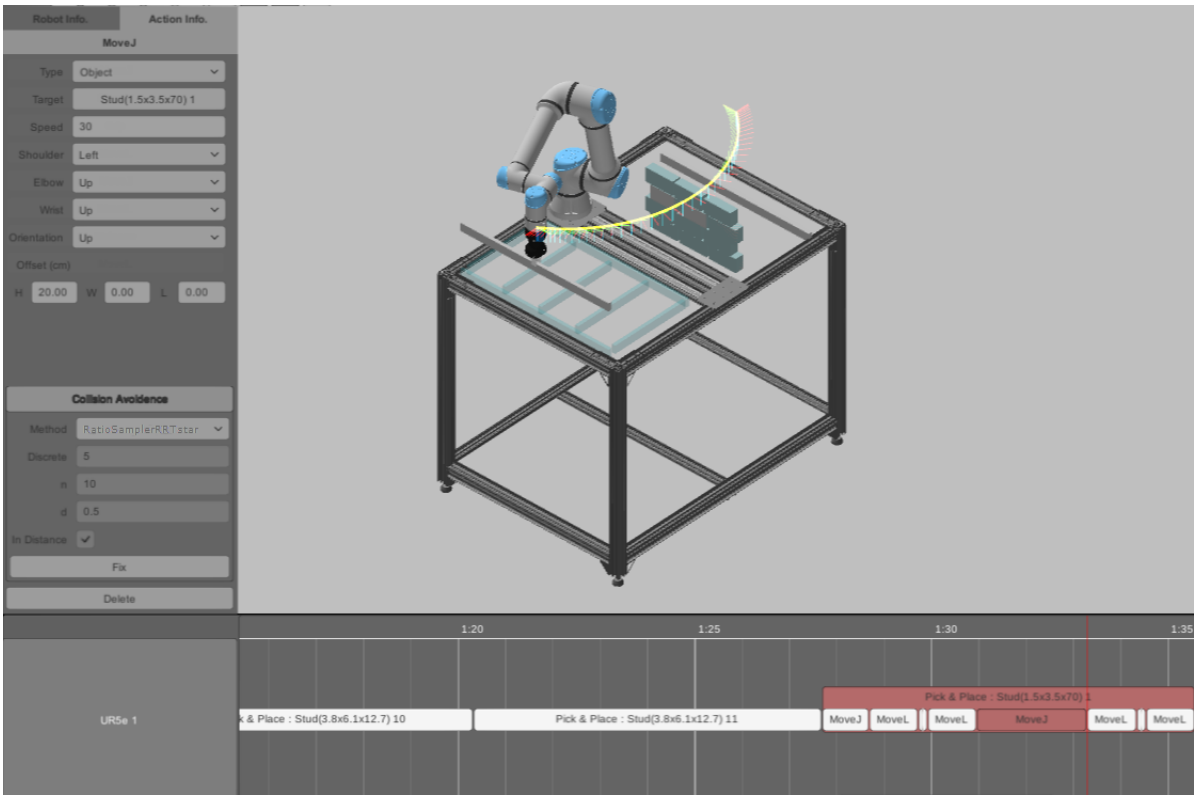


Figure 142. Example of Configuration Sampling in RoboSimX During a Lab Experiment

7.5.3 Path Searching with the Enhanced RRT* Algorithm

The enhanced RRT* algorithm is tasked with identifying a collision-free trajectory through the array of configurations generated during the sampling stage. Within this phase, the assignment of a cost penalty for the evaluation of each configuration's traversal expense is a manual procedure. This penalty must be substantial enough to effectively discern the presence of collisions between configurations, with the outcomes recorded as a sequence of configurations. Subsequently, RoboSimX's FK/IK solvers are employed to construct PTP movements linking these configurations.

With reference to the case study explained below, Figure 143 exhibits the pathfinding outcomes visualized in the RoboSimX environment, where the collision penalty was established at 10,000 seconds. To avoid obstacles, multiple intermediary configurations were strategically chosen as

transitional nodes between the start and goal configurations. The FK/IK solvers within RoboSimX facilitated the integration of these points through the generation of multiple PTP transitions.

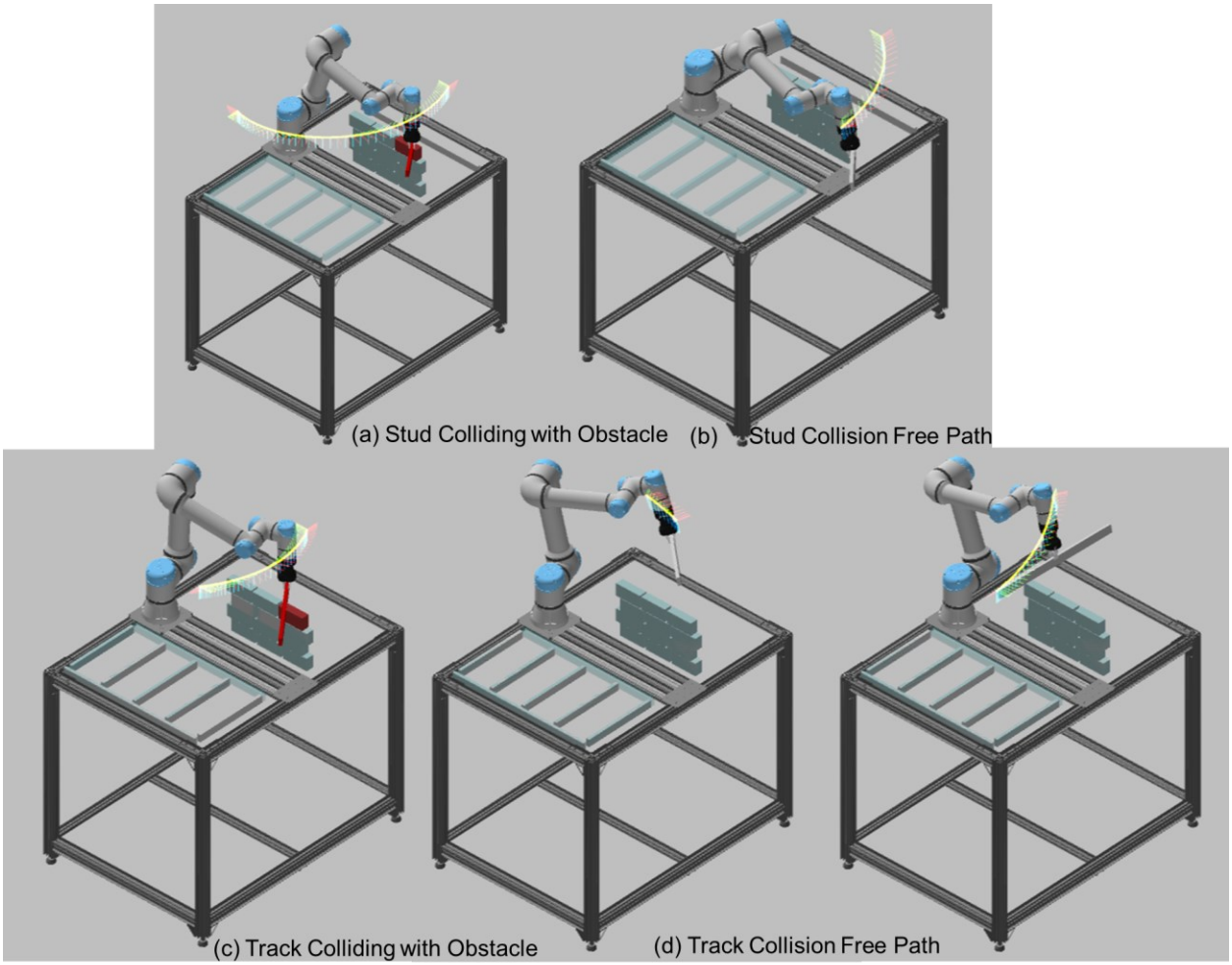


Figure 143. Collision-Free Path with Six New Configurations Generated by the Developed Method

7.6 Validations

This research introduces a novel collision avoidance strategy tailored for the OSC prefabrication sector. The methodology's effectiveness in resolving motion planning challenges inherent to the prefabrication process was initially assessed through simulation trials. These trials aimed to mimic collision-free navigation paths for two multi-wall panels extracted from Base Model.

Subsequently, the feasibility of this approach was further corroborated through a scaled-down experimental evaluation, employing an actual robotic arm to validate the practical applicability of the proposed method.

7.6.1 Planning Effectiveness Confirmation

The proposed path-finding algorithm was implemented within a simulated framework specifically for the frame assembly operation, to ascertain its effectiveness within the context of robotic-assisted OSC prefabrication. The validation process involved the utilization of two distinct variants of wood-framed multi-wall panels, with the objective of determining the algorithm's capability to effectively mitigate collision incidents during the assembly of each panel type. This approach was aimed at demonstrating the algorithm's robustness in addressing collision avoidance challenges across various components involved in the wall panel fabrication process.

7.6.2 Path-Finding Method on Wall Frames Assembly

The robotic assembly line shown in Figure 140 was deployed to evaluate the path-finding methodology's performance. The configuration necessitates setting the count of discrete segments to five and assigning ten nodes per segment, with the provision for engineers to adjust these parameters based on the production line's specific design requirements. The complexity of the operational environment dictates the optimal settings for the number of discrete segments and nodes per segment, with higher values enhancing the robotic arm's ability to navigate collision-free routes at the expense of increased computational time during the pathfinding phase. The threshold for configuration spacing was established at 180 degrees, equating to 180 degrees per joint. Given that the baseline PTP movements for assembling specified construction elements within this setup were under 20 seconds, a collision penalty of 10,000 seconds was instituted.

For the effectiveness assessment, two wood-framed multi-wall panels from a BIM Base Model, each characterized by a unique composition of structural elements, openings, and dimensions, were selected. Table 28 provides a detailed account of their attributes. The RoboSimX platform's pick-and-place functionality facilitated the generation of assembly motions for each element. The assembly of the first multi-wall panel, comprising 26 elements, encountered collisions in eight instances during PTP movements. In the assembly of the second multi-wall panel, which consisted of 25 elements, collisions occurred in six instances, as shown in Figure 144.

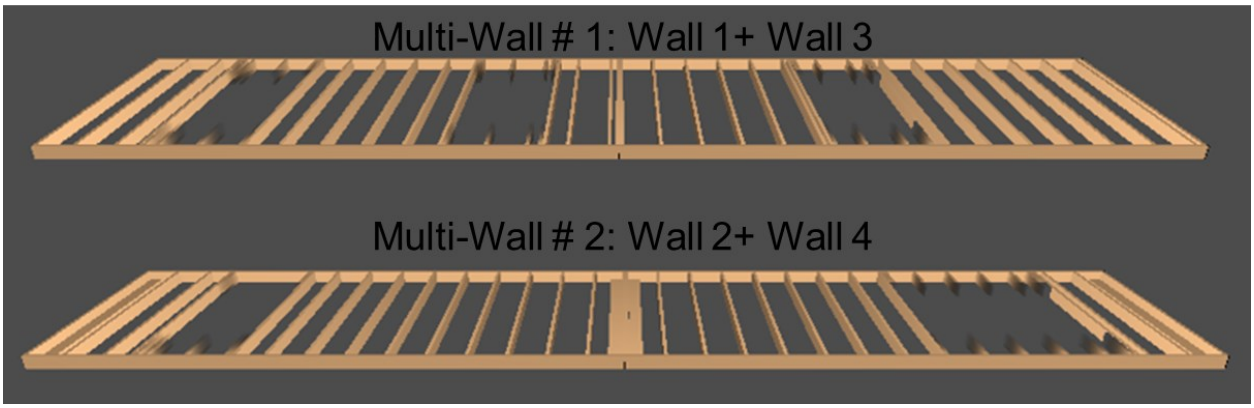


Figure 144. Simulated Multi-Wall Combo

Table 28. Specifications of the Two Types of Multi-Wall Panels

Multi-Wall ID	**Dimensions (W, H, D) ft**	**# of Elements**	**# of Assemblies (Window/Door)**	**# of Elements with Collision**
1+3	(38, 16, 0.45)	26	3	8
2+4	(40, 16, 0.45)	25	2	6

7.6.3 Simulation Results

The path-finding algorithm under investigation was deployed on 24 elements that encountered collisions during their assembly via direct PTP movements. This research executed the path-finding routine 100 iterations per component to assess effectiveness. The outcomes, including computation durations and the count of intermediary configurations for each newly established path, are listed in Table 29. The algorithm demonstrated proficiency in generating collision-free trajectories for 24 of the elements, translating to an 89% success rate. However, an element identified as Stud # 10 from multi-wall # 1 exhibited a lower success rate of 30%, and another element, Stud # 17 from the same panel, consistently failed to yield any collision-free pathways through the applied method.

The analysis captured a range of computation times across the components, with median, minimum, and maximum durations noted. Excluding the two components with low success rates, the longest computation time recorded was 52.24 seconds for Stud # 09 of panel multi-wall # 2, while the shortest was 5.23 seconds for Stud # 02 of the same panel. Remarkably, the configuration sampling phase of the computation was consistently completed within 5 microseconds, indicating that the bulk of computation time, approximately 98%, was consumed during the path searching phase. The median computation times revealed that, except for Stud # 10 and Stud # 17 of multi-wall # 1, collision-free paths were determined within a 35-sec threshold for all elements.

The low success rates observed for the two elements can be attributed to their relatively non-uniform lengths compared to other components in the assembly. This reduced scale translates into a constrained $\frac{r}{R^{(3)}}L^{(n)}$ search space within the algorithmic framework, effectively limiting the

scope of exploration for potential collision-free trajectories. Such a restricted search domain could impede the algorithm's ability to identify viable alternative pathways.

Table 29. Results of the Developed Method in Simulated Environment

Wall ID	Collided Element ID	Computational Time (s)			# of Intermediary Configurations						Success Rate (%)
		Min	Med	Max	1	2	3	4	5	6	
1	Stud 1	1.86	4.55	18.91	7	42	44	7	0	0	100
	Stud 2	1.83	2.83	16.03	16	53	27	2	0	0	98
	Stud 3	1.45	10.97	25.05	15	53	28	3	0	0	99
	Stud 6	1.80	10.65	23.86	19	43	35	3	0	0	100
	Stud 8	1.71	11.25	37.41	0	25	37	11	4	3	80
	Stud 9	1.45	11.62	23.39	0	79	11	0	0	0	90
	Stud 10	1.11	10.28	24.62	7	53	35	4	0	0	99
	Stud 12	1.75	4.24	5.8	5	50	38	7	0	0	100
	Stud 14	17.22	38.2	39.4	0	0	0	38	53	0	91
2	Stud 3	1.65	9.87	18.91	9	36	40	13	0	0	98
	Stud 4	1.56	6.25	23.27	5	33	43	16	1	0	98
	Stud 6	1.90	7.22	23.15	2	79	11	0	0	0	92
	Stud 7	1.87	8.17	24.42	10	50	32	6	1	0	99
3	Stud 1	1.18	4.47	24.62	8	47	41	3	0	0	99
	Stud 2	1.11	1.77	5.8	9	16	23	20	17	0	85
	Stud 3	1.79	14.04	26.19	5	24	51	18	0	0	98
	Stud 4	1.62	6.71	20.42	3	39	43	14	0	0	99
	Stud 8	1.45	11.65	18.43	10	52	35	3	0	0	100
4	Stud 1	1.86	6.19	25.22	18	61	19	2	0	0	100
	Stud 2	1.61	1.77	15.21	8	61	27	3	0	0	99
	Stud 4	1.45	6.71	18.49	6	43	29	9	1	0	88
	Stud 15	1.82	18.64	22.94	0	25	37	11	4	3	80
	Stud 16	1.71	10.84	19.14	0	0	0	49	38	1	88
	Stud 17	1.59	12.34	22.35	0	25	37	22	9	0	93

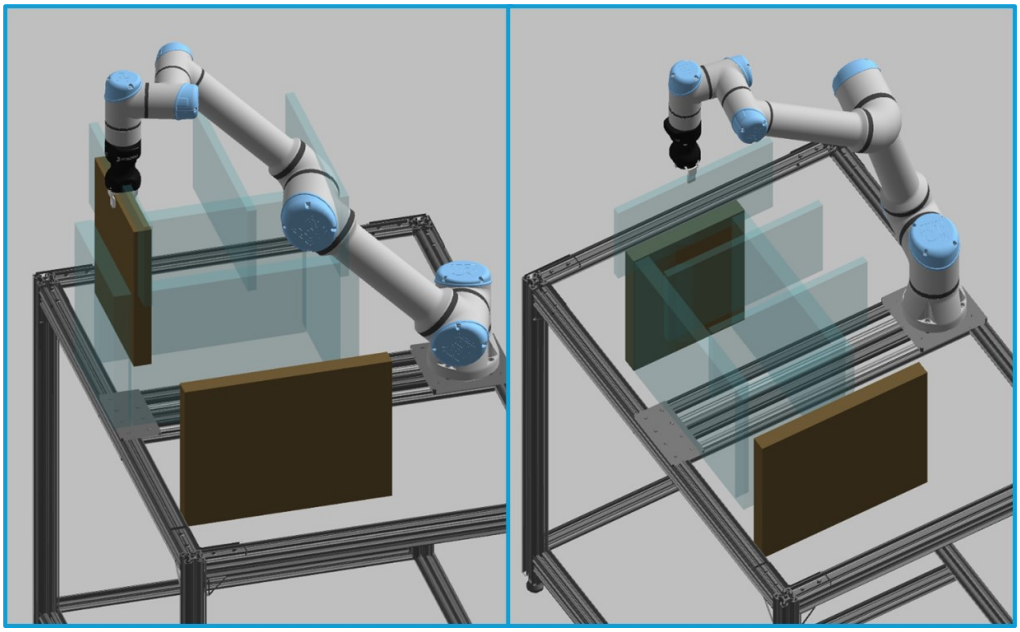


Figure 145. Sampling Results of Wall Panel #1 of the Base Model

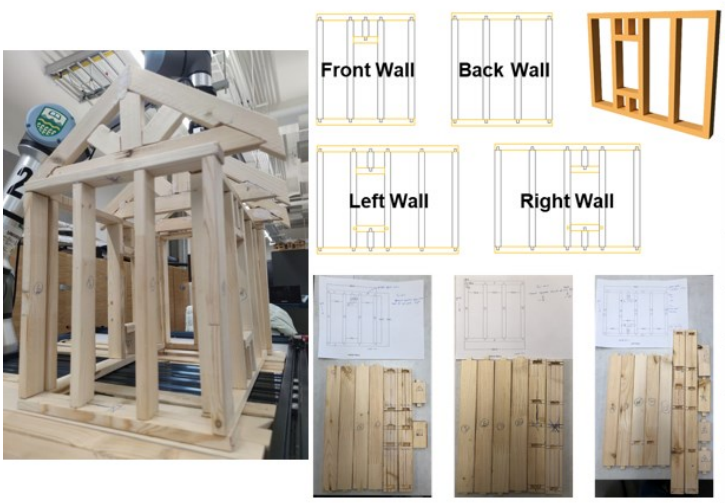
Figure 145 provides a visual representation of the configuration sampling outcomes for wall panel # 1 of the BIM Base model, revealing that the sampled configurations were in close proximity to the predefined PTP route, a consequence of the limited dimensions of these components.

7.6.4 Scaled In-lab Experiment

To test the practical applicability of the developed path-finding algorithm on an actual robotic manipulator, a laboratory-based experiment was performed utilizing a scale model of a wood-framed wall panels. The algorithmic outcomes, initially derived within the RoboSimX simulation environment, were converted into a robotic control script. This script was subsequently integrated into the operational framework of a Universal Robots (UR5e) robotic arm for execution and validation of the method in a real-world setting.

7.6.4.1 Experiment Environment Setup

The experimental model details (components, dimensions, and numbers) are shown in Figure 146 while the setup for the in-lab validation is shown in Figure 147. The experiment employed a UR5e robotic arm equipped with a two-finger gripper to serve as the manipulative device. To mimic the functionality of a material feeding mechanism, a fixed raw material picking point was set up. The designated area for frame fabrication was situated in the middle of the robotic assembly table. A digital counterpart of the experimental environment was constructed within the RoboSimX simulation platform, facilitating the derivation of pick-and-place sequences. The simulation results indicated that seven out of the nine studs (studs 1,2, 7, 8, and 9) would encounter collisions with the simulated virtual wall obstacle during the execution of the programmed tasks.



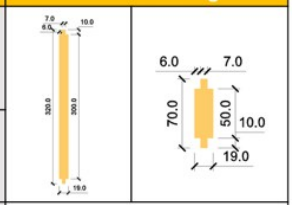
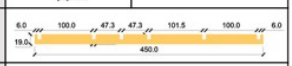
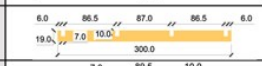
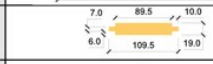
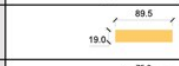
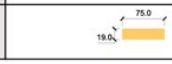
Items	Quantities	Dimensions and Legends
V-Stud-320	18	
V-Stud-70	5	
H-Stud-450	4	
H-Stud-300	4	
H-Stud-110	2	
V-Stud-90	2	
V-Stud-75	1	

Figure 146. Wall Frames Used for the Scaled Experiment

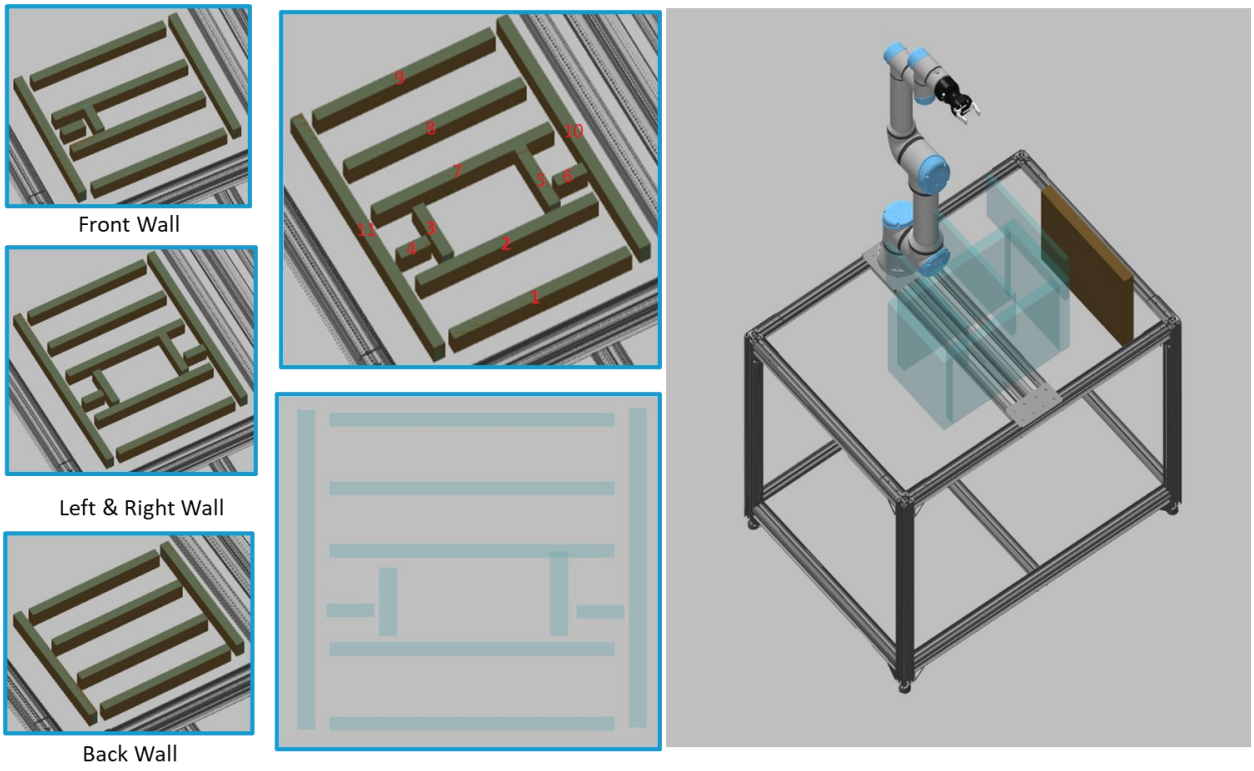


Figure 147. Environment Setup for the Scaled Experiment

Figure 148. Assembled Wall Panels

7.6.4.2 Collision Avoidance Results

The implemented algorithm was employed for five specific studs experiencing collision issues. As documented in Table 30, the computational duration and the number of transit configurations for each element (stud) are listed. This approach achieved a 99% success rate, averaging 13.56 secs to compute a collision-avoidant trajectory for each stud. The algorithm generated between one to three intermediary configurations in the revised trajectory for every element.

Subsequently, the simulation outcomes were converted into URScript, a syntax compatible with UR machines, and integrated into the UR5e for tangible robotic manipulations. Figure 149 illustrates a representative instance of the newly formulated collision-avoidant motion executed on an actual robot arm. The findings validate the capability of the developed method to address collision challenges in the OSC prefabrication workflow within a simulated framework and its applicability to an operational robotic arm. While the experiment substantiates the practical utility of the method on an actual robotic arm, the discrepancies between the simulated and real-world calibrations remain unresolved. Specifically, the inaccuracies stemming from component imperfections, mainly in the use of wood materials, are likely to impact the robotic assembly

process. Additionally, future research should focus on calibrating the virtual environment with the real-world setting to achieve acceptable precision levels.

Figure 149. In-Lab Scaled Testing Experiment of the Developed RMP for Obstacle Avoidance

Table 30. Collision-Free Path Outcome

Collided Element ID	Calculation Time (s)	# of Intermediary Configurations
Stud 1	18.24	2
Stud 2	5.68	1
Stud 7	11.81	3
Stud 8	17.23	1
Stud 9	14.84	2

The effectiveness of the proposed methodology was tested from three distinct points: 1) its effectiveness in preventing collisions, 2) the smoothness of the generated trajectories, and 3) the extent and duration of these trajectories. Detailed outcomes of these assessments are elaborated in the subsequent subsections.

7.6.4.3 Reachability for the Assembly Coordinates

In this study, the robotic manipulator utilized comprises a single prismatic joint coupled with six revolute joints. The kinematic relationship connecting the joint coordinates of the manipulator with the cartesian coordinates of the end-effector is presented by the kinematic (*K*) Equation (6). As a result, the determination of whether a specified assembly coordinate is within the kinematic reach of the robotic manipulator can be addressed by constructing the subsequent equation:

$$\begin{bmatrix} d_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \end{bmatrix} = K^{-1}\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right) \tag{6}$$

In this equation, the left side variables represent the parameters of the manipulator's joints, which are confined within specific motion parameters as explained before in Table 26. Here, K^{-1} signifies the inverse operation pertaining to the *K* while (*X, Y, Z*) corresponds to a specific assembly coordinate as shown in Table 31 with reference to for scaled in-lab experiment wall assembly (left/right) (see Figure 147). It's noteworthy that the assembly coordinates, as derived through the task planning algorithm, correspond to the geometric centroids of the prefabricated components. The cartesian coordinates of the manipulator's end-effector are determined as the point at which a component is secured by the gripper, as explained before in the RAP module.

Table 31. Manipulator Joints Parameters for Scaled In-Lab Experiment Wall Assembly

$d_1(cm)$	$\theta_2(rad)$	$\theta_3(rad)$	$\theta_4(rad)$	$\theta_5(rad)$	$\theta_6(rad)$	$\theta_7(rad)$
0.809	0.000	-1.253	0.281	-1.500	1.324	-2.600
10.850	0.000	-1.253	0.605	0.000	0.673	0.000
50.329	0.452	-0.954	0.284	-1.854	-1.006	0.675
40.138	0.021	-0.024	-0.025	0.323	-0.025	0.323
10.500	-0.162	-1.332	1.407	-1.582	1.605	-0.988
10.888	-0.821	-0.399	-0.112	0.705	-1.020	0.000
0.825	0.000	-1.253	1.144	3.764	1.182	4.493

7.6.4.4 Capability to Avoid Obstacles

Evaluation of the prototype within the simulated construction context also confirmed its proficient performance in obstacle avoidance. The motion planning algorithm, RRT*, effectively identified both the robotic manipulator and the pre-installed prefabricated components as potential obstacles. It subsequently devised a sequence of optimized robotic movements to avoid these obstacles.

7.6.4.5 Trajectory Smoothness

Utilizing the sampling-based methodology, trajectories generated may be unpredictable movements due to the sequential connection of states, potentially leading to sudden directional changes. As discussed before, the application of the Kinodynamic Relocating function plays an important role in enhancing trajectory smoothness, and this technique was incorporated in Algorithm 1. This function effectively optimized and smoothed sudden turns in the trajectory

planning within this study, leading to uniform movements of the end-effector. The wood stud assembly illustrated in Figure 150 demonstrates this concept, where both the positional and orientational trajectories exhibit smoothness, as further demonstrated in Figure 151.

For a quantitative analysis of the trajectory smoothing effectiveness of the proposed approach, the trajectory derivatives $X'(t), Y'(t), Z'(t)$ were calculated. According to Fan et al. (2020), the hallmarks of a smooth trajectory are the continuity of its derivatives and their non-zero concurrent values. The wood stud assembly trajectory shown in Figure 150 was used for this purpose. The derivatives were calculated, with the outcomes graphically represented in Figure 151. The continuous nature of the derivatives without simultaneous zero values confirm the smoothness of the trajectory.

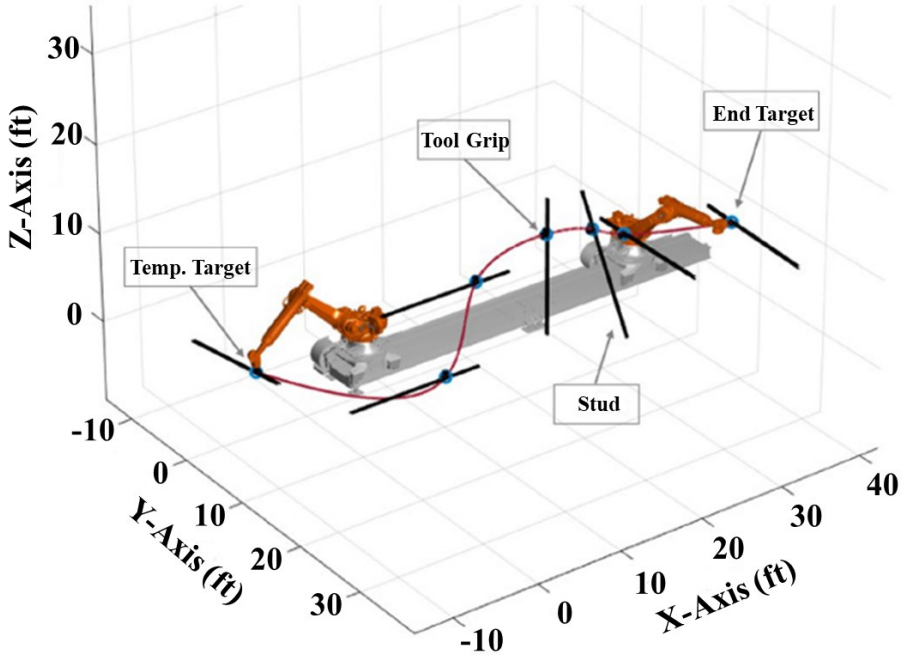


Figure 150. Wood Stud Assembly Trajectory

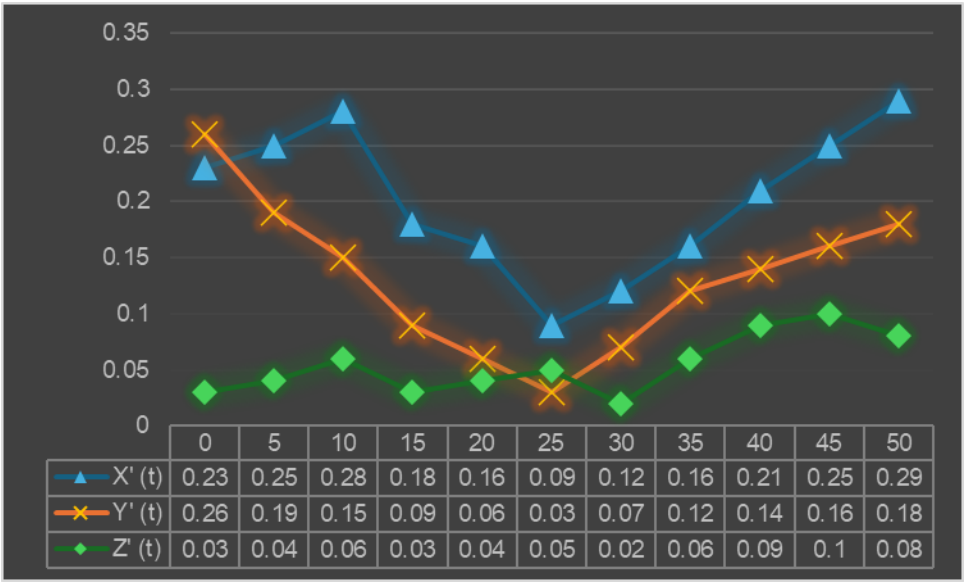


Figure 151. The Derivatives of the Planned Trajectory for Wood Studs Assembly

7.6.4.6 Trajectory Length and Execution Time

Table 32 presents the data regarding the average trajectory length and the execution time for each category of the wall panel, along with the cumulative execution time. The data indicates that the total execution time approximates 10 minutes, considering the simultaneous operation of four robotic manipulators. Consultations with industry experts from contracting firms revealed that the anticipated manual assembly duration for two individuals working concurrently is around 55 minutes. This comparison suggests a reduction in construction time when employing robotic manipulators.

It is important to note that the primary objective of this study was to design and develop a collision-free trajectory method that integrate with robotic manipulators to accurately position construction components at specified locations. An additional step in this process involves the nailing of these components. Therefore, the reported total execution time might be subject to an upward adjustment to incorporate the time necessary for this nailing procedure.

Table 32. Trajectory Length and Execution Time Results

Element Type	Number of Elements	Average Trajectory Length (m)	Average Execution Time (s)
Vertical Studs	52	3.870	299
Horizontal Tracks	8	7.871	116
OSB Sheathing	31	7.045	85
Window Component	6	3.408	49
Door Component	1	2.435	30

7.7 Discussion

This study introduces an innovative approach for planning collision-free trajectories to facilitate the robotic assembly of prefabricated elements in OSC settings. A review of existing literature in construction robotics revealed that while robot end-effectors and manipulated building components are often modelled as particles in trajectory planning, the substantial volume of prefabricated building components suggests they should more accurately be represented as rigid bodies. Treating these elements as mere particles in trajectory planning could lead to potential collisions with surrounding structures. The research addresses this gap by enhancing the RRT* algorithm with dimensional-centric collision detection, allowing for trajectory adjustments based on the actual dimensions of prefabricated building components in OSC.

The effectiveness of this method was assessed based on three criteria: 1) collision avoidance, 2) trajectory smoothness, and 3) trajectory length and execution time. To evaluate obstacle avoidance, Minkowski distances between the obstacle space and the manipulated objects at different states

along the planned trajectories were calculated. The results indicated that the building elements were consistently kept at a safe distance from obstacles throughout the assembly process, successfully preventing collisions. For assessing trajectory smoothness, the trajectory derivatives were analyzed, showing continuous values without simultaneous zeroes, indicative of smooth trajectories. Additionally, the average trajectory lengths and execution times for various elements categories were recorded, yielding high efficiencies in both labor and time compared to manual construction methods.

This approach proved to be more efficient in terms of labor and time. The total assembly time using this method was approximately 10 minutes compared to the 55 minutes estimated for manual assembly by two workers, as per industry professionals. However, it's important to note that the total time for robotic assembly might increase once additional processes, such as nailing the elements, are considered.

This research has developed a sophisticated approach for collision-free trajectory planning in the robotic assembly of prefabricated components specifically tailored for OSC. This approach encompasses an RAP algorithm (discussed in the previous chapter) and an RMP planning algorithm that effectively: 1) derive vectors to mathematically correlate the coordinates of prefabricated components with their assembly sequence, incorporating geometry and centroid considerations for robotic construction and 2) analyze the RAP output to autonomously generate kinematic parameters for the robotic assembly, eliminating the need for human intervention.

While the prototype was evaluated within a ROS environment rather than in actual factory conditions, the simulation environment was crafted to mirror real-world conditions. Firstly, the environmental physical attributes, including density, gravity, damping, dimensions, and material

properties like color and texture, were closely aligned with reality. Secondly, for real-world applications, OSC resources can be precisely mirrored in the ROS environment. This includes accurate representations of the quantity and locations of prefabricated building components, ensuring a seamless transfer of digital object properties to their physical counterparts, thereby enhancing spatial reasoning and motion planning efficiency. Thirdly, the environment encompasses an exact kinematic model of the robot, ensuring that the motions planned and validated in the simulation can be accurately replicated in real-world assembly tasks. A laboratory-based pilot trial was conducted to confirm the feasibility of these robot motions in a real setting. The results affirmed that the joint movements and end-effector reaches, as planned and verified in ROS, were achievable in real-world condition.

Chapter 8 : Conclusion, Contributions, Limitation, and Future Roadmap

8.1 General Conclusion

This research introduces a comprehensive method that significantly advances the field of BIM and robotic construction, especially in wood/LGS construction, across four main areas:

(1) Innovative Technique for BIM Analysis: The method utilizes rule-based representation and rationalization through FOL and SOL to extract construction-level information. This data can seamlessly integrate into downstream processes like CNC machining or robotic operations, thereby promoting a fully digital workflow. The study showcases the effectiveness of this approach in handling IFC schema data within the AEC industry, enhancing information extraction from IFC-based BIM models and moving beyond the binary constraints of FOL to include set quantifications for richer querying capabilities.

(2) Standardization and Interoperability: By employing IFC for BIM data representation, the method diverges from conventional BIM APIs embedded in proprietary software, enhancing standardization and fostering interoperability across the AEC sector.

(3) Practical Implementation and Validation: The method was implemented using B-Prolog and validated across six BIM instance models, demonstrating high precision, accuracy, and F1 measure. By continuously expanding the rule set with additional assembly rules, the method's analytical capabilities are progressively enhanced.

(4) Addressing Limitations and Future Directions: Despite its advancements, the study recognizes limitations such as the reliance on bounding boxes for dimensioning irregularly shaped BIM objects, which can introduce inaccuracies. The effectiveness of the method is also dependent on the level of development or detail in the BIM instance model. Future work includes developing a knowledge model to better analyze wood/LGS construction

and extracting more detailed content from LOD 400 BIM instance models or incorporating additional data from shop drawings or specifications.

Additionally, this study makes a novel contribution by developing the ASP algorithm. This algorithm facilitates the determination of both assembly coordinates and the sequential order of assembly for components within a BIM model. The research also proposes a methodology for automatically generating simulations of robotic automation in wood/LGS framing and sheathing construction, including nailing operations. This approach integrates BIM with robotics and was validated through simulations involving two types of industrial robots, considering their specific characteristics.

This research also introduces a comprehensive trajectory planning methodology for the robotic assembly of lightweight structures, specifically designed for OSC prefabricated elements. The developed method incorporated into three planning levels: assembly, planning, and joint. These planner levels sequentially collaborate to formulate plans enabling robotic manipulators to execute assembly tasks while avoiding potential collisions. The ASP module generates a sequence of assembly tasks, detailing both the assembly coordinates and the order of prefabricated components for the OSC prefabricated components. These coordinates determine the positions of the robotic TCP during trajectory planning. At the manipulator or TCP planning level, the configuration of the robotic TCP for each task is determined. The joint planning level then synthesizes information from the preceding levels, computing joint movements that adhere to kinematic constraints, thereby guiding the manipulators to the desired end-effector positions in a time-sequenced manner for component assembly.

Acknowledging the substantial volume of prefabricated components in OSC construction, the approach integrates dimensional-based collision checks using Minkowski Difference computations. This enhancement allows for trajectory adjustments in response to the physical dimensions of the components, thereby averting potential collisions. The RRT* motion planning algorithm considers the manipulator base's movement in kinematic analysis and treats the manipulator and installed components as obstacles, optimizing robotic motions accordingly. This innovation addresses a significant gap in construction literature, providing a method for trajectory planning of voluminous prefabricated components. The effectiveness of this approach was evaluated from three perspectives: collision avoidance, trajectory smoothness, and trajectory length and execution time. The results confirmed its satisfactory performance across these metrics.

In conclusion, this research proposes a visible, game engine-based collision avoidance method for robotic OSC prefabrication. Despite its effectiveness, as evidenced by in-lab experiments and simulated applications, the method faces limitations, particularly in handling small-sized components, being confined to revolute joint robotic arms, and requiring user-provided 3D models and FK/IK solvers. Future research will aim to refine the method, expanding its applicability to include robotic arms with multi-prismatic joints and addressing the challenges of smaller components.

8.2 Contributions to the body of knowledge

The proposed RAP methodology enriches the existing body of knowledge in the AEC domain in four significant ways:

(1) Integration Framework for Design and Construction Phases: The methodology offers an integration framework that bridges the gap between building design and construction

phases. This is achieved through a BIM-robotics workflow, fostering enhanced collaboration and coordination. This approach contrasts with the current practice in offsite construction, where operations are often conducted in siloes.

(2) Incorporation of Practical Knowledge from Robotics and Construction: It encompasses practical insights from both robotics and construction fields. The methodology details the implementation of robotic systems, promoting broader adoption of construction robotics within the AEC sector.

(3) Automated BIM Data Utilization for Robotic System Analysis: The methodology provides a tool for automatically generating necessary information from BIM. This serves as input for operational analysis of robotic systems, particularly supporting automation in wood frame assembly for offsite construction. The simulation-based approach, utilizing robotic simulation, enables preconstruction analysis of industrial arms for framing and fastening operations, aiding in the planning and strategizing of automation solutions.

(4) Promoting Interoperability with IFC-Based BIM and Open-Source Robotics Simulation: By integrating IFC-based BIM with an open-source robotic simulator like ROS, the methodology addresses and contributes to resolving interoperability issues in information exchange within the AEC industry. This not only aids in the current project but also sets a precedent and provides a reusable blueprint for future research and practical applications in analyzing various robotic systems and construction workflows.

Overall, this methodology presents a comprehensive and innovative approach, combining theoretical insights with practical applications, to advance the integration of robotic systems in construction, thereby contributing to the evolution and efficiency of the AEC industry.

The RMP research introduces an innovative path-finding approach for employing an industrial robot arm in the prefabrication of OSC, offering significant contributions to the existing knowledge in this domain:

- Collision-Free Path Generation with Component Size Consideration: The path-finding method developed in this study addresses collision issues by sampling configurations of the robot arm, while incorporating the dimensions of the building component. Neglecting the size of the target object may lead to unnecessarily extended computation times and unpredictable robot arm movements, potentially resulting in impractical outcomes. By restricting the search area in line with the component's size, the proposed method efficiently identifies new paths within a confined search zone.
- Visualization of Path-Finding Process: Unlike traditional methods that operate in configuration space (C-space), this method employs a virtual robot arm navigating within a simulated environment, thereby avoiding the computational demands of constructing C-space. This approach is particularly advantageous given the dynamic nature of the environment and the object being manipulated during assembly. High-dimensional path-finding in C-space can be challenging to conceptualize or visualize for engineers. The new collision-free path and its searching process are graphically displayed in the virtual environment, utilizing a game engine-based platform. This simulation allows engineers to review outcomes and plan human-robot interactions prior to transitioning to actual robotic production lines.
- Universal Pathfinder for OSC Prefabrication: This research provides a versatile pathfinder applicable to modular home prefabrication, adaptable to various commercial robot arms.

By supplying the 3D model and the FK/IK solver of the robot arm, the path generator can be universally implemented across different robotic systems.

8.3 Research Limitations and Future Research Roadmap

While the path-finding method developed in this study facilitates the identification of collision-free trajectories for OSC prefabrication, several challenges remain to be addressed:

- Limitation with Small-Sized Components: The current method leverages the size of building components to guide the robot arm in performing a sub-global search, avoiding unpredictable transitional locations. However, this approach struggles to find collision-free paths for smaller components due to the restricted search area. In instances of failure, manual intervention might be required to insert new transit points between the start and goal positions, enabling the robot arm to avoid obstacles. Future research should concentrate on establishing a minimal search boundary $D_{min_{(n)}}$ to avoid these failures when dealing with small-sized components.

- Restriction to Revolute Joints: Presently, the method is applicable exclusively to industrial robot arms equipped with revolute joints and one prismatic joint. However, multi-prismatic joints, such as inter-connected linear tracks, are often essential for handling larger panels frequently encountered in OSC prefabrication and larger accessible means. Future studies should expand the scope to include multi-prismatic joints, thereby enhancing the versatility of the robotic arm.

- Necessity for FK/IK Solver: The path-finding method necessitates the pre-preparation of both 3D models and FK/IK solvers for the robot arm. This prerequisite may pose a

challenge for users who must ensure these components are ready before implementing the proposed method.

- Sim-to-Real Calibrations: Although the method permits preliminary planning of robotic motions in a virtual environment, the calibration between this virtual setting and the actual real-world environment has not been fully addressed. Upcoming research will feature sim-to-real calibrations, aiming to dynamically adapt robot motions in response to imperfections in building components. This adaptation could involve the integration of external sensors such as vision sensors, LiDAR, or theodolites.

Future directions involve expanding the methodology to accommodate various shapes and configurations of wood/LGS frames, testing different building components and robotic systems, and conducting a comparative analysis with other robotic simulators. There is also an initiative to develop a unified system that integrates the simulation approach with a rule-based system for a more efficient and comprehensive analysis of BIM and robotic systems in construction.

References

Ab Rashid, M. F. F. (2017). A hybrid Ant-Wolf algorithm to optimize assembly sequence planning problem. Assembly Autom, 37(2), 238–248.

Abdullah, M. A., Ab Rashid, M. F. F., & Ghazalli, Z. (2019). Optimization of assembly sequence planning using soft computing approaches: a review. Archives of Computational Methods in Engineering, 26, 461-474.

Aghababa, M. P. (2012). 3D path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles. Applied Ocean Research, 38, 48–62.

Aguiar, A. J. C., Silva, A. S. A., & Villani, E. (2008). Graphic Robot Simulation for the Design of Work Cells in the Aeronautic Industry. BCM Symposium Series in Mechatronics, 3, 346-354.

Aibinu AM, Bello Salau H, Rahman NA, Nwohu MN, Akachukwu CM (2016) A novel clustering based genetic algorithm for route optimization. Eng Sci Technol Int J 19(4):2022–2034

Alfadhlani, S., Samadhi, T. A., Ma'ruf, A. N., et al. (2011). Automatic collision detection for assembly sequence planning using a three-dimensional solid model. J Adv Manuf Syst, 10, 277–291.

Almasri, E., & Uyguroğlu, M. K. (2021). Trajectory optimization in robotic applications: survey of recent developments. Mechatronics, 39, 174-184.

Alwisy, A., Bu Hamdan, S., Barkokebas, B., Bouferguene, A., & Al-Hussein, M. (2019). A BIM-based automation of design and drafting for manufacturing of wood panels for modular residential buildings. International Journal of Construction Management, 19(3), 187-205.

Arunkumar, G., Gnanambal, I., Naresh, S., Karthik, P. C., & Patra, J. K. (2016). Parameter optimization of three-phase boost inverter using genetic algorithm for linear loads. Energy Procedia, 90, 559–565.

Auinger, F., Vorderwinkler, M., & Buchtela, G. (1999, December). Interface driven domain-independent modeling architecture for "soft-commissioning" and "reality in the loop". In Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1 (pp. 798-805).

Bahubalendruni, M. R., & Biswal, B. B. (2016). A note on mechanical feasibility predicate for robotic assembly sequence generation. In CAD/CAM robotics and factories of the future (pp. 397–404). New Delhi: Springer.

Bahubalendruni, M. R., & Biswal, B. B. (2014). Computer aid for automatic liaisons extraction from CAD-based robotic assembly. In 8th IEEE international conference on intelligent systems and control (ISCO), Karpagam College of Engineering, Coimbatore, India (pp. 42–45). IEEE.

Bahubalendruni, M. R., & Biswal, B. B. (2014). An algorithm to test feasibility predicate for robotic assemblies. Trends Mech Eng Technol, 4, 11–16.

Bahubalendruni, M. R., & Biswal, B. B. (2015). Influence of assembly predicate consideration on optimal assembly sequence generation. Assembly Autom, 35, 309–316.

Bai, Y. W., Chen, Z. N., Bin, H. Z., et al. (2005). An effective integration approach toward assembly sequence planning and evaluation. Int J Adv Manuf Technol, 27, 96–105.

Baldwin, D. F., Abell, T. E., Lui, M. C., et al. (1991). An integrated computer aid for generating and evaluating assembly sequences for mechanical products. IEEE Trans Robot Autom, 7, 78–94.

Bahubalendruni, M. R., & Biswal, B. B. (2014). Computer aid for automatic liaisons extraction from CAD-based robotic assembly. In 8th IEEE international conference on intelligent systems and control (ISCO), Karpagam College of Engineering, Coimbatore, India (pp. 42–45). IEEE.

Bai, Y. W., Chen, Z. N., Bin, H. Z., et al. (2005). An effective integration approach toward assembly sequence planning and evaluation. Int J Adv Manuf Technol, 27, 96–105.

Baldwin, D. F., Abell, T. E., Lui, M. C., et al. (1991). An integrated computer aid for generating and evaluating assembly sequences for mechanical products. IEEE Trans Robot Autom, 7, 78–94.

Baskerville, R. L., Kaul, M., & Storey, V. C. (2015). Genres of inquiry in design-science research. MIS Quarterly, 39(3), 541-564.

Biswal, B. B., Deepak, B. B., & Rao, Y. (2013). Optimization of robotic assembly sequences using immune-based technique. J Manuf Technol Manage, 24, 384–396.

Bock, T., & Linner, T. (2015). Robotic industrialization. Cambridge University Press.

Bock, T., & Linner, T. (2016). Site automation. Cambridge University Press.

Bokor, O., Florez, L., Osborne, A., & Gledson, B. J. (2019). Overview of construction simulation approaches to model construction processes. Organization Technology & Management in Construction: An International Journal, 11(1), 1853-1861.

Bonneville, F., Perrard, C., & Henrioud, J. M. (1995). A genetic algorithm to generate and evaluate assembly plans. In INRIA/IEEE symposium on emerging technologies and factory automation (ETFA'95), Vellore Institute of technology, Vellore, Chennai, India (Vol. 2, pp. 231–239). IEEE.

Boothroyd, G., Dewhurst, P., & Knight, W. A. (2002). Product design for manufacture and assembly. Revised and expanded. CRC press.

Bourjault, A. (1984). Contribution a une approche méthodologique de l'assemblage automatisé: elaboration automatiquede séquences opératoires. Thése d'état. Besançon, France: Université de Franche-Comté.

Brell-Cokcan, S., & Braumann, J. (Eds.). (2013). Rob|Arch 2012: Robotic fabrication in architecture, art and design. Springer Science & Business Media.

Bullinger, H. J., & Ammer, E. D. (1984). Computer-aided depicting of precedence diagrams – a step towards efficient planning in assembly. Comput Ind Eng, 8, 165–169.

Cao, P. B., & Xiao, R. B. (2007). Assembly planning using a novel immune approach. Int J Adv Manuf Technol, 31, 770–782.

Carpo, M. (2017). The second digital turn: Design beyond intelligence. MIT Press.

Chen, R. S., Lu, K. Y., & Yu, S. C. (2002). A hybrid genetic algorithm approach on multi-objective of assembly planning problem. Eng Appl Artif Intell, 15, 447–457.

Chen, S. F., & Liu, Y. J. (2001). An adaptive genetic assembly-sequence planner. Int J Comput Integr Manuf, 14(5), 489–500.

Cheng, D., & Ma, J. (2020, July). Research on the restrictive factors of the development of Chinese prefabricated buildings. In IOP Conference Series: Earth and Environmental Science (Vol. 531, No. 1, p. 012044). IOP Publishing.

Choi, Y. K., Lee, D. M., & Cho, Y. B. (2008). An approach to multicriteria assembly sequence planning using genetic algorithms. Int J Adv Manuf Technol, 42(1–2), 180–188.

Chryssolouris, G. (2006). Manufacturing Systems: Theory and Practice. 2nd ed. New York: Springer-Verlag.

Chryssolouris, G., Mavrikios, D., Rentzos, L., & Mourtzis, D. (2009). Manufacturing Systems: Skills and competences to realise the vision of Industry 4.0. Procedia CIRP, 7, 17–24.

Chryssolouris, G., Rentzos, L., & Mavrikios, D. (2012). The teaching factory: A manufacturing paradigm. Procedia CIRP, 3, 338–343.

Dachs, B., Fu, X., & Jäger, A. (2022). The diffusion of industrial robots. In The Routledge handbook of smart technologies (pp. 290-311). Routledge.

Dachs, B., Kinkel, S., Jäger, A., & Palčič, I. (2019). Backshoring of production activities in European manufacturing. *Journal of Purchasing and Supply Management, 25*(3), 100531.

Dantam, N. T. (2020). Task and motion planning. In Springer encyclopedia of robotics. Springer.

Davtalab, O., Kazemian, A., & Khoshnevis, B. (2018). Perspectives on a BIM-integrated software platform for robotic construction through contour crafting. *Automation in Construction, 89*, 13-23.

de Giorgio, A., Maffei, A., Onori, M., & Wang, L. (2021). Towards online reinforced learning of assembly sequence planning with interactive guidance systems for industry 4.0 adaptive manufacturing. *Journal of Manufacturing Systems, 60*, 22-34.

de Soto, B. G., Agustí-Juan, I., Hunhevicz, J., Joss, S., Graser, K., Habert, G., & Adey, B. T. (2018). Productivity of digital fabrication in construction: Cost and time analysis of a robotically built wall. *Automation in Construction, 92*, 297-311.

Delgado, J. M. D., Oyedele, L., Ajayi, A., Akanbi, L., Akinade, O., Bilal, M., & Owolabi, H. (2019). Robotics and automated systems in construction: Understanding industry-specific challenges for adoption. *Journal of Building Engineering, 26*, 100868.

Devadass, P., Stumm, S., & Brell-Cokcan, S. (2019, May). Adaptive haptically informed assembly with mobile robots in unstructured environments. In *Proceedings of the 36th International Symposium on Automation and Robotics in Construction (ISARC)* (Vol. 36, pp. 469-476). International Association for Automation and Robotics in Construction (IAARC).

DiLuoffo, V., Michalson, W. R., & Sunar, B. (2018). Robot Operating System 2: The need for a holistic security approach to robotic architectures. *International Journal of Advanced Robotic Systems, 15*(3), 1729881418770011.

Erdem, E., Patoglu, V., & Schüller, P. (2016). A systematic analysis of levels of integration between high-level task planning and low-level feasibility checks. *AI Communications, 29*(2), 319-349.

Eversmann, P., Gramazio, F., & Kohler, M. (2017). Robotic prefabrication of timber structures: Towards automated large-scale spatial assembly. *Construction Robotics, 1*(1-4), 49-60.

Garcia del Castillo Lopez, J. (2019). Machina.NET: A library for programming and real-time control of industrial robots. *Journal of Open Research Software*.

Giftthaler, M., Sandy, T., Dörfler, K., Brooks, I., Buckingham, M., Rey, G., ... & Buchli, J. (2017). Mobile robotic fabrication at 1:1 scale: The in situ fabricator: System, experiences and current developments. *Construction Robotics, 1*, 3-14.

Gurgul, M. (2018). *Industrial robots and cobots: Everything you need to know about your future co-worker*. Michał Gurgul.

Hevner, A., & Chatterjee, S. (2010). *Design research in information systems: Theory and practice*. Springer.

Hevner, A., Chatterjee, S., Hevner, A., & Chatterjee, S. (2010). Design science research in information systems: Theory and practice. In *Design research in information systems: Theory and practice* (pp. 9-22). Springer.

Hoover, S., Trombitas, P., & Cowles, E. (2017). Prefabrication: The changing face of engineering and construction. *Fails Management Institute-FMI/BIM Forum Prefabrication Survey*: Denver, CO, USA.

International Organization for Standardization. (2022). ISO/IEC 2382 (en), robotics—vocabulary. Retrieved from https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-3:v1:en

Iturralde, K., & Bock, T. (2018). Integrated, automated and robotic process for building upgrading with prefabricated modules. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction* (Vol. 35, pp. 1-8). IAARC Publications.

Janssen, P. A. T. R. I. C. K. (2015). Parametric BIM workflows. In *CAADRIA 2015-20th International Conference on Computer-Aided Architectural Design Research in Asia: Emerging Experiences in the Past, Present and Future of Digital Architecture* (pp. 437-446).

Kiefer, J. (2007). *Mechatronikorientierte Planung automatisierter Fertigungszellen im Bereich Karosserierohbau*.

Kim, S., Peavy, M., Huang, P. C., & Kim, K. (2021). Development of BIM-integrated construction robot task planning and simulation system. *Automation in Construction, 127*, 103720.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele, UK: Keele University.

Lee, C. G., & Park, S. C. (2014). Survey on the virtual commissioning of manufacturing systems. *Journal of Computational Design and Engineering, 1*(3), 213-222.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics, 7*(66), eabm6074.

Mah, D., & Al-Hussein, M. (2008). An integrated evaluation framework for sustainable residential construction. *The International Journal of Interdisciplinary Social Sciences, 3*(6), 129-136.

MBI. (2018). *Permanent Modular Construction Report*. Modular Building Institute: Charlottesville, VA, USA.

Mongeon, P., & Paul-Hus, A. (2016). Liputan jurnal Web of Science dan Scopus: Analisis komparatif. *Scientometrics, 106*(1), 213-228.

Pérez, L., Rodríguez-Jiménez, S., Rodríguez, N., Usamentiaga, R., & García, D. F. (2020). Digital twin and virtual reality based methodology for multi-robot manufacturing cell commissioning. *Applied Sciences, 10*(10), 3633.

García de Soto, B., Agustí-Juan, I., Joss, S., & Hunhevicz, J. (2022). Implications of Construction 4.0 to the workforce and organizational structures. International journal of construction management, 22(2), 205-217.

Poirier, E. A.,Forgues, D., & Staub-French, S. (2014). Dimensions of interoperability in the AEC industry. In *Construction Research Congress 2014: Construction in a Global Network* (pp. 1987-1996).

Poirier, E. A., Frénette, S., Carignan, V., Paris, H., & Forgues, D. (2018). Increasing the performance of the Quebec construction sector through the digital shift: Study on the deployment of building data modeling tools and practices in Quebec.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: An open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

Race, S. (2019). *BIM demystified*. Routledge.

Renaud, M. (2000). A simplified inverse kinematic model calculation method for all 6R type manipulators. In *Current Advances in Mechanical Design and Production VII* (pp. 15-25). Pergamon.

Shepherd, D. (2019). BIM adoption and maturity levels. In *BIM management handbook* (pp. 50-63). RIBA Publishing.

Siciliano, B., Khatib, O., & Kröger, T. (Eds.). (2008). *Springer handbook of robotics* (Vol. 200). Springer.

Singh, V. K., Singh, P., Karmakar, M., Leta, J., & Mayr, P. (2021). The journal coverage of Web of Science, Scopus and Dimensions: A comparative analysis. *Scientometrics, 126*, 5113-5142.

Søndergaard, A., Amir, O., Eversmann, P., Piskorec, L., Stan, F., Gramazio, F., & Kohler, M. (2016). Topology optimization and robotic fabrication of advanced timber space-frame structures. In *Robotic Fabrication in Architecture, Art and Design 2016* (pp. 190-203).

Statista - The Statistics Portal. (2017a). Worldwide sales of industrial robots from 2004 to 2017 (in 1,000 U.S. dollars). Retrieved from https://www.statista.com/statistics/264084/worldwide-sales-of-industrial-robots/

Statista - The Statistics Portal. (2017b). Degree of robotic process automation/digital labor implementation in U.S. engineering and construction companies as of 2017. Retrieved from https://www.statista.com/statistics/805207/degree-of-technological-adoption-in-us-engineering-and-construction-companies-2017/

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: A framework for evaluation in design science research. *European Journal of Information Systems, 25*, 77-89.

Wan, A., Xu, J., Chen, H., Zhang, S., & Chen, K. (2017). Optimal path planning and control of assembly robots for hard-measuring easy-deformation assemblies. *IEEE/ASME Transactions on Mechatronics, 22*(4), 1600-1609.

Waltman, L., Van Eck, N. J., & Noyons, E. C. (2010). A unified approach to mapping and clustering of bibliometric networks. *Journal of Informetrics, 4*(4), 629-635.

Yin, X., Liu, H., Chen, Y., & Al-Hussein, M. (2019). Building information modelling for off-site construction: Review and future directions. *Automation in Construction, 101*, 72-91.

Zhang, J., Keramat, F., Yu, X., Hern, D. M., Queralta, J. P., & Westerlund, T. (2022). Distributed robotic systems in the edge-cloud continuum with ROS 2: A review on novel architectures and technology readiness. *arXiv preprint arXiv:2211.00985*.

Zhang, Y., Wu, H., Zhang, Z., Xiao, L., & Guo, D. (2013). Acceleration-level repetitive motion planning of redundant planar robots solved by a simplified LVI-based primal-dual neural network. *Robotics and Computer-Integrated Manufacturing, 29*(2), 328-343.

Kalpakjian S and Schmid S. Manufacturing, engineering and technology. SI 6th ed. The University of Notre Dame, Paris: Digital Designs, 2006.

Bahubalendruni MR and Biswal BB. A review on assembly sequence generation and its automation. Proc IMechE, Part C: J Mechanical Engineering Science 2016; 230: 824–838.

Wilson RH and Latombe JC. Geometric reasoning about mechanical assembly. Artif Intell 1994; 71: 371–396.

Mok SM, Ong K and Wu CH. Automatic generation of assembly instructions using STEP. In: IEEE international conference on robotics and automation. Proceedings 2001 ICRA. Vol. 1, pp. 313–318. COEX Seoul, South Korea: IEEE.

Pan C, Smith SS and Smith GC. Automatic assembly sequence planning from STEP CAD files. Int J Comput Integr Manuf 2006; 19: 775–783.

Li YL, Gao JM, Shi L, et al. Assembly modeling for 3d component based on polychromatic sets. In: Prof. Graeme E. Murch and Prof. Iulian Antoniac (eds) Advanced materials research. vol. 411, Switzerland: Trans Tech Publications, 2012, pp.388–392.

Giri R and Kanthababu M. Generating complete disassembly sequences by utilising two-dimensional views. Int J Prod Res 2015; 53: 5118–5138.

Yu J and Wang C. Method for discriminating geometric feasibility in assembly planning based on extended and turning interference matrix. Int J Adv Manuf Technol 2013; 67: 1–6.

Yu J, Da Xu L, Bi Z, et al. Extended interference matrices for exploded view of assembly planning. IEEE Trans Autom Sci Eng 2014; 11: 279–286.

Bahubalendruni MR, Biswal BB, Kumar M, et al. Influence of assembly predicate consideration on optimal assembly sequence generation. Assembly Autom 2015; 35: 309–316.

Ghandi S and Masehian E. A breakout local search (BLS) method for solving the assembly sequence planning problem. Eng Appl Artif Intell 2015 Mar 31; 39:245–266.

Guo J, Tang H, Sun Z, et al. An improved shuffled frog leaping algorithm for assembly sequence planning of remote handling maintenance in radioactive environment. Sci Technol Nucl Install 2015; 1: 1–14.

Ibrahim I, Ibrahim Z, Ahmad H, et al. An assembly sequence planning approach with a rule-based multistate gravitational search algorithm. Int J Adv Manuf Technol 2015; 79: 1363–1376.

Marian RM. Optimisation of assembly sequences using genetic algorithms. University of South Australia, Adelaide, SA, Australia, 2003.

Ghandi S and Masehian E. Review and taxonomies of assembly and disassembly path planning problems and approaches. Comput Aid Des 2015; 67: 58–86.

Youhui L, Xinhua L, Qi L (2012) Assembly sequence planning based on ant colony algorithm. Future Commun Comput Control Manag 141(51105069):397–404.

Hsu HY, Lin GCI (2002) Quantitative measurement of component accessibility and product assemblability for design for assembly application. Robot Comput Integr Manuf 18(1):13–27.

Li M, Zhang Y, Zeng B, Zhou H, Liu J (2015) The modified firefly algorithm considering fireflies' visual range and its application in assembly sequences planning. Int J Adv Manuf Technol 82(5–8):1381–1403.

Lu C, Yang Z (2016) Integrated assembly sequence planning and assembly line balancing with ant colony optimization approach. Int J Adv Manuf Technol 83(1):243–256.

Wang Y, Liu JH (2010) Chaotic particle swarm optimization for assembly sequence planning. Robot Comput Integr Manuf 26(2):212–222.

Kumar ER, Annamalai K (2015) Recent nontraditional optimization techniques for Assembly sequence planning. Int J Pure Appl Math 101(5):707–715.

Zhang H, Liu H, Li L (2013) Research on a kind of assembly sequence planning based on immune algorithm and particle swarm optimization algorithm. Int J Adv Manuf Technol 71(5–8):795–808.

Rashid MFF, Hutabarat W, Tiwari A (2011) A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. Int J Adv Manuf Technol 59(1–4):335–349.

Marian RM, Luong LHS, Abhary K (2006) A genetic algorithm for the optimisation of assembly sequences. Comput Ind Eng 50(4):503–527.

Ghandi S, Masehian E (2015) Assembly sequence planning of rigid and flexible parts. J Manuf Syst 36:128–146.

Choi Y-K, Lee DM, Bin Cho Y (2008) An approach to multicriteria assembly sequence planning using genetic algorithms''. Int J Adv Manuf Technol 42(1–2):180–188.

Li M, Wu B, Hu Y, Jin C, Shi T (2013) A hybrid assembly sequence planning approach based on discrete particle swarm optimization and evolutionary direction operation. Int J Adv Manuf Technol 68(1–4):617–630.

Wang H, Rong Y, Xiang D (2014) Mechanical assembly planning using ant colony optimization. Comput Des 47:59–71.

Guo J, Sun Z, Tang H, Yin L, Zhang Z (2015) Improved cat swarm optimization algorithm for assembly sequence planning. Open Autom Control Syst J 7(1):792–799.

Lv H and Lu C. An assembly sequence planning approach with a discrete particle swarm optimization algorithm. Int J Adv Manuf Technol 2010; 50: 761–770.

19. De Fazio T and Whitney D. Simplified generation of all mechanical assembly sequences. IEEE J Robot Autom 1987; 3: 640–658.

De Mello LH and Sanderson AC. AND/OR graph representation of assembly plans. IEEE Trans Robot Autom 1990; 6: 188–199.

Sanderson AC, de Mello LS and Zhang H. Assembly sequence planning. AI Mag 1990; 11: 62.

De Mello LH and Sanderson AC. A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE Trans Robot Autom 1991; 7: 228–240.

Baldwin DF, Abell TE, Lui MC, et al. An integrated computer aid for generating and evaluating assembly sequences for mechanical products. IEEE Trans Robot Autom 1991; 7: 78–94.

Chryssolouris, G. 2006. Manufacturing Systems: Theory and Practice. 2nd ed. New York: Springer-Verlag.

Bourjault, A. 1984. “Contribution a une approche méthodologique de l’assemblage automatisé: elaboration automatiquede séquences opératoires.” Thése d’état. Besançon, France: Université de Franche-Comté.

Homem de Mello, L. S., and A. C. Sanderson. 1990. “AND/OR Graph Representation of Assembly Plans.” IEEE Transactions on Robotics and Automation 6 (2): 188–199.

Suszyn´ski M, Z_ urek J, Legutko S (2014) Modelling of assembly sequences using hypergraph and directed graph. Tech Gaz 3651:1229–1233.

Liu X, Liu Y, Xu B (2013) A converse method-based approach for assembly sequence planning with assembly tool. Int J Adv Manuf Technol 69(5–8):1359–1371.

Rashid MFF, Tiwari A, Hutabarat W (2011) An integrated representation scheme for assembly sequence planning and assembly line balancing. In: Proceedings of the 9th International Conference on Manufacturing Research ICMR 2011, September, pp 125–131.

Rekiek, B., A. Dolgui, A. Delchambre, and A. Bratcu. 2002. “State of Art of Optimization Methods for Assembly Line Design.” Annual Reviews in Control 26 (2): 163–174. doi:10.1016/S1367-5788(02)00027-5.

Riggs, R. J., and S. J. Hu. 2013. ‘Disassembly Liaison Graphs Inspired by Word Clouds.’ Procedia CIRP 7: 521–526.

Papakostas, N., G. Michalos, S. Makris, D. Zouzias, and G. Chryssolouris. 2011. “Industrial Applications with Cooperating Robots for the Flexible Assembly.” International Journal of Computer Integrated Manufacturing 24 (7): 650–660.

Tseng H-E, Tang C-E (2006) A sequential consideration for assembly sequence planning and assembly line balancing using the connector concept. Int J Prod Res 44(1):97–116.

Su YY, Liang D, Dong H (2014) Connector structure-based modeling of assembly sequence planning. Appl Mech Mater 496–500:2729–2732.

Chen S-F, Liu Y-J (2001) An adaptive genetic assembly-sequence planner. Int J Comput Integr Manuf 14(5):489–500.

Liu SL, Tang M, Dong JX (2003) Solving geometric constraints with genetic simulated annealing algorithm. J Zhejiang Univ Sci 4(5):532–541.

Su Q, Lai S-J (2010) 3D geometric constraint analysis and its application on the spatial assembly sequence planning. Int J Prod Res 48(5):1395–1414.

Sinanoglu C, Bo¨rklu¨ HR (2005) An assembly sequence-planning system for mechanical parts using neural network. Assem Autom 25(1):38–52.

Yuan X (2002) An interactive approach of assembly planning. IEEE Trans Syst Man Cybern Part A Syst Hum 32(4):522–526.

Zhang J, Sun J, He Q (2010) An approach to assembly sequence planning using ant colony optimization. In: 2010 International conference on intelligent control and information processing, August, pp 230-233.

Pan C, Smith SS and Smith GC. Determining interference between parts in CAD STEP files for automatic assembly planning. J Comput Inf Sci Eng 2005; 5: 56–62.

Boothroyd G, Dewhurst P and Knight WA. Product design for manufacture and assembly. Revised and expanded. CRC press, 22 June 2002.

Kuo TC, Huang SH and Zhang HC. Design for manufacture and design for ‘X’: concepts, applications, and perspectives. Comput Ind Eng 2001; 41: 241–260.

Prenting TO and Battaglin RM. The precedence diagram: a tool for analysis in assembly line balancing. J Ind Eng 1964; 15: 208–213.

Bullinger HJ and Ammer ED. Computer-aided depicting of precedence diagrams – a step towards efficient planning in assembly. Comput Ind Eng 1984; 8: 165–169.

Mathew AT and Rao CS. A novel method of using API to generate liaison relationships from an assembly. J Softw Eng Appl 2010; 3: 167.

Bahubalendruni MR, Biswal BB, Kumar M, et al. A note on mechanical feasibility predicate for robotic assembly sequence generation. In: CAD/CAM, robotics and factories of the future 2016, pp. 397–404. New Delhi: Springer.

Bahubalendruni MR and Biswal BB. Computer aid for automatic liaisons extraction from cad based robotic assembly. In: 8th IEEE international conference on intelligent systems and control (ISCO), Karpagam College of Engineering, Coimbatore, India, 10 January 2014, pp. 42–45. IEEE.

Bahubalendruni MV and Biswal BB. An algorithm to test feasibility predicate for robotic assemblies. Trends Mech Eng Technol 2014; 4: 11–16.

Linn RJ and Liu H. An automatic assembly liaison extraction method and assembly liaison model. IIE Trans 1999; 31: 353–363.

Smith SS, Smith GC and Liao X. Automatic stable assembly sequence generation and evaluation. J Manuf Syst 2001; 20: 225–235.

Soltani M, Panichella A, Van Deursen A (2017) A guided genetic algorithm for automated crash reproduction. In Proceedings of the 39th international conference on software engineering, Buenos Aires, Argentina, May 20–28, 2017, pp 209–22.

Kumar ER, Annamalai K (2015) Recent nontraditional optimization techniques for Assembly sequence planning. Int J Pure Appl Math 101(5):707–71.

Mahmoodabadi MJ, Nemati AR (2016) A novel adaptive genetic algorithm for global optimization of mathematical test functions and real-world problems. Eng Sci Technol Int J 19(4):2002–2021.

Hong T, Peng Y, Lin W, Wang S (2017) Empirical comparison of level-wise hierarchical multi-population genetic algorithm. J Inf Telecommun 1(1):66–78.

Mathew AT, Rao CSP (2014) Implementation of genetic algorithm to optimize the assembly sequence plan based on penalty function. ARPN J Eng Appl Sci 9(4):453–456.

Yasin A, Puteh N (2010) Product assembly sequence optimization based on genetic algorithm. Int J Comupt Sci Eng 02(09):3065–307.

Bonneville F, Perrard C and Henrioud JM. A genetic algorithm to generate and evaluate assembly plans. In: INRIA/IEEE symposium on emerging technologies and factory automation. ETFA'95 Proceedings, Vellore Institute of technology, Vellore, chennai, India, 10 October 1995, Vol. 2, pp. 231–239. IEEE.

Wong H and Leu MC. Adaptive genetic algorithm for optimal printed circuit board assembly planning. CIRP Ann Manuf Technol 1993; 42: 17–20.

Hong DS and Cho HS. A genetic-algorithm-based approach to the generation of robotic assembly sequences. Control Eng Pract 1999; 7: 151–159.

Chen SF and Liu YJ. The application of multi-level genetic algorithms in assembly planning. J Ind Technol 2001; 17: 1–9.

Smith GC and Smith SS. An enhanced genetic algorithm for automated assembly planning. Robot Comput-Integr Manuf 2002; 18: 355–364.

Chen SF and Liu YJ. An adaptive genetic assemblysequence planner. Int J Comput Integr Manuf 2001; 14: 489–500.

Senin N, Groppetti R and Wallace DR. Concurrent assembly planning with genetic algorithms. Robot Comput-Integr Manuf 2000; 16: 65–72.

Lazzerini B and Marcelloni F. A genetic algorithm for generating optimal assembly plans. Artif Intell Eng 2000; 14: 319–329.

Valle C, Gasca R, Toro M, et al. A genetic algorithm for assembly sequence planning. International Work- Conference on Artificial Neural Networks, Springer, Berlin, Heidelberg. 2003, pp.337–344.

Marian RM, Luong LH and Abhary K. Assembly sequence planning and optimisation using genetic algorithms: part I. Automatic generation of feasible assembly sequences. Appl Soft Comput 2003; 2: 223–253.

Tseng HE, Li JD and Chang YH. Connector-based approach to assembly planning using a genetic algorithm. Int J Prod Res 2004; 42: 2243–2261.

Bai YW, Chen ZN, Bin HZ, et al. An effective integration approach toward assembly sequence planning and evaluation. Int J Adv Manuf Technol 2005; 27: 96–105.

Hui W, Dong X and Guanghong D. A genetic algorithm for product disassembly sequence planning. Neurocomputing 2008; 71: 2720–2726.

Huang J, Du PA and Liao WZ. Genetic algorithm for assembly sequences planning based on assembly constraint. Comp Integr Manufac Syst – Beijing 2007; 13: 756.

Ping Duan YA (2016) Research on an improved ant colony optimization algorithm and its application. Int J Hybrid Inf Technol 9(4):223–234.

Sivakumar P, Elakia K (2016) A survey of ant colony optimization. Int J Adv Res Comput Sci Softw Eng 6(3):574–578.

Wang JF, Liu JH, Zhong YF (2004) A novel ant colony algorithm for assembly sequence planning. Int J Adv Manuf Technol 25(11–12):1137–1143.

Shuang B, Chen J, Li Z (2007) Microrobot based micro-assembly sequence planning with hybrid ant colony algorithm. Int J Adv Manuf Technol 38(11–12):1227–1235.

Yan C, Zhunxia L, Sen C (2015) A fixture assembly sequence planning method based on ant colony algorithm. In: International industrial informatics and computer engineering conference (IIICEC 2015), pp 559–562.

Guntsch M, Middendort M, Schmeck H, Middendorf M, Schmeck H (2001) An ant colony optimization approach to dynamic TSP. In: Conference on genetic and evolutionary computation, pp 860–867.

Hui L, Jingxiao Z, Lieyan R, Zhen S (2013) Scheduling optimization of construction engineering based on ant colony optimized hybrid genetic algorithm. J Netw 8(6):1411–1416.

Sharma S, Biswal BB, Dash P, et al. Generation of optimized robotic assembly sequence using ant colony optimization. In: IEEE international conference on automation science and engineering, Indian Institute of Science, Bangalore, India, 23 August 2008, pp. 894– 899. IEEE.

Shi SC, Li R, Fu YL, et al. Assembly sequence planning based on improved ant colony algorithm. Comp Integr Manuf Syst 2010; 16: 1189–1194.

Wang H, Rong Y and Xiang D. Mechanical assembly planning using ant colony optimization. Comput Aid Des 2014; 47: 59–71.

Lu C and Yang Z. Integrated assembly sequence planning and assembly line balancing with ant colony optimization approach. Int J Adv Manuf Technol 2016; 83: 243–256.

Yu J and Wang C. A max-min ant colony system for assembly sequence planning. Int J Adv Manuf Technol 2013; 67: 2819–2835.

Chen CP and Pao YH. An integration of neural network and rule-based systems for design and planning of mechanical assemblies. IEEE Trans Syst Man Cybernet 1993; 23: 1359–1371.

Hong DS and Cho HS. Optimization of robotic assembly sequences using neural network. In: IEEE/RSJ international conference on intelligent robots and systems' 93, IROS'93, Conference Center in Pacific Convention Plaza Yokohama, Yokohama, Japan, 26 July 1993, Vol. 1, pp. 232–239. IEEE.

Hong DS and Cho HS. A neural-network-based computational scheme for generating optimized robotic assembly sequences. Eng Appl Artif Intell 1995; 8: 129–145.

Chen WC, Tai PH, Deng WJ, et al. A three-stage integrated approach for assembly sequence planning using neural networks. Expert Syst Appl 2008; 34: 1777–1786.

Hong DS and Cho HS. Generation of robotic assembly sequences with consideration of line balancing using simulated annealing. Robotica 1997; 15: 663–673.

Hong DS and Cho HS. Generation of robotic assembly sequences using a simulated annealing. In: IEEE/RSJ international conference on intelligent robots and systems, Kyongju, South Korea, 1999, Vol. 2, pp. 1247–1252. IEEE.

Huang FY, Tseng YJ (2011) An integrated design evaluation and assembly sequence planning model using a particle swarm optimization approach. World Acad Sci Eng Technol 77:416–421.

Lv H, Lu C (2010) An assembly sequence planning approach with a discrete particle swarm optimization algorithm. Int J Adv Manuf Technol 50(5–8):761–770.

Yu H, Yu JP, Zhang WL (2009) An particle swarm optimization approach for assembly sequence planning. Appl Mech Mater 16–19:1228–1232.

Liu J, Wang Y, Gu Z (2008) Generation of optimal assembly sequences using particle swarm optimization, pp 1–8.

Ibrahim I, Ahmad H, Ibrahim Z, Mat Jusoh MF, Yusof ZM, Nawawi SW, Khalil K, Rahim MAA (2014) Multi-state particle swarm optimization for discrete combinatorial optimization problem. Int J Simul Syst Sci Technol 15(1):15–25.

Li JG, Yao YX, Gao D, Liu CQ, Yuan ZJ (2008) Cutting parameters optimization by using particle swarm optimization (PSO). Appl Mech Mater 12:879–883.

Yu H, Wang CE, Yu JP, et al. Assembly sequence planning based on particle swarm optimization algorithm for complex product. J North-East Univ (Nat Sci) 2010; 2: 28.

Tseng YJ, Yu FY and Huang FY. A green assembly sequence planning model with a closed-loop assembly and disassembly sequence planning using a particle swarm optimization method. Int J Adv Manuf Technol 2011; 57: 1183–1197.

Cao PB and Xiao RB. Assembly planning using a novel immune approach. Int J Adv Manuf Technol 2007; 31: 770–782.

Chang CC, Tseng HE and Meng LP. Artificial immune systems for assembly sequence planning exploration. Eng Appl Artif Intell 2009; 22: 1218–1232.

Biswal BB, Deepak BB and Rao Y. Optimization of robotic assembly sequences using immune based technique. J Manuf Technol Manage 2013; 24: 384–396.

Tseng HE, Wang WP and Shih HY. Using memetic algorithms with guided local search to solve assembly sequence planning. Expert Syst Appl 2007; 33: 451–467.

Zeng C, Gu T, Zhong Y, et al. A multi-agent evolutionary algorithm for connector-based assembly sequence planning. Proc Eng 2011; 15: 3689–3693.

Gao L, Qian W, Li X, et al. Application of memetic algorithm in assembly sequence planning. Int J Adv Manuf Technol 2010; 49: 1175–1184.

Zhou W, Yan J, Li Y, et al. Imperialist competitive algorithm for assembly sequence planning. Int J Adv Manuf Technol 2013; 67: 9–12.

Chen RS, Lu KY and Yu SC. A hybrid genetic algorithm approach on multi-objective of assembly planning problem. Eng Appl Artif Intell 2002; 15: 447–457.

Shan H, Zhou S and Sun Z. Research on assembly sequence planning based on genetic simulated annealing algorithm and ant colony optimization algorithm. Assembly Autom 2009; 29: 249–256.

Ning LH and Gu TL. Hybrid algorithm for assembly sequence planning. Comput Integr Manuf Syst – Beijing 2007; 13: 762.

Shuang B, Chen J and Li Z. Micro robot based microassembly sequence planning with hybrid ant colony algorithm. Int J Adv Manuf Technol 2008; 38: 1227–1235.

Tseng HE, Chen MH, Chang CC, et al. Hybrid evolutionary multi-objective algorithms for integrating assembly sequence planning and assembly line balancing. Int J Prod Res 2008; 46: 5951–5977.

Zhou W, Zheng JR, Yan JJ, et al. A novel hybrid algorithm for assembly sequence planning combining bacterial chemo taxis with genetic algorithm. Int J Adv Manuf Technol 2011; 52: 715–724.

Li M, Wu B, Hu Y, et al. A hybrid assembly sequence planning approach based on discrete particle swarm optimization and evolutionary direction operation. Int J Adv Manuf Technol 2013; 68: 617–630.

Zhang H, Liu H and Li L. Research on a kind of assembly sequence planning based on immune algorithm and particle swarm optimization algorithm. Int J Adv Manuf Technol 2014; 71: 795–808.

Huang YF and Lee CG. A framework of knowledgebased assembly planning. In: IEEE international conference on robotics and automation, Sacramento, CA, USA, 9 April 1991, pp. 599–604. IEEE.

Dong T, Tong R, Zhang L, et al. A knowledge-based approach to assembly sequence planning. Int J Adv Manuf Technol 2007; 32: 1232–1244.

Hsu YY, Tai PH, Wang MW, et al. A knowledgebased engineering system for assembly sequence planning. Int J Adv Manuf Technol 2011; 55: 763–782.

Zha XF, Lim SY and Fok SC. Integrated knowledgebased assembly sequence planning. Int J Adv Manuf Technol 1998; 14: 50–64.

Kashkoush M and ElMaraghy H. Knowledge-based model for constructing master assembly sequence. J Manuf Syst 2015; 34: 43–52.

Belhadj I, Trigui M and Benamara A. Subassembly generation algorithm from a CAD model. Int J Adv Manuf Technol 2016; 87: 2829–2840.

Lu C, Yang Z (2016) Integrated assembly sequence planning and assembly line balancing with ant colony optimization approach. Int J Adv Manuf Technol 83(1):243–256

Meng Y (2016) An application of the geometric method to assembly sequence planning. Int J Simul Syst Sci Technol 17(19):17.1–17.4.

Li M, Wu B, Hu Y, Jin C, Shi T (2013) A hybrid assembly sequence planning approach based on discrete particle swarm optimization and evolutionary direction operation. Int J Adv Manuf Technol 68(1–4):617–630.

Mishra A, Deb S (2016) An intelligent methodology for assembly tools selection and assembly sequence optimisation. In: Mandal DK, Syan CS (eds) CAD/CAM, robotics and factories of the future. Springer India, New Delhi, pp 323–333.

Lu C, Wong YS, Fuh JYH (2006) An enhanced assembly planning approach using a multi-objective genetic algorithm. Proc Inst Mech Eng Part B J Eng Manuf 220(2):255–272.

Lv H, Lu C (2009) A discrete particle swarm optimization algorithm for assembly sequence planning. In: 2009 8th international conference on reliability, maintainability and safety, pp 1119–1122.

Raja P, Pugazhenthi S (2012) Optimal path planning of mobile robots: A review. International Journal of Physical Sciences 7: 1314-1320.

Kala, R. On-Road Intelligent Vehicles: Motion Planning for Intelligent Transportation Systems; Butterworth-Heinemann: Oxford, UK, 2016.

Liu, C.; Atkeson, C.G.; Feng, S.; Xinjilefu, X. Full-body motion planning and control for the car egress task of the DARPA robotics challenge. In Proceedings of the 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, South Korea, 3–5 November 2015; pp. 527–532.

Halperin, D.; Salzman, O.; Sharir, M. Algorithmic motion planning. In Handbook of Discrete and Computational Geometry; Chapman and Hall/CRC: Boca Raton, FL, USA, 2017; pp. 1311–1342.

Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. Robot. Auton. Syst. 2016, 86, 13–28.

Tamizi, M.G.; Yaghoubi, M.; Najjaran, H. A review of recent trend in motion planning of industrial robots. Int. J. Intell. Robot. Appl. 2023, 7, 253–274.

Sun,W.; Torres, L.G.; Van Den Berg, J.; Alterovitz, R. Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In Robotics Research; Springer: Cham, Switzerland, 2016; pp. 685–701.

Roozegar, M.; Mahjoob, M.; Jahromi, M. Optimal motion planning and control of a nonholonomic spherical robot using dynamic programming approach: Simulation and experimental results. Mechatronics 2016, 39, 174–184.

Nguyen, Q.V.; Colas, F.; Vincent, E.; Charpillet, F. Motion planning for robot audition. Auton. Robot. 2019, 43, 2293–2317.

Mccrackin, D.C.; Johnson, S.W. Syntactic Inferential Motion Planning Method for Robotic Systems. U.S. Patent 9,855,657, 2 January 2018.

Yang, L.; Qi, J.; Song, D.; Xiao, J.; Han, J.; Xia, Y. Survey of robot 3D path planning algorithms. J. Control Sci. Eng. 2016, 2016, 7426913.Gillani, M.; Akbari, A.; Rosell, J. Physics-based motion planning: Evaluation criteria and benchmarking. In Proceedings of the Robot 2015: Second Iberian Robotics Conference, Lisbon, Portugal, 19–21 November 2015; pp. 43–55.

Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Path planning and trajectory planning algorithms: A general overview. In Motion and Operation Planning of Robotic Systems; Springer: Cham, Switzerland, 2015; pp. 3–27.

Contreras-Cruz, M.A.; Ayala-Ramirez, V.; Hernandez-Belmonte, U.H. Mobile robot path planning using artificial bee colony and evolutionary programming. Appl. Soft Comput. 2015, 30, 319–328.

Holladay, R.; Lozano-Pérez, T.; Rodriguez, A. Force-and-Motion Constrained Planning for Tool Use. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019.

Schmerling, E.; Janson, L.; Pavone, M. Optimal sampling-based motion planning under differential constraints: The driftless case. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle,WA, USA, 26–30 May 2015; pp. 2368–2375.

Sivaramakrishnan, A.; Littlefield, Z.; Bekris, K.E. Towards Learning Efficient Maneuver Sets for Kinodynamic Motion Planning. arXiv 2019, arXiv:1907.07876.

Elbanhawi, M. Randomised Parameterisation Motion Planning for Autonomous Cars. Ph.D. Thesis, RMIT University, Melbourne, Australia, 2016.

H. M. Choset, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, 2005.

S. M. LaValle, Planning Algorithms, Cambridge University Press, 2006.

J.-C. Latombe, Robot Motion Planning, vol. 124, Kluwer Academic, Norwell, Mass, USA, 1991.

Siciliano, B., Khatib, O.: Springer Handbook of Robotics. Springer, Berlin, Heidelberg (2016).

Meyes, R., et al.: Motion planning for industrial robots using reinforcement learning. Procedia CIRP. 63, 107–112 (2017).

Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4, 100–107 (1968).

Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. 5, 90–98 (1986).

LaValle, S.M., Kuffner, J.J., Jr.: Randomized kinodynamic planning. Int. J. Robot. Res. 20, 378–400 (2001).

E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," Robotics and Autonomous Systems, vol. 61, no. 12, pp. 1258–1276, 2013.

S. Flemming, C.-H. Anders la, and B. Morten, Configuration Space and Visibility Graph Generation from Geometric Workspaces for UAVs, Book Section 4, American Institute of Aeronautics and Astronautics, 2011.

LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. Research Report 9811.

L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996.

E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol. 1, pp. 269–271, 1959.

P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

Stentz, "Optimal and efficient path planning for partially known environments," in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3310–3317, San Diego, Calif, USA, May 1994.

Zefran, M., Kumar, V., & Croke, C. B. (1998). On the generation of smooth three-dimensional rigid body motions. IEEE Transactions on Robotics and Automation, 14(4), 576-589.

Meyer-Delius, D., Beinhofer, M., & Burgard, W. (2012). Occupancy grid models for robot mapping in changing environments. In Proceedings of the AAAI conference on artificial intelligence (Vol. 26, No. 1, pp. 2024-2030).

Majumdar, A., Ahmadi, A. A., & Tedrake, R. (2013, May). Control design along trajectories with sums of squares programming. In 2013 IEEE International Conference on Robotics and Automation (pp. 4054-4061). IEEE.

Pham, Q.-C.: Trajectory planning. In: Handbook of Manufacturing Engineering and Technology, pp. 1873–1887 (2015).

Piazzi, A., Visioli, A.: Global minimum-time trajectory planning of mechanical manipulators using interval analysis. Int. J. Control 71(4), 631–652 (1998).

Piazzi, A., Visioli, A.: Global minimum-jerk trajectory planning of robot manipulators. IEEE Trans. Industr. Electron. 47(1), 140– 149 (2000).

Lo Bianco, C.G., Piazzi, A.: Minimum-time trajectory planning of mechanical manipulators under dynamic constraints. Int. J. Control 75(13), 967–980 (2002).

Zhang, T., Zhang, M., Zou, Y.: Time-optimal and smooth trajectory planning for robot manipulators. Int. J. Control Autom. Syst. 19(1), 521–531 (2021).

Carabin, G., Scalera, L.: On the trajectory planning for energy efficiency in industrial robotic systems. Robotics 9(4), 89 (2020).

Field, G., Stepanenko, Y.: Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 3. IEEE, pp. 2755–2760 (1996).

Lu, S., Ding, B., Li, Y.: Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation. Adv. Mech. Eng. 12(3), 1687814020913667 (2020).

Lynch, K.M., Park, F.C.: Modern Robotics. Cambridge University Press, Cambridge (2017).

Gilbert, E.G., Johnson, D.W., Keerthi, S.S.: A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE J Robot Automat 4(2), 193–203 (1988).

Hubbard, P.M.: Approximating polyhedra with spheres for time-critical collision detection. ACM Trans Gr (TOG) 15(3), 179–210 (1996).

Kleinbort, M., Salzman, O., Halperin, D.: Collision detection or nearest-neighbor search? on the computational bottleneck in sampling-based motion planning (2016). arXiv preprint arXiv: 1607. 04800.

Pan, J., Manocha, D.: Efficient configuration space construction and optimization for motion planning. Engineering 1(1), 046–057 (2015).

Huh, J., Lee, D.D.: Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 63–69 (2016).

Pan, J., Manocha, D.: Fast probabilistic collision checking for sampling- based motion planning using locality-sensitive hashing. Int. J. Robot. Res. 35(12), 1477–1496 (2016).

Das, N., Yip, M.: Learning-based proxy collision detection for robot motion planning applications. IEEE Trans. Rob. 36(4), 1096– 1114 (2020).

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. The international journal of robotics research, 30(7), 846-894.

Phiquepal, C.; Toussaint, M. Combined task and motion planning under partial observability: An optimization-based approach. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 9000–9006.

Dantam, N.T.; Chaudhuri, S.; Kavraki, L.E. The Task-Motion Kit: An Open Source, General-Purpose Task and Motion-Planning Framework. IEEE Robot. Autom. Mag. 2018, 25, 61–70.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. The international journal of robotics research, 5(1), 90-98.

Deits, R., & Tedrake, R. (2014, November). Footstep planning on uneven terrain with mixed-integer convex optimization. In 2014 IEEE-RAS international conference on humanoid robots (pp. 279-286). IEEE.

Luchnikov, V. A., Medvedev, N. N., Oger, L., & Troadec, J. P. (1999). Voronoi-Delaunay analysis of voids in systems of nonspherical particles. Physical review E, 59(6), 7205.

Koenig, S., & Likhachev, M. (2002, May). Improved fast replanning for robot navigation in unknown terrain. In Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292) (Vol. 1, pp. 968-975). IEEE.

Koenig, S., & Likhachev, M. (2005). Fast replanning for navigation in unknown terrain. IEEE transactions on robotics, 21(3), 354-363.

Chamseddine, Y. Zhang, C. A. Rabbath, C. Join, and D. Theilliol, “Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle,” IEEE Transactions on Aerospace and Electronic Systems, vol. 48, no. 4, pp. 2832–2848, 2012.

Yue, R., Xiao, J., Wang, S., & Joseph, S. L. (2009, December). Modeling and path planning of the city-climber robot part II: 3D path planning using mixed integer linear programming. In 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO) (pp. 2391-2396). IEEE.

Masehian, E., & Habibi, G. (2007). Robot path planning in 3D space using binary integer programming. International Journal of Computer and Information Engineering, 1(5), 1255-1260.

Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press.

Moscato, P., & Norman, M. G. (1992). A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. Parallel Computing and Transputer Applications, 1, 177–186.

Kennedy, J. (2010). Particle swarm optimization. In Encyclopedia of Machine Learning (section 630, pp. 760–766). Springer US.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 26(1), 29–41.

Eusuff, M. M., & Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources Planning and Management, 129(3), 210–225.

Chitta, S., Sucan, I., & Cousins, S. (2012). Moveit![ros topics]. IEEE Robotics & Automation Magazine, 19(1), 18–19.

Karaman, S., & Frazzoli, E. (2010, December). Optimal kinodynamic motion planning using incremental sampling-based methods. In Proceedings of the 49th IEEE Conference on Decision and Control (CDC '10) (pp. 7681–7687). Atlanta, GA, USA.Yershova.

L. Jaillet, T. Simeon, and S. M. LaValle, "Dynamic domain RRTs: efficient exploration by controlling the sampling domain," in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3856–3861, Barcelona, Spain, April 2005.

S. Choudhury, S. Scherer, and S. Singh, "Rrt∗-ar: sampling based alternate routes planning with applications to autonomous emergency landing of a helicopter," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '13), pp. 3947–3952, IEEE, Karlsruhe, Germany, 2013.

L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration space for fast path planning," in Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 2138–2145, May 1994.

N. M. Amato, O. B. Bayazit, and L. K. Dale, "OBPRM: an obstacle-based PRM for 3D workspaces," in Proceedings of the 3rd Workshop on the Algorithmic Foundations of Robotics on Robotics : The Algorithmic Perspective: The Algorithmic Perspective (WAFR '98), pp. 155–168, 1998.

D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," The International Journal of Robotics Research, vol. 21,no. 3, pp. 233–255, 2002.

F. Yan, Y.-S. Liu, and J.-Z. Xiao, "Path planning in complex 3D environments using a probabilistic roadmap method," International Journal of Automation and Computing, vol. 10, no. 6, pp. 525–533, 2013.

R. Gayle, A. Sud, M. C. Lin, and D. Manocha, “Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments,” in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS ’07), pp. 3777–3783, San Diego, Calif, USA, November 2007.

M. I. Shamos and D. Hoey, “Closest-point problems,” in Proceedings of the 16th Annual Symposium on Foundations of Computer Science, pp. 151–162, Berkeley, Calif, USA, 1975.

L. Lifeng and Z. Shuqing, “Voronoi diagram and gis-based 3d path planning,” in Proceedings of the 17th International Conference on Geoinformatics, vol. 5, pp. 1–5, Fairfax, Va, USA, 2009.

E. Masehian and M. R. Amin-Naseri, “A voronoi diagram visibility graph-potencial field compound algorith for robot path planning,” Journal of Robotic Systems, vol. 21,no. 6, pp.275– 300, 2004.

Y. K. Hwang and N. Ahuja, “A potential field approach to path planning,” IEEE Transactions on Robotics and Automation, vol. 8, no. 1, pp. 23–32, 1992.

T. Roos and H. Noltemeier, Dynamic Voronoi Diagrams in Motion Planning, vol. 553 of Lecture Notes in Computer Science, Book Section 17, Springer, Berlin, Germany, 1991.

C. I. Connolly, J. B. Burns, and R. Weiss, “Path planning using Laplace’s equation,” in Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 2102–2106, Cincinnati, Ohio, USA, May 1990.

E. Rimon and D. E. Koditschek, “Exact robot navigation using artificial potential functions,” IEEE Transactions on Robotics and Automation, vol. 8, no. 5, pp. 501–518, 1992.

S. Sundar and Z. Shiller, “Optimal obstacle avoidance based on the hamilton-jacobi-bellman equation,” in Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 2424–2429, San Diego, Calif, USA, 1994.

R. Olfati-Saber, “Flocking for multi-agent dynamic systems: algorithms and theory,” IEEE Transactions on Automatic Control, vol. 51, no. 3, pp. 401–420, 2006.

S. S. Ge and Y. J. Cui, “New potential functions for mobile robot path planning,” IEEE Transactions on Robotics and Automation, vol. 16, no. 5, pp. 615–620, 2000.

R. Geraerts, “Planning short paths with clearance using explicit corridors,” in Proceedings of the IEEE International Conference on Robotics andAutomation (ICRA’10), pp. 1997–2004, Anchorage, Alaska, USA, 2010.

F. B. Zhan and C. E. Noon, “Shortest path algorithms: an evaluation using real road networks,” Transportation Science, vol. 32, no. 1, pp. 65–73, 1998.

L. Verscheure, L. Peyrodie, N.Makni, N. Betrouni, S.Maouche, and M. Vermandel, “Dijkstra’s algorithm applied to 3D skeletonization of the brain vascular tree: evaluation and application to symbolic,” in Proceedings of the Annual International Conference of the IEEE Engineering

in Medicine and Biology Society (EMBC ’10), pp. 3081–3084, Buenos Aires, Argentina, September 2010.

M. Wan, Z. Liang, Q. Ke, L. Hong, I. Bitter, and A. Kaufman, “Automatic centerline extraction for virtual colonoscopy,” IEEE Transactions on Medical Imaging, vol. 21, no. 12, pp. 1450–1460, 2002.

A. Musliman, A. A. Rahman, and V. Coors, “Implementing 3d network analysis in 3d-gis,” in Proceedings of the 21st ISPRS Congress Silk Road for Information from Imagery, vol. 8, pp. 113– 120, Beijing, China, 2008.

P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

L. Niu and G. Zhuo, “An improved real 3d a∗ algorithm for difficult path finding situation,” in Proceeding of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 37, Beijing, China, 2008.

B. C. Williams and R. J. Ragno, “Conflict-directed A∗ and its role in model-based embedded systems,” Discrete Applied Mathematics, vol. 155, no. 12, pp. 1562–1595, 2007.

Nash, K. Daniel, S. Koenig, and A. Felner, “Theta∗: Any angle path planning on grids,” in Proceedings of the National Conference on Artificial Intelligence, vol. 22, pp. 1177–1198, AAAI Press, MIT Press, Vancouver, Canada, 2007.

Nash, S. Koenig, and C. Tovey, “Lazy theta∗: any-angle path planning and path length analysis in 3D,” in Proceedings of the Third Annual Symposium on Combinatorial Search, vol. 2, pp. 153–154, Atlanta, Ga, USA, 2010.

L. De Filippis, G. Guglieri, and F. Quagliotti, “Path planning strategies for UAVS in 3D environments,” Journal of Intelligent and Robotic Systems, vol. 65, no. 1–4, pp. 247–264, 2012.

M. Likhachev and D. Ferguson, “Planning long dynamically feasible maneuvers for autonomous vehicles,” The International Journal of Robotics Research, vol. 28, no. 8, pp. 933–945, 2009.

Stentz, “The focussed d-star algorithm for real-time replanning,” in Proceedings of the International Joint Conference on AI, vol. 95, pp. 1652–1659, Montreal, Canada, 1995.

Smirnov, Y. V. (1997). Hybrid algorithms for on-line search and combinatorial optimization problems (Doctoral dissertation, Carnegie Mellon University).

Tovey, S. Greenberg, and S. Koenig, “Improved analysis of D∗,” in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3371–3378, Taipei, Taiwan, September 2003.

Miller, K. Stepanyan, A. Miller, and M. Andreev, “3D path planning in a threat environment,” in Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC ’11), vol. 6, pp. 6864–6869, IEEE, Orlando, Fla, USA, December 2011.

L. Shih, T.-T. Lee, andW. A. Gruver, “A unified approach for robot motion planning with moving polyhedral obstacles,” IEEE Transactions on Systems,Man and Cybernetics, vol. 20, no. 4, pp. 903–915, 1990.

K. Culligan, M. Valenti, Y. Kuwata, and J. P. How, “Three dimensional flight experiments using on-line mixed-integer linear programming trajectory optimization,” in Proceedings of the American Control Conference (ACC ’07), vol. 6, pp. 5322– 5327, New York, NY, USA, July 2007.

R. Bhattacharya, “Optragen: a matlab toolbox for optimal trajectory generation,” in Proceedings of the 45th IEEE Conference on Decision and Control, pp. 6832–6836, San Diego, Calif, USA, December 2006.

S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma, “An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance Scenarios,” International Journal of Vehicle Autonomous Systems, vol. 8, no. 2–4, pp. 190–216, 2010.

C. S. Ma and R. H. Miller, “Milp optimal path planning for real-time applications,” in Proceedings of the American Control Conference, p. 6, Minneapolis, Minn, USA, 2006.

J. Dong and V. Juris, Parallel Evolutionary Algorithms for UAV Path Planning, section 1, American Institute of Aeronautics and Astronautics, 2004.

C. Fonlupt, P. Preux, and D. Robilliard, “Preventing premature convergence via cooperating genetic algorithms,” in Proceedings of the ACM Workshop on Foundations of Genetic Algorithms, pp. 1–6, Citeseer, 1993.

Hacioglu and I. Ozkol, “Transonic airfoil design and optimisation by using vibrational genetic algorithm,” Aircraft Engineering and Aerospace Technology, vol. 75, no. 4, pp. 350– 357, 2003.

Hacioglu and I. Ozkol, “Vibrational genetic algorithm as a new concept in airfoil design,” Aircraft Engineering and Aerospace Technology, vol. 74, no. 3, pp. 228–236, 2002.

Y. V. Pehlivanoglu, “A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV,” Aerospace Science and Technology, vol. 16, no. 1, pp. 47–55, 2012.

G. Erinc and S. Carpin, “A genetic algorithm for nonholonomic motion planning,” in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1843–1849, Roma, Italy, April 2007.

M. Dorigo and M. Birattari, “Ant colony optimization,” in Encyclopedia ofMachine Learning, pp. 36–39, SpringerUS, 2010.

Y. Chen, X. Zhao, C. Zhang, and J. Han, “Relative coordination 3D trajectory generation based on the trimmed ACO,” in Proceedings of the International Conference on Electrical and Control Engineering (ICECE ’10), vol. 1, pp. 1531–1536,Wuhan, China, June 2010.

Y. Saber and A. M. Alshareef, “Scalable unit commitment by memory-bounded ant colony optimization with A∗ local search,” International Journal of Electrical Power and Energy Systems, vol. 30, no. 6-7, pp. 403–414, 2008.

R. Glasius, A. Komoda, and S. C. A. M. Gielen, “Neural network dynamics for path planning and obstacle avoidance,” Neural Networks, vol. 8, no. 1, pp. 125–133, 1995.

W. Ahn, R. S. Ramakrishna, C. G. Kang, and I. C. Choi, “Shortest path routing algorithm using Hopfield neural network,” Electronics Letters, vol. 37, no. 19, pp. 1176–1178, 2001.

Kassim and B. Kumar, “A neural network architecture for path planning,” in Proceedings of the International Joint Conference on Neural Networks (IJCNN ’92), vol. 2, pp. 787–792, IEEE, Baltimore, Md, USA, 1992.

J. Tang, J. Zhu, and Z. Sun, “A novel path planning approach based on appart and particle swarm optimization,” Advances in Neural Networks, vol. 3498, pp. 253–258, 2005.

Rahimi-Vahed and A. H. Mirzaei, “A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem,” Computers and Industrial Engineering, vol. 53, no. 4, pp. 642–666, 2007.

Yan, Y.-S. Liu, and J.-Z. Xiao, “Path planning in complex 3D environments using a probabilistic roadmap method,” International Journal of Automation and Computing, vol. 10, no. 6, pp. 525–533, 2013.

Schøler, A. la Cour-Harbo, and M. Bisgaard, “Generating approximative minimum length paths in 3D for UAVs,” in Proceedings of the IEEE Intelligent Vehicles Symposium (IV ’12), pp. 229–233, Madrid, Spain, June 2012.

X. Zhang, H. Duan, and Y. Yu, “Receding horizon control for multi-UAVs close formation control based on differential evolution,” Science China Information Sciences, vol. 53, no. 2, pp. 223–235, 2010.

Gwiazda A 2013, Designing of Technical Means in Integrative Methodological Approach Monografia GIG [in English].

Monica Z 2015 Optimization of the production process using virtual model of a workspace IOP Conf. Series: Materials Science and Engineering 95 012102.

Wittler G and Moritz W 1998 Mechatronic Design Methods and Software in Mechanical Engineering. 9. Symposium ”Fertigungsgerechtes konstruieren” Schnaittach 1-6.

Lückel J and Wallaschek J 1997 Functional Modelling and Simulation in Mechanical Design and Mechatronics 2nd MATHMOD Vienna.

Cheng F S 2000 A Methodology for Developing Robotic Workcell Simulation Models Proceedings of the 32nd Conference on Winter Simulation IEEE 1265-1271.

Aguiar A J C and Silva A S A and Villani E 2008 Graphic Robot Simulation for the Design of Work Cells in the Aeronautic Industry BCM Symposium Series in Mechatronics 3 346-354.

Grajo, E. S., Gunal, A., Sathyadev, D., & Ulgen, O. (1994). A uniform methodology for discrete-event and robotic simulation. In Proceedings of the Deneb Users Group Meeting (pp. 17-24).

Qureshi, A.H., Yip, M.C.: Deeply informed neural sampling for robot motion planning. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 6582–6588 (2018).

Pfeiffer, M., Schaeuble, M., Nieto, J., Siegwart, R., & Cadena, C. (2017, May). From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In 2017 ieee international conference on robotics and automation (icra) (pp. 1527-1533). IEEE.

Hamandi, M., D'Arcy, M., Fazli, P.: Deepmotion: learning to navigate like humans. In: Proceedings of the 2019 28th IEEE International Conference on Robot and Human Interactive Communication (ROMAN). pp. 1–7. IEEE, New Delhi (2019).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Bency, M.J., Qureshi, A.H., Yip, M.C.: Neural Path Planning: Fixed Time, Near-optimal Path Generation Via Oracle Imitation. arXiv preprint arXiv:1904.11102 (2019).

Kurutach, T., Tamar, A., Yang, G., Russell, S. J., & Abbeel, P. (2018). Learning plannable representations with causal infogan. Advances in Neural Information Processing Systems, 31.

Qureshi, A. H., Simeonov, A., Bency, M. J., & Yip, M. C. (2019, May). Motion planning networks. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 2118-2124). IEEE.

Qureshi, A. H., Miao, Y., Simeonov, A., & Yip, M. C. (2020). Motion planning networks: Bridging the gap between learning-based and classical motion planners. IEEE Transactions on Robotics, 37(1), 48-66.

Ichter, B., & Pavone, M. (2019). Robot motion planning in learned latent spaces. IEEE Robotics and Automation Letters, 4(3), 2407-2414.

Huh, J., Isler, V., & Lee, D. D. (2021, May). Cost-to-go function generating networks for high dimensional motion planning. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 8480-8486). IEEE.

Ichter, B., Schmerling, E., Lee, T. W. E., & Faust, A. (2020, May). Learned critical probabilistic roadmaps for robotic motion planning. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 9535-9541). IEEE.

Sarkar, M., Pradhan, P., & Ghose, D. (2019). Planning robot motion using deep visual prediction. arXiv preprint arXiv:1906.10182.

Yang, G., Zhang, A., Morcos, A., Pineau, J., Abbeel, P., & Calandra, R. (2020, July). Plan2vec: Unsupervised representation learning by latent plans. In Learning for Dynamics and Control (pp. 935-946). PMLR.

Nian, R., Liu, J., Huang, B.: A review on reinforcement learning: introduction and applications in industrial process control. Comput. Chem. Eng. 139, 106886 (2020)

Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. 57(5), 469–483 (2009).

Mukherjee, D., Gupta, K., Chang, L.H., Najjaran, H.: A survey of robot learning strategies for human-robot collaboration in industrial settings. Robot. Comput. Integr. Manuf. 73, 102231 (2022).

Calinon, S., Billard, A.: Active teaching in robot programming by demonstration. In: RO-MAN 2007—The 16th IEEE International Symposium on Robot and Human Interactive Communication. IEEE, pp. 702–707 (2007).

Ghahramani, Z.: Probabilistic machine learning and artificial intelligence. Nature 521(7553), 452–459 (2015).

Duque, D.A., Prieto, F.A., Hoyos, J.G.: Trajectory generation for robotic assembly operations using learning by demonstration. Robot. Comput. Integr. Manuf. 57, 292–302 (2019)

Y.U. Cao, A.S. Fukunaga, A. Kahng, Cooperative mobile robotics: Antecedents and directions, Auton. Robots 4 (1) (1997) 7–27.

B.P. Gerkey, M.J. Matarić, A formal analysis and taxonomy of task allocation in multi-robot systems, Int. J. Robot. Res. 23 (9) (2004) 939–954.

K. Lerman, C. Jones, A. Galstyan, M.J. Matarić, Analysis of dynamic task allocation in multi-robot systems, Int. J. Robot. Res. 25 (3) (2006) 225–241.

Khamis, A. Hussein, A. Elmogy, Multi-robot Task Allocation: A Review of the State-of-the-Art, in: Cooperative Robots and Sensor Networks 2015, vol. 604, 2015, pp. 31–51.

Quinton, C. Grand, C. Lesire, Market approaches to the multi-robot task allocation problem: a survey, J. Intell. Robot. Syst. 107 (29) (2023).

E. Nunes, M. Manner, H. Mitiche, M. Gini, A taxonomy for task allocation problems with temporal and ordering constraints, Robot. Auton. Syst. 90 (2017) 55–70.

Y. Rizk, M. Awad, E.W. Tunstel, Cooperative Heterogeneous Multi-Robot Systems: A Survey, ACM Comput. Surv. 52 (2) (2020) 1–31.

S. N., K.C. R.M., R. M.M., M.N. Janardhanan, Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems, Ind. Robot 47 (6) (2020) 929–942.

B.B. Werger, M.J. Mataric, Broadcast of local eligibility: behavior-based control for strongly cooperative robot teams, in: Proceedings of the Fourth International Conference on Autonomous Agents, 2000, pp. 21–22.

H.G. Tanner, Switched UAV-UGV cooperation scheme for target detection, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 3457–3462.

R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, G. Cai, A survey on inspecting structures using robotic systems, Int. J. Adv. Robot. Syst. 13 (6) (2016).

W. Zhao, Q. Meng, P.W.H. Chung, A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario, IEEE Trans. Cybern. 46 (4) (2016) 902–915.

K. Nagatani, Y. Okada, N. Tokunaga, K. Yoshida, S. Kiribayashi, K. Ohno, E. Takeuchi, S. Tadokoro, H. Akiyama, I. Noda, T. Yoshida, E. Koyanagi, Multi-robot exploration for search and rescue missions: A report of map building in RoboCupRescue 2009, in: 2009 IEEE International Workshop on Safety, Security & Rescue Robotics (SSRR 2009), 2009, pp. 1–6.

R.R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, A.M. Erkmen, Search and Rescue Robotics, in: Springer Handbook of Robotics, 2008, pp. 1151–1173.

L. Luo, N. Chakraborty, K. Sycara, Distributed algorithms for multirobot task assignment with task deadline constraints, IEEE Trans. Autom. Sci. Eng. 12 (3) (2015) 876–888.

J.K. Verma, V. Ranga, Multi-Robot Coordination Analysis, Taxonomy, Challenges and Future Scope, J. Intell. Robot. Syst. 102 (1) (2021) 10.

Liu, A. Kroll, A Centralized Multi-Robot Task Allocation for Industrial Plant Inspection by Using a* and Genetic Algorithms, in: Artificial Intelligence and Soft Computing, 2012, pp. 466–474.

S. Chopra, G. Notarstefano, M. Rice, M. Egerstedt, A distributed version of the Hungarian method for multirobot assignment, IEEE Trans. Robot. 33 (4) (2017) 932–947.

Kanakia, B. Touri, N. Correll, Modeling multi-robot task allocation with limited information as global game, Swarm Intelligence 10 (2) (2016) 147–160.

Stentz, R. Zlot, An auction-based approach to complex task allocation for multirobot teams (Ph.D. thesis), Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, 2006.

Fang, B., Zhang, Q., Wang, H. and Yuan, X. (2018), “Personality driven task allocation for emotional robot team”, International Journal of Machine Learning and Cybernetics, Vol. 9 No. 12, pp. 1955-1962.

Palmer, A.W., Hill, A.J. and Scheding, S.J. (2018), “Modelling resource contention in multi-robot task allocation problems with uncertain timing”, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE.

D’Emidio, M. and Khan, I. (2017), “Multi-robot task allocation problem: current trends and new ideas”, in: Joint Proceedings of the 18th Italian Conference on Theoretical Computer Science and the 32nd Italian Conference on Computational Logic (ICTCS/CILC).

Li, Z. and Li, X. (2017), “Research on model and algorithm of task allocation and path planning for multi-robot”, Open Journal of Applied Sciences, Vol. 7 No. 10, pp. 511.

Rishwaraj, G. and Ponnambalam, S. (2017), “Integrated trust based control system for multirobot systems: development and experimentation in real environment”, Expert Systems with Applications, Vol. 86, pp. 177-189.

Johnson, L.B., Choi, H.-L. and How, J.P. (2016), “The role of information assumptions in decentralized task allocation: a tutorial”, IEEE Control Systems Magazine, Vol. 36 No. 4, pp. 45-58.

Semwal, T., Jha, S.S. and Nair, S.B. (2017), “On ordering multi-robot task executions within a cyber physical system”, ACM Transactions on Autonomous and Adaptive Systems (Systems), Vol. 12No. 4, pp. 1-27.

Li, D., Fan, Q. and Dai, X. (2017a), “Research status of multirobot systems task allocation and uncertainty treatment”, in: J. Phys. Conf. Ser.

Sung, Y., Budhiraja, A.K., Williams, R.K. and Tokekar, P. (2018), “Distributed simultaneous action and target assignment for multi-robot multi-target tracking”, in: 2018 IEEE International conference on robotics and automation (ICRA), IEEE.

Giordani, S., Lujak, M. and Martinelli, F. (2010), “A distributed algorithm for the multi-robot task allocation problem”, in: International conference on industrial, engineering and other applications of applied intelligent systems, Springer.

Korsah, G. A., Stentz, A., & Dias, M. B. (2013). A comprehensive taxonomy for multi-robot task allocation. The International Journal of Robotics Research, 32(12), 1495-1512.

Darmanin, R.N. and Bugeja, M.K. (2017), “A review on multi-robot systems categorised by application domain”, in: 2017 25th Mediterranean Conference on Control and Automation (MED), IEEE.

L. Rabiner, Combinatorial optimization:Algorithms and complexity, IEEE Trans. Acoust. Speech Signal Process. 32 (6) (1984) 1258–1259.

Dantzig, A. Orden, P. Wolfe, The generalized simplex method for minimizing a linear form under linear inequality restraints, vol. 5, (2) 1955, pp. 183–195,

Y. Chen, U. Rosolia, A.D. Ames, Decentralized task and path planning for multi-robot systems, IEEE Robot. Autom. Lette. 6 (3) (2021) 4337–4344.

J.G. Martin, J.R.D. Frejo, R.A. García, E.F. Camacho, Multi-robot task allocation problem with multiple nonlinear criteria using branch and bound and genetic algorithms, vol. 14, (5) 2021, pp. 707–727.

Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.

X. Kong, Y. Gao, T. Wang, J. Liu, W. Xu, Multi-robot Task Allocation Strategy based on Particle Swarm Optimization and Greedy Algorithm, in: 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2019, pp. 1643–1646.

David, T. Rögnvaldsson, Multi-Robot Routing Problem with Min– Max Objective, Robotics 10 (4) (2021) 122.

M. Shelkamy, C.M. Elias, D.M. Mahfouz, O.M. Shehata, Comparative Analysis of Various Optimization Techniques for Solving Multi-Robot Task Allocation Problem, in: 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), 2020, pp. 538–543.

Garrido, S., Moreno, L., Abderrahim, M., Martin, F.: Path planning for mobile robot navigation using voronoi diagram and fast marching. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2376–2381 (2006).

Yao, J., Lin, C., Xie, X., Wang, A., Hung, C.C.: Path planning for virtual human motion using improved a* star algorithm. In: Seventh International Conference on Information Technology: New Generations (ITNG), 2010, pp. 1154–1158. IEEE (2010).

Kuffner, J.J., Lavalle, S.M.: RRT-Connect: An efficient approach to single-query path planning. In: IEEE International Conference On Robotics and Automation, pp. 995– 1001 (2000).

Masoud, A.A.: A harmonic potential approach for simultaneous planning and control of a generic UAV platform. J. Intell. Robot. Syst. 65(1–4), 153–173 (2012).

Doria, N.S.F., Freire, E.O., Basilio, J.C.: An algorithm inspired by the deterministic annealing approach to avoid local minima in artificial potential fields. In: 16th International Conference on Advanced Robotics (ICAR), 2013, pp. 1–6. IEEE (2013).

LaValle, S.: Planning Algorithms. Cambridge University Press (2006).

Dawande, M.W., Geismar, H.N., Sethi, S.P., Sriskandarajah, C.: Throughput Optimization in Robotic Cells, International Series in Operations Research & Management Science, vol. 101. Springer, US (2007).

Zhang, K., Collins Jr, E.G., Shi, D.: Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. ACM Trans. Auton. Adapt. Syst. (TAAS) 7(2), 21 (2012)

Zhang, K., Collins Jr, E.G., Barbu, A.: An efficient stochastic clustering auction for heterogeneous robotic collaborative teams. J. Intell. Robot. Syst. 72(3–4), 541–558 (2013).

Portugal, D., Rocha, R.: A survey on multi-robot patrolling algorithms. In: Technological Innovation for Sustainability, pp. 139–146. Springer (2011).

Maimon, O.: The robot task-sequencing planning problem. IEEE Trans. Robot. Autom. 6(6), 760–765 (1990).

Cao, T., Sanderson, A.C.: Task sequence planning in a robot workcell using and/or nets. In: Proceedings of the 1991 IEEE International Symposium on Intelligent Control, 1991, pp. 239–244. IEEE (1991).

Chien, S.Y., Xue, L.Q., Palakal, M.: Task planning for a mobile robot in an indoor environment using objectoriented domain information. IEEE Trans. Syst. Man Cybern. B 27(6), 1007–1016 (1997).

Chien, Y., Hudli, A., Palakal, M.: Using many-sorted logic in the object-oriented data model for fast robot task planning. J. Intell. Robot. Syst. 23(1), 1–25 (1998).

Galindo, C., Fernandez-Madrigal, J.A., Gonzalez, J.: Improving efficiency in mobile robot task planning through world abstraction. IEEE Trans. Robot. 20(4), 677–690 (2004).

Bhatia, A., Maly, M.R., Kavraki, E., Vardi, M.Y.: Motion planning with complex goals. IEEE Robot. Autom. Mag. 18(3), 55–64 (2011).

Gaschler, A., Petrick, R., Giuliani, M., Rickert, M., Knoll, A.: Kvp: A knowledge of volumes approach to robot task planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 202–208. IEEE (2013).

Mansard, N., Chaumette, F.: Task sequencing for highlevel sensor-based control. IEEE Trans. Robot. 23(1), 60– 72 (2007).

Diankov, R., Srinivasa, S.S., Ferguson, D., Kuffner, J.: Manipulation planning with caging grasps. In: 8th IEEERAS International Conference on Humanoid Robots, 2008. Humanoids 2008, pp. 285–292. IEEE (2008).

Berenson, D.: Manipulation of deformable objects without modeling and simulating deformation (2013).

Craig, J.: Introduction to Robotics: Mechanics and Control. Pearson (2005).

Pan, Z., Polden, J., Larkin, N., Duin, S.V., Norrish, J.: Recent progress on programming methods for industrial robots. In: Proceedings for the joint conference of 41st Internationel Symposium on Robotics and 6th German Conference on Robotics (2010).

Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press (2007).

Montemanni, R., Smith, D.H., Gambardella, L.M.: A heuristic manipulation technique for the sequential ordering problem. Comput. Oper. Res. 35(12), 3931–3944 (2008).

Wurll, C., Henrich, D., W¨orn, H.: Multi-goal path planning for industrial robots. In: International Conference on Robotics and Application (1999).

Srivastava, S., Kumar, S., Garg, R., Sen, P.: Generalized traveling salesman problem through n sets of nodes. In: CORSE Journal, vol. 7, pp. 97–101 (1969).

Dror, M., Efrat, A., Lubiw, A., Mitchell, J.S.B.: Touring a sequence of polygons. In: 35th Annual ACM Symposium on Theory of Computing, pp. 473–482. ACM Press (2003).

Johnson, D.S., McGeoch, L.A.: Local Search in Combinatorial Optimization. Wiley, London (1997).

Sarkar, C., Dey, S. and Agarwal, M. (2018a), "Semantic knowledge driven utility calculation towards efficient multirobot task allocation", in: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), IEEE.

Schneider, E., Sklar, E.I. and Parsons, S. (2017), "Mechanism selection for multi-robot task allocation", in: Annual Conference Towards Autonomous Robotic Systems, Springer.

Otte, M., Kuhlman, M.J. and Sofge, D. (2020), "Auctions for multi-robot task allocation in communication limited environments", Autonomous Robots, Vol. 44 Nos 3/4, pp. 547-584.

Turner, J. (2018), "Distributed task allocation optimization techniques", in: Proceedings of the 17th international conference on autonomous agents and multiagent systems.

Talebpour, Z. and Martinoli, A. (2018), "Risk-based humanaware multi-robot coordination in dynamic environments shared with humans", in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE.

Li, J. and Yang, F. (2018), "Research on multi-robot scheduling algorithms based on machine vision", EURASIP Journal on Image and Video Processing, Vol. 2018 No. 1, pp. 1-11.

Arif, M.U. and Haider, S. (2017), "An evolutionary traveling salesman approach for Multi-Robot task allocation", in: Icaart (2).

Huang, L., Ding, Y., Zhou, M., Jin, Y. and Hao, K. (2018), "Multiple-solution optimization strategy for multirobot task allocation", IEEE Transactions on Systems, Man, and Cybernetics: Systems.

Shenoy, M.V. and Anupama, K. (2017), "DTTA-Distributed, time-division multiple access based task allocation framework for swarm robots", Defence Science Journal, Vol. 67No. 3, p. 316.

Chetty, R.K., Singaperumal, M., Nagarajan, T. and Tetsunari, I. (2011), “Coordination control of wheeled mobile robots? A hybrid approach”, International Journal of Computer Applications in Technology, Vol. 41Nos 3/4, pp. 195-204.

Schillinger, P., Bürger, M. and Dimarogonas, D.V. (2018), “Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems”, The International Journal of Robotics Research, Vol. 37 No. 7, pp. 818-838.’

Lee, W. and Kim, D. (2019), “Adaptive approach to regulate task distribution in swarm robotic systems”, Swarm and Evolutionary Computation, Vol. 44, pp. 1108-1118.

Chen, X., Zhang, P., Li, F. and Du, G. (2018b), “A cluster first strategy for distributed multi-robot task allocation problem with time constraints”, in: 2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA), IEEE.

Sarkar, C., Paul, H.S. and Pal, A. (2018b), “A scalable multirobot task allocation algorithm”, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE.

Ghassemi, P. and Chowdhury, S. (2018), “Decentralized task allocation in multi-robot systems via bipartite graph matching augmented with fuzzy clustering”, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers.

Mitiche, H., Boughaci, D. and Gini, M. (2019), “Iterated local search for time-extended multi-robot task allocation with spatio-temporal and capacity constraints”, Journal of Intelligent Systems, Vol. 28No. 2, pp. 347-360.

W.P.N. dos Reis, G.L. Lopes, G.S. Bastos, An arrovian analysis on the multirobot task allocation problem: Analyzing a behavior-based architecture, Robot. Auton. Syst. 144 (2021) 103839.

M. Badreldin, A. Hussein, A. Khamis, A Comparative Study between Optimization and Market-Based Approaches to Multi-Robot Task Allocation, Adv. Artif. Intell. 2013 (2013) 256524.

H.W. Kuhn, The Hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1–2) (1955) 83–97.

M. Kasahara, K. Cai, Online Supervisory Control with Optimal Task Assignment for Efficient and Adaptive Warehouse Automation, vol. 63, 2020, pp. 90–93.

N. Atay, B. Bayazit, Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem, 2006.

M. Koes, I. Nourbakhsh, K. Sycara, Constraint Optimization Coordination Architecture for Search and Rescue Robotics, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., 2006, pp. 3977–3982.

Z. Li, X. Li, Genetic Algorithm for Task Allocation and Path Planning of Multi-robot System, J. Math. Sci. Appl. 4 (1) (2016) 34–38.

S. Saeedvand, H.S. Aghdasi, J. Baltes, Robust multi-objective multihumanoid robots task allocation based on novel hybrid metaheuristic algorithm, Appl. Intell. 49 (12) (2019) 4097–4127.

R.J. Alitappeh, K. Jeddisaravi, Multi-robot exploration in task allocation problem, Appl. Intell. 52 (2) (2022) 2189–2211.

M. Elango, S. Nachiappan, M.K. Tiwari, Balancing task allocation in multirobot systems using K-means clustering and auction based mechanisms, Expert Syst. Appl. 38 (6) (2011) 6486–6491.

Janati, F. Abdollahi, S.S. Ghidary, M. Jannatifar, J. Baltes, S. Sadeghnejad, Multi-robot Task Allocation Using Clustering Method, in: Robot Intelligence Technology and Applications 4, 2017, pp. 233–247.

Little, J.D., Murty, K.G., Sweeney, D.W., Karel, C.: An algorithm for the traveling salesman problem. Oper. Res. 11(6), 972–989 (1963).

Abdel-Malek, L.L., Li, Z.: The application of inverse kinematics in the optimum sequencing of robot tasks. Int. J. Prod. Res. 28(1), 75–90 (1990).

Edan, Y., Flash, T., Peiper, U.M., Shmulevich, I., Sarig, Y.: Near-minimum-time task planning for fruit-picking robots. IEEE Trans. Robot. Autom. 7(1), 48–56 (1991).

Zacharia, P.T., Aspragathos, N.A.: Optimal robot task scheduling based on genetic algorithms. Robot. Comput. Integr. Manuf. 21, 67–79 (2005).

Reinhart, G., Munzert, U., Vogl, W.: A programming system for robot-based remote-laser-welding with conventional optics. CIRP Ann. Manuf. Technol. 57, 37–40 (2008).

Loredo-Flores, A., Gonz´alez-Galv´an, E.J., Cervantes- S´anchez, J.J., Martinez-Soto, A.: Optimization of industrial, vision-based, intuitively generated robot pointallocating tasks using genetic algorithms. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 38(4), 600–608 (2008).

Spitz, S.N., Requicha, A.A.G.: Multiple-goals path planning for coordinate measuring machines. In: IEEE International Conference on Robotics and Automation, pp. 2322– 2327 (2000).

Saha, M., S´anchez-Ante, G., Latombe, J.C.: Planning multi-goal tours for robot arms. In: International Conference on Robotics and Automation (ICRA), vol. 3, pp. 3797–3803. IEEE (2003).

Bu, W., Liu, Z., Tan, J.: Industrial robot layout based on operation sequence optimisation. Int. J. Prod. Res. 41, 4125–4145 (2009).

Xidias, E.K., Zacharia, P.T., Aspragathos, N.A.: Timeoptimal task scheduling for articulated manipulators in environments cluttered with obstacles. In: Robotica, vol. 28, pp. 427–440. Cambridge University Press (2010).

Zacharia, P., Xidias, E., Aspragathos, N.: Task scheduling and motion planning for an industrial manipulator. Robot. Comput. Integr. Manuf. 29, 449–462 (2013).

Huang, Y., Gueta, L.B., Chiba, R., Arai, T., Ueyama, T., Ota, J.: Selection of manipulator system for multiple-goal task by evaluating task completion time and cost with computational time constraints. Adv. Robot. 27(4), 233–245 (2013).

Lattanzi, L., Cristalli, C.: An efficient motion planning algorithm for robot multi-goal tasks. In: IEEE International Symposium on Industrial Electronics (ISIE) (2013).

Gentilini, I., Nagamatsu, K., Shimada, K.: Cycle time based multi-goal path optimization for redundant robotic systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2013).

Jang, I., Shin, H.-S. and Tsourdos, A. (2018), "Anonymous hedonic game for task allocation in a large-scale multiple agent system", IEEE Transactions on Robotics, Vol. 34 No. 6, pp. 1534-1548.

Aapaoja, A., & Haapasalo, H. (2014, June). The challenges of standardization of products and processes in construction. In Proceedings of the 22nd Annual Conference of the International Group for Lean (pp. 983-993).

Abushwereb, M., Liu, H., & Al-Hussein, M. (2019). A knowledge-based approach towards automated manufacturing-centric BIM: Wood frame design and modelling for light-frame buildings. In Modular Offsite Construction Summit Proceedings (pp. 100-107). Banff, AB.

Adey, B. T., García de Soto, B., & Chen, Q. (2018). Construction automation: Research areas, industry concerns and suggestions for advancement. Automation in Construction, 94, 22-38.

Altaf, M. S., Bouferguene, A., Liu, H., Al-Hussein, M., & Yu, H. (2018). Integrated production planning and control system for a panelized home prefabrication facility using simulation and RFID. Automation in Construction, 85, 369-383.

Apolinarska, A. A., Pacher, M., Li, H., Cote, N., Pastrana, R., Gramazio, F., & Kohler, M. (2021). Robotic assembly of timber joints using reinforcement learning. Automation in Construction, 125, 103569.

Associated General Contractors. (2019). Eighty percent of contractors report difficulty finding qualified craft workers to hire as firms give low marks to quality of new worker pipeline. Retrieved May 6, 2023, from https://www.agc.org/news/2019/08/27/eighty-percent-contractors-report-difficulty-finding-qualified-craft-workers-hire-0

Becerik-Gerber, B., & Kensek, K. (2010). Building information modeling in architecture, engineering, and construction: Emerging research directions and trends. Journal of Professional Issues in Engineering Education and Practice, 136, 139-147.

Bennulf, M., Svensson, B., & Danielsson, F. (2018). Verification and deployment of automatically generated robot programs used in prefabrication of house walls. Procedia CIRP, 72, 272-276.

Bock, T. (2015). The future of construction automation: Technological disruption and the upcoming ubiquity of robotics. Automation in Construction, 59, 113-121.

Bock, T., & Linner, T. (2016). Robotic industrialization: Automation and robotic technologies for customized component, module, and building prefabrication. Cambridge University Press.

Bousquin, J. (2020). Report: Nearly half of America's deadliest jobs are in construction. Construction Dive. Retrieved from https://www.constructiondive.com/news/report-nearly-half-of-americas-deadliest-jobs-are-in-construction/586801/

Brissi, S. G., & Debs, L. (2019, July). Lean, automation and modularization in construction. In 27th Annual Conference of the International Group for Lean Construction, IGLC (Vol. 2019, pp. 711-722).

Buchli, J., Giftthaler, M., Kumar, N., Lussi, M., Sandy, T., Dörfler, K., & Hack, N. (2018). Digital in situ fabrication: Challenges and opportunities for robotic in situ fabrication in architecture, construction, and beyond. Cement and Concrete Research, 112, 66-75.

BuildingSMART. (2023). Certified software, software certification. Retrieved August 11, 2023, from https://www.buildingsmart.org/compliance/software-certification/certified-software/

Castro-Lacouture, D. (2009). Construction automation. In Springer Handbook of Automation (pp. 1063-1078). Springer. ISBN: 978-3-540-78831-7.

Chea, C. P., Bai, Y., Pan, X., Arashpour, M., & Xie, Y. (2020). An integrated review of automation and robotic technologies for structural prefabrication and construction. Transportation Safety and Environment, 2(2), 81-96.

Chen, Q., García de Soto, B., & Adey, B. T. (2018). Construction automation: Research areas, industry concerns and suggestions for advancement. Automation in Construction, 94, 22-38.

Chui, M., & Mischke, J. (2019). The impact and opportunities of automation in construction. Voices on Infrastructure. Retrieved from https://www.mckinsey.com/~/media/McKinsey/Industries

Craveiro, F., Duarte, J. P., Bartolo, H., & Bartolo, P. J. (2019). Additive manufacturing as an enabling technology for digital construction: A perspective on Construction 4.0. Automation in Construction, 103, 251-267.

Davila Delgado, J. M., Oyedele, L., Ajayi, A., Akanbi, L., Akinade, O., Bilal, M., & Owolabi, H. (2019). Robotics and automated systems in construction: Understanding industry-specific challenges for adoption. Journal of Building Engineering, 26, 100868.

Ding, L., Jiang, W., Zhou, Y., Zhou, C., & Liu, S. (2020). BIM-based task-level planning for robotic brick assembly through image-based 3D modeling. Advanced Engineering Informatics, 43, 100993.

Eastman, C., Teicholz, P., Sacks, R., & Liston, K. (2011). BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors (2nd ed.). Wiley.

Fernández-Rodríguez, S., Cortés-Pérez, J. P., Muriel, P. P., Tormo-Molina, R., & Maya-Manzano, J. M. (2018). Environmental impact assessment of Pinaceae airborne pollen and green infrastructure using BIM. Automation in Construction, 96, 494-507.

Gallaher, M. P., O'Connor, A. C., Dettbarn, J. L., & Gilday, L. T. (2004). Cost analysis of inadequate interoperability in the U.S. capital facilities industry. NIST. http://www.bentleyuser.dk/sites/default/files/nist_report.pdf

Gibb, A. (2001). Standardization and pre-assembly: Distinguishing myth from reality using case study research. Construction Management and Economics, 19, 307-315.

Goodman, J. (2019). More than 7K robots to take on construction work by 2025. Construction Dive. Retrieved from https://www.constructiondive.com/news/more-than-7k-robots-to-take-on-construction-work-by-2025/554653/

Jeffers, M. (2016). Autonomous robotic assembly with variable material properties. In D. Reinhardt, R. Saunders, & J. Burry (Eds.), Robotic Fabrication in Architecture, Art and Design (pp. 49-62). Springer.

Kamath, A., & Sharma, R. K. (2019). Robotics in construction: Opportunities and challenges. International Journal of Recent Technology and Engineering, 8(2S11), 2227-2230.

King, N., Bechthold, M., Kane, A., & Michalatos, P. (2014). Robotic tile placement: Tools, techniques and feasibility. Automation in Construction, 39, 161-166.

Kontovourkis, O., Tryfonos, G., & Georgiou, C. (2020). Robotic additive manufacturing (RAM) with clay using topology optimization principles for toolpath planning: The example of a building element. Architectural Science Review, 63, 105-118.

Kyjanek, O., Al Bahar, B., Vasey, L., Wannemacher, B., & Menges, A. (2019, May). Implementation of an augmented reality (AR) workflow for human robot collaboration in timber prefabrication. In Proceedings of the 36th International Symposium on Automation and Robotics in Construction, ISARC (pp. 1223-1230). International Association for Automation and Robotics in Construction (IAARC). Banff, Canada.

Lawson, M., Ogden, R., & Goodier, C. I. (2014). Design in modular construction (Vol. 476, p. 280). Boca Raton, FL: CRC Press.

Leviäkangas, P., Paik, S. M., & Moon, S. (2017). Keeping up with the pace of digitization: The case of the Australian construction industry. Technology in Society, 50, 33-43.

Linner, T., & Bock, T. (2012). Evolution of large-scale industrialisation and service innovation in Japanese prefabrication industry. Construction Innovation, 12(2), 156-178.

Lopez, D., & Froese, T. M. (2016). Analysis of costs and benefits of panelized and modular prefabricated homes. Procedia Engineering, 145, 1291-1297.

Lu, N., & Korman, T. (2010). Implementation of building information modeling (BIM) in modular construction: Benefits and challenges. In Construction Research Congress 2010 (pp. 1136-1145).

Martinez, P., Livojevic, M., Jajal, P., Aldrich, D. R., Al-Hussein, M., & Ahmad, R. (2020). Simulation-driven design of wood framing support systems for off-site construction machinery. Journal of Construction Engineering and Management, 146(7), 04020075.

McGraw-Hill Construction. (2011). Prefabrication and modularization. McGraw-Hill Construction.

McGraw-Hill Construction. (2016). The business value of BIM for construction in major global markets: How contractors around the world are driving innovation with building information modeling. Bedford, MA. Retrieved from http://static.autodesk.net/dc/content/dam/autodesk/www/solutions/building-information-modeling/construction/business-value-of-bim-for-construction-in-global-markets.pdf

Menges, A., Schwinn, T., Krieg, O. D., & Schwinn, T. (2015). Landesgartenschau Exhibition Hall. Interlocking Digital and Material Cultures, 55-71.

Meyer, T., & Jacob, F. (2008). Network design: Optimizing the global production footprint. In Global Production: A Handbook for Strategy and Implementation (pp. 140-190).

National Academy of Engineering. (2008). NAE Grand Challenges for Engineering.

Nunnally, S. W. (2007). Construction methods and management (7th ed.). Pearson Education. ISBN: 0135000793.

Orlowski, K. (2020). Automated manufacturing for timber-based panelised wall systems. Automation in Construction, 109, 102988.

Pan, M., Linner, T., Pan, W., Cheng, H., & Bock, T. (2018). A framework of indicators for assessing construction automation and robotics in the sustainability context. Journal of Cleaner Production, 182, 82-95.

Ramaji, I. J., & Memari, A. M. (2016). Product architecture model for multistory modular buildings. Journal of Construction Engineering and Management, 142, 1-14. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001159

Razkenari, M. A., Fenner, A. E., Woo, J., Hakim, H., & Kibert, C. J. (2018). A systematic review of applied information systems in industrialized construction. In Construction Research Congress 2018 (pp. 101-110).

Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors (3rd ed.). Wiley.

Salvador, F., Rungtusanatham, M., Forza, C., & Trentin, A. (2007). Mix flexibility and volume flexibility in a build-to-order environment: Synergies and trade-offs. International Journal of Operations & Production Management, 27(11), 1173-1191.

Santana-Sosa, A., & Riola-Parada, F. (2018). A theoretical approach towards resource efficiency in multi-story timber buildings through BIM and lean. In Proceedings of the World Conference on Timber Engineering (WCTE).

Schindler, C. (2010). Die Standards des Nonstandards. In GAM Architecture Magazine 06 (pp. 180-193). Springer, Vienna.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., ... & Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. International Journal of Robotics Research, 33(9), 1251-1270.

Steel, J., Drogemuller, R., & Toth, B. (2012). Model interoperability in building information modelling. Software and Systems Modeling, 11(1), 99-109.

Sterling, L., & Shapiro, E. (1994). The Art of Prolog: Advanced Programming Techniques (2nd ed.). MIT Press. ISBN: 9780262193382.

Tang, F., Ma, T., Zhang, J., Guan, Y., & Chen, L. (2020). Integrating three-dimensional road design and pavement structure analysis based on BIM. Automation in Construction, 113, 103152.

Tao, B., Zhao, X., Yan, S., & Ding, H. (2020). Kinematic modeling and control of mobile robot for large-scale workpiece machining. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 236, 29-38.

Terada, Y., & Murata, S. (2008). Automatic modular assembly system and its distributed control. The International Journal of Robotics Research, 27(3-4), 445-462.

U.S. Bureau of Labor Statistics. (2020). Job openings and labor turnover. Economic news release. Retrieved August 8, 2020, from https://www.bls.gov/news.release/jolts.a.htm

Van Ooteghem, K., & Xu, L. (2012). The life-cycle assessment of a single-storey retail building in Canada. Building and Environment, 49, 212-226.

Vähä, P., Heikkilä, T., Kilpeläinen, P., Järviluoma, M., & Gambao, E. (2013). Extending automation of building construction—Survey on potential sensor technologies and robotic applications. Automation in Construction, 36, 168-178.

Wang, M., Wang, C. C., Sepasgozar, S., & Zlatanova, S. (2020). A systematic review of digital technology adoption in off-site construction: Current status and future direction towards industry 4.0. Buildings, 10(11), 204.

Willmann, J., Knauss, M., Bonwetsch, T., Apolinarska, A. A., Gramazio, F., & Kohler, M. (2016). Robotic timber construction - Expanding additive fabrication to new dimensions. Automation in Construction, 61, 16-23.

Wong Chong, O., Baker, C., Afsari, K., & Zhang, J. (2020). Integration of BIM processes in architectural design, structural analysis, and detailing: Current status and limitations. In M. El Asmar, D. Grau, & P. Tang (Eds.), Construction Research Congress 2020 (pp. 127). ASCE.

Zhang, J., & El-Gohary, N. M. (2017). Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. Automation in Construction, 73, 45-57.

Zhang, J., Laddipeerla, S., & El-Gohary, N. M. (2018). A feasibility study of IFC-based BIM 4D simulation using commercial systems to support construction planning in the U.S. In T. Sulbaran (Ed.), 54th ASC Annual International Conference Proceedings (pp. 1-10).

Associated Schools of Construction. Retrieved May 3, 2022, from http://ascpro0.ascweb.org/archives/cd/2018/paper/CPRT119002018.pdf

Zhang, J., & El-Gohary, N. M. (2016). Semantic-based logic representation and reasoning for automated regulatory compliance checking. Journal of Computing in Civil Engineering, 31, 1-12.

Ataka, A., Lam, H., & Althoefer, K. (2018). Reactive magnetic-field-inspired navigation for non-holonomic mobile robots in unknown environments. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 6983-6988). IEEE.

Alam, M. M., Ibaraki, S., Fukuda, K., Morita, S., & Usuki, H. (2020, July). Identification of a kinematic model of a 6DOF industrial manipulator with angular positioning deviation "Error Map" of rotary axes. In International Symposium on Flexible Automation (Vol. 83617, p. V001T05A003). American Society of Mechanical Engineers.

Balaguer, C., Abderrahim, M., Navarro, J. M., Boudjabeur, S., Aromaa, P., Kahkonen, K., & Atkin, B. (2002). FutureHome: An integrated construction automation approach. IEEE Robotics and Automation Magazine, 9(1), 55-66.

Becker, M., Lilge, T., Müller, M. A., & Haddadin, S. (2021). Circular fields and predictive multi-agents for online global trajectory planning. IEEE Robotics and Automation Letters, 6, 2618-2625.

Bock, T. (2007). Construction robotics. Autonomous Robots, 22(3), 201-209.

Bock, T. (2015). The future of construction automation: Technological disruption and the upcoming ubiquity of robotics. Automation in Construction, 59, 113-121.

Bock, T., & Linner, T. (Eds.). (2015). Automation and robotics in building component manufacturing. In Robotic industrialization: Automation and robotic technologies for customized component, module, and building prefabrication (pp. 7-65). Cambridge University Press.

Borenstein, J., & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. IEEE Transactions on Systems, Man, and Cybernetics, 19(5), 1179-1187.

Carlson, D. O. (2015). Automated Builder: Dictionary Encyclopedia of Industrialized Housing. Automated Builder. ISBN: 0960740821.

Ding, L., Jiang, W., Zhou, Y., Zhou, C., & Liu, S. (2020). BIM-based task-level planning for robotic brick assembly through image-based 3D modeling. Advanced Engineering Informatics, 43, 100993.

Dörfler, K., Hack, N., Sandy, T., Giftthaler, M., Lussi, M., Walzer, A. N., ... & Kohler, M. (2019). Mobile robotic fabrication beyond factory conditions: Case study Mesh Mould wall of the DFAB HOUSE. Construction Robotics, 3, 53-67.

Fan, X., Guo, Y., Liu, H., Wei, B., & Lyu, W. (2020). Improved artificial potential field method applied for AUV path planning. Mathematical Problems in Engineering, 2020, 6523158.

Gambao, E., Balaguer, C., & Gebhart, F. (2000). Robot assembly system for computer-integrated construction. Automation in Construction, 9(5), 479-487.

Golabchi, A., Han, S., Seo, J., Han, S., Lee, S., & Al-Hussein, M. (2015). An automated biomechanical simulation approach to ergonomic job analysis for workplace design. Journal of Construction Engineering and Management, 141(8), 1-12.

Henrich, D., & Qin, C. (1996). Path planning for industrial robot arms-A parallel randomized approach.

Huang, Y., Garrett, C. R., & Mueller, C. T. (2018). Automated sequence and motion planning for robotic spatial extrusion of 3D trusses. Construction Robotics, 2(1-4), 15-39.

Ichter, B., Harrison, J., & Pavone, M. (2018). Learning sampling distributions for robot motion planning. In Proceedings of the IEEE International Conference on Robotics and Automation (pp. 7087-7094).

Iturralde, K., Kinoshita, T., & Bock, T. (2019). Grasped element position recognition and robot pose adjustment during assembly. In M. Al-Hussein (Ed.), Proceedings of the 36th International Symposium on Automation and Robotics in Construction (ISARC) (pp. 461-468). International Association for Automation and Robotics in Construction (IAARC).

Kamali, M., & Hewage, K. (2016). Life cycle performance of modular buildings: A critical review. Renewable and Sustainable Energy Reviews, 62, 1171-1183.

Kasperzyk, C., Kim, M. K., & Brilakis, I. (2017). Automated re-prefabrication system for buildings using robotics. Automation in Construction, 83, 184-195.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011, May). STOMP: Stochastic trajectory optimization for motion planning. In 2011 IEEE international conference on robotics and automation (pp. 4569-4574). IEEE.Kavraki,

KUKA. (2023). KUKA KR QUANTEC PA 3200/120. Retrieved August 21, 2023, from https://www.kuka.com/en-cn/products/robotics-systems/industrial-robots/kr-quantec.

Latombe, J. C. (1991). Robot motion planning. Springer US.

Martinez, P., Ahmad, R., & Al-Hussein, M. (2019). A vision-based system for pre-inspection of steel frame manufacturing. Automation in Construction, 97, 151-163.

MoveIt, Retrieved March 08, 2024, from, https://moveit.ros.org.

Mullens, M. A. (2011). Factory design for modular homebuilding: Equipping the modular factory for success. Constructability Press. ISBN: 9780470378366.

Nahangi, M., Yeung, J., Haas, C. T., & Walbridge, S. (2015). Automated assembly discrepancy feedback using 3D imaging and forward kinematics. Automation in Construction, 56, 36-46.

Petrović, L., Peršić, J., Seder, M., & Marković, I. (2019, September). Stochastic optimization for trajectory planning with heteroscedastic Gaussian processes. In 2019 European Conference on Mobile Robots (ECMR) (pp. 1-6). IEEE.

Randek, ZeroLabor Robitc System, Retrieved March 08, 2024, from, www.randek.com/en/, 2021.

Ratliff, N., Zucker, M., Bagnell, J. A., & Srinivasa, S. (2009). CHOMP: Gradient optimization techniques for efficient motion planning. In 2009 IEEE International Conference on Robotics and Automation (pp. 489-494).

Reif, J. H. (1979). Complexity of the mover's problem and generalizations. In 20th Annual Symposium on Foundations of Computer Science (sfcs 1979) (pp. 421-427).

Renuka, K., & Bhuvanesh, N. (2020). Kinematic and dynamic modeling and PID control of three degree-of-freedom robotic.

Salama, T., Salah, A., Moselhi, O., & Al-Hussein, M. (2017). Near optimum selection of module configuration for efficient modular construction. Automation in Construction, 83, 316-329.

Saxena, D. M., Bae, S., Nakhaei, A., Fujimura, K., & Likhachev, M. (2020). Driving in dense traffic with model-free reinforcement learning. In IEEE International Conference on Robotics and Automation (ICRA) 2020 (pp. 5385-5392).

Schoenborn, J. M., Jones, J. R., Schubert, R. P., & Hardiman, T. E. (2012). A case study approach to identifying the constraints and barriers to design innovation for modular construction.

Smith, R. E. (2010). Prefab architecture: A guide to modular design and construction. John Wiley and Sons, Inc. ISBN: 9780470378366.

Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N., & Feyzeau, P. (2013). Path planning: A 2013 survey. In Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM) (pp. 1-8).

Vagale, A., Oucheikh, R., Bye, R. T., & Osen, O. L. (2021). Path planning and collision avoidance for autonomous surface vehicles I: A review. Journal of Marine Science and Technology, 26, 1292-1306.

Wang, P., Gao, S., Li, L., Sun, B., & Cheng, S. (2019). Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. Energies, 12(12), 2342.

Xiao, W., Cassandras, C. G., & Belta, C. A. (2021). Bridging the gap between optimal trajectory planning and safety-critical control with applications to autonomous vehicles. Automatica, 129, 109592.

Yang, A., Liu, Q., Naeem, W., & Fei, M. (2021). Robot dynamic collision detection method based on obstacle point cloud envelope model. In Q. Han, S. McLoone, C. Peng, & B. Zhang (Eds.), Intelligent equipment, robots, and vehicles - 7th International Conference on Life System Modeling and Simulation (pp. 370-378). Springer Singapore.

Zhang, K., Zhu, Y., Lou, C., Zheng, P., & Kovač, M. (2019). A design and fabrication approach for pneumatic soft robotic arms using 3D printed origami skeletons. In 2019 2nd IEEE International Conference on Soft Robotics (RoboSoft) (pp. 821-827).

Gbadamosi, A. Q., Mahamadu, A. M., Oyedele, L. O., Akinade, O. O., Manu, P., Mahdjoubi, L., & Aigbavboa, C. (2019). Offsite construction: Developing a BIM-Based optimizer for assembly. Journal of cleaner production, 215, 1180-1190.

Isaac, S., Bock, T., & Stoliar, Y. (2016). A methodology for the optimal modularization of building design. Automation in construction, 65, 116-124.

Jensen, P., Olofsson, T., & Johnsson, H. (2012). Configuration through the parameterization of building components. Automation in construction, 23, 1-8.

Mekawy, M., & Petzold, F. (2017). 'Exhaustive Exploration of Modular Design Options to Inform Decision Making'. Sharing of Computable Knowledge: Proceedings of eCAADe 2017, 107-114.

Root, S. P., Cribbs, J., & Chasey, A. D. (2019). Case Study: Off-site manufacturing of EIFS Panelized Wall Assemblies to Gain Efficiency in Construction Sequencing. Modular and Offsite Construction (MOC) Summit Proceedings, 357-364.

Isikdag, U., Aouad, G., Underwood, J., & Wu, S. (2007). Building information models: a review on storage and exchange mechanisms. Bringing ITC knowledge to work, 135-144.

Arayici, Y., Fernando, T., Munoz, V., & Bassanino, M. (2018). Interoperability specification development for integrated BIM use in performance based design. Automation in Construction, 85, 167-181.

Ren, R., Zhang, J., & Dib, H. N. (2018). BIM interoperability for structure analysis. In Construction Research Congress 2018 (pp. 470-479).

Wu, J., & Zhang, J. (2019). New automated BIM object classification method to support BIM interoperability. Journal of Computing in Civil Engineering, 33(5), 04019033.

Fürnkranz, J., & Kliegr, T. (2015, July). A brief overview of rule learning. In International symposium on rules and rule markup languages for the semantic web (pp. 54-69). Cham: Springer International Publishing.

Johansson, I. (2011). Order, direction, logical priority and ontological categories. Ontological Categories, Ontos Verlag, 89-107.

Chong, O. W., & Zhang, J. (2021). Logic representation and reasoning for automated BIM analysis to support automation in offsite construction. Automation in Construction, 129, 103756.

Uszkoreit, H. (1986). Constraints on order.

Väänänen, J. (2019). Second-order and Higher-order Logic.

Makowsky, J. A. (1987). Why Horn formulas matter in computer science: Initial structures and generic examples. Journal of Computer and System Sciences, 34(2-3), 266-292.

Pereira, F. C., & Shieber, S. M. (2002). Prolog and natural-language analysis. Microtome Publishing.

Zhou, N. F. (1997). B-Prolog user's manual (version 2.1). Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Fukuoka, Japan.

Luger, G. F., & Stubblefield, W. A. (2009). AI algorithms, data structures, and idioms in Prolog, Lisp, and Java. Pearson Addison-Wesley.

Ritchie, G. (2002). Prolog Step-by-Step.

Covington, M. A., Nute, D., & Vellino, A. (1995). PROLOG programming in depth. USA: Artificial Intelligence Programs The University of Georgia.

Portoraro, F. (2001). Automated reasoning.

Nehra, E. (2015). Artificial Intelligence in Modern Times. In International Conference on Recent Innovations in Science, Engineering and Management.

Darko, A., Chan, A. P., Adabre, M. A., Edwards, D. J., Hosseini, M. R., & Ameyaw, E. E. (2020). Artificial intelligence in the AEC industry: Scientometric analysis and visualization of research activities. Automation in construction, 112, 103081.

OWL Working Group, Web Ontology Language (OWL), OWL, 2012. https://www.w3.org/OWL/ (accessed January 5, 2024).

El-Gohary, N. M., & El-Diraby, T. E. (2010). Domain ontology for processes in infrastructure and construction. Journal of construction engineering and management, 136(7), 730-744.

Dahn, B. I. (1998). Robbins algebras are Boolean: A revision of McCune's computer-generated solution of Robbins problem. Journal of Algebra, 208(2), 526-532.

Kotelnikov, E. (2018). Automated Theorem Proving with Extensions of First-Order Logic. Chalmers Tekniska Hogskola (Sweden).

Pauwels, P., & Terkaj, W. (2016). EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in construction, 63, 100-133.

Gonzalez, E., Piñeiro, J. D., Toledo, J., Arnay, R., & Acosta, L. (2021). An approach based on the ifcOWL ontology to support indoor navigation. Egyptian Informatics Journal, 22(1), 1-13.

Pauwels, P., Krijnen, T., Terkaj, W., & Beetz, J. (2017). Enhancing the ifcOWL ontology with an alternative representation for geometric data. Automation in Construction, 80, 77-94.

Allen, E., Thallon, R., & Schreyer, A. C. (2017). Fundamentals of residential construction. John Wiley & Sons.

Staranowicz, A., & Mariottini, G. L. (2011, May). A survey and comparison of commercial and open-source robotic simulator software. In Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments (pp. 1-8).

Meschini, S., Iturralde, K., Linner, T., & Bock, T. (2016). Novel applications offered by integration of robotic tools in BIM-based design workflow for automation in construction processes. In Proceedings of the CIB* IAARC W119 CIC 2016 Workshop.

Kim, S., Peavy, M., Huang, P. C., & Kim, K. (2021). Development of BIM-integrated construction robot task planning and simulation system. Automation in Construction, 127, 103720.

Wagner, H. J., Alvarez, M., Groenewolt, A., & Menges, A. (2020). Towards digital automation flexibility in large-scale timber construction: integrative robotic prefabrication and co-design of the BUGA Wood Pavilion. Construction Robotics, 4(3), 187-204.

Yang, C. H., & Kang, S. C. (2021). Collision avoidance method for robotic modular home prefabrication. Automation in Construction, 130, 103853.

Sarkar, M., Pradhan, P., & Ghose, D. (2019). Planning robot motion using deep visual prediction. arXiv preprint arXiv:1906.10182.

Bertram, N., Fuchs, S., Mischke, J., Palter, R., Strube, G., & Woetzel, J. (2019). Modular construction: From projects to products. McKinsey & Company: Capital Projects & Infrastructure, 1, 1-34.

Appendix-A

A-1 RAP Result for the Testing Model 1

Component/ Element	**Item**	**No. of Relevant RBF**	**No. of Extracted RBF**	**No. of Correctly Extracted RBF**	***Sn.*** **%**	**Sp. %**	**Pr. %**	**Acc. %**	**F1m. %**
Exterior Wall	Quantity (Integer)	12	30	12	100	100	40	100	57
Exterior Wall	Dimensions (L,W,H)	36	90	36	100	100	40	10	57
Exterior Wall	Material (String)	60	78	60	100	100	77	100	87
Exterior Wall	Rough Opening Quantity (Integer)	14	14	14	100	100	100	100	100
Exterior Wall	Rough Opening Dimensions (L,W,H)	42	42	42	100	100	100	100	100
Interior Wall	Quantity (Integer)	30	30	30	100	100	100	100	100
Interior Wall	Dimensions (L,W,H)	90	81	81	90	90	100	90	95
Interior Wall	Material (String)	90	90	90	100	100	100	100	100
Interior Wall	Rough Opening Quantity (Integer)	17	17	17	100	100	100	100	100

Interior Wall	Rough Opening Dimensions (L,W,H)	51	45	45	88	88	100	88	94
Floor	Quantity (Integer)	3	3	3	100	100	100	100	100
Floor	Dimensions (L,W,H)	9	6	6	67	67	100	67	80
Floor	Material (String)	9	12	9	100	100	75	100	86
Floor	Rough Opening Quantity (Integer)	1	1	1	100	100	100	100	100
Floor	Rough Opening Dimensions (L,W,H)	1	1	1	100	100	100	100	100
Roof	Quantity (Integer)	11	11	11	100	100	100	100	100
Roof	Dimensions (L,W,H)	33	0	0	0	0	0	0	0
Roof	Material (String)	33	33	33	100	100	100	100	100
Roof	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Roof	Rough Opening	-	-	-	-	-	-	-	-

	Dimensions (L,W,H)								
Column	Quantity (Integer)	-	-	-	-	-	-	-	-
Column	Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Column	Material (String)	-	-	-	-	-	-	-	-
Beam	Quantity (Integer)	-	-	-	-	-	-	-	-
Beam	Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Beam	Material (String)	-	-	-	-	-	-	-	-
Total Instances		542	584	491	91	91	84	91	87

A-2 RAP Result for the Testing Model 2

Component/ Element	**Item**	**No. of Relevant RBF**	**No. of Extracted RBF**	**No. of Correctly Extracted RBF**	***Sn.* %**	**Sp. %**	**Pr. %**	**Acc. %**	**F1m. %**
Exterior Wall	Quantity (Integer)	138	138	138	100	100	100	100	100
Exterior Wall	Dimensions (L,W,H)	414	198	198	48	48	100	48	65
Exterior Wall	Material (String)	552	558	552	100	100	99	100	99
Exterior Wall	Rough Opening	180	180	180	100	100	100	100	100

	Quantity (Integer)								
Exterior Wall	Rough Opening Dimensions (L,W,H)	540	528	528	98	98	100	98	99
Interior Wall	Quantity (Integer)	126	124	124	98	98	100	98	99
Interior Wall	Dimensions (L,W,H)	378	366	366	97	97	100	97	98
Interior Wall	Material (String)	378	372	372	98	98	100	98	99
Interior Wall	Rough Opening Quantity (Integer)	60	60	60	100	100	100	100	100
Interior Wall	Rough Opening Dimensions (L,W,H)	180	180	180	100	100	100	100	100
Floor	Quantity (Integer)	8	8	8	100	100	100	100	100
Floor	Dimensions (L,W,H)	24	24	24	100	100	100	100	100
Floor	Material (String)	24	24	24	100	100	100	100	100
Floor	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-

Floor	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Roof	Quantity (Integer)	32	18	18	56	56	100	56	72
Roof	Dimensions (L,W,H)	96	54	54	56	56	100	56	72
Roof	Material (String)	68	54	54	79	79	100	79	89
Roof	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Roof	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Column	Quantity (Integer)	16	16	16	100	100	100	100	100
Column	Dimensions (L,W,H)	48	48	48	100	100	100	100	100
Column	Material (String)	16	16	16	100	100	100	100	100
Beam	Quantity (Integer)	18	18	18	100	100	100	100	100
Beam	Dimensions (L,W,H)	54	54	54	100	100	100	100	100
Beam	Material (String)	18	18	18	100	100	100	100	100
Total Instances		3368	3056	3050	91	91	100	91	95

A-3 RAP Result for the Testing Model 3

Component/ Element	**Item**	**No. of Relevant RBF**	**No. of Extracted RBF**	**No. of Correctly Extracted RBF**	***Sn.*** **%**	**Sp. %**	**Pr. %**	**Acc. %**	**F1m. %**
Exterior Wall	Quantity (Integer)	48	48	48	100	100	100	100	100
Exterior Wall	Dimensions (L,W,H)	145	145	145	100	100	100	100	100
Exterior Wall	Material (String)	213	213	213	100	100	100	100	100
Exterior Wall	Rough Opening Quantity (Integer)	29	29	29	100	100	100	100	100
Exterior Wall	Rough Opening Dimensions (L,W,H)	87	87	87	100	100	100	100	100
Interior Wall	Quantity (Integer)	80	74	80	93	93	93	93	93
Interior Wall	Dimensions (L,W,H)	172	168	168	98	98	100	98	99
Interior Wall	Material (String)	172	172	172	100	100	100	100	100

Interior Wall	Rough Opening Quantity (Integer)	16	16	16	100	100	100	100	100
Interior Wall	Rough Opening Dimensions (L,W,H)	49	49	49	100	100	100	100	100
Floor	Quantity (Integer)	6	6	6	100	100	100	100	100
Floor	Dimensions (L,W,H)	18	18	18	100	100	100	100	100
Floor	Material (String)	20	20	20	100	100	100	100	100
Floor	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Floor	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Roof	Quantity (Integer)	2	2	2	100	100	100	100	100
Roof	Dimensions (L,W,H)	6	6	6	100	100	100	100	100

Roof	Material (String)	2	2	2	100	100	100	100	100
Roof	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Roof	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Column	Quantity (Integer)	13	13	13	100	100	100	100	100
Column	Dimensions (L,W,H)	28	28	28	100	100	100	100	100
Column	Material (String)	8	8	8	100	100	100	100	100
Beam	Quantity (Integer)	-	-	-	-	-	-	-	-
Beam	Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Beam	Material (String)	-	-	-	-	-	-	-	-
Total Instances		1114	1104	1110	99.6	99.6	100	99.6	100

A-4 RAP Result for the Testing Model 4

Component/Element	**Item**	**No. of Relevant RBF**	**No. of Extracted RBF**	**No. of Correctly Extracted RBF**	***Sn.* %**	**Sp. %**	**Pr. %**	**Acc. %**	**F1m. %**
Exterior Wall	Quantity (Integer)	4	4	4	100	100	100	100	100
Exterior Wall	Dimensions (L,W,H)	12	18	12	100	100	67	100	80
Exterior Wall	Material (String)	12	21	12	100	100	57	100	73
Exterior Wall	Rough Opening Quantity (Integer)	13	76	13	100	100	17	100	29
Exterior Wall	Rough Opening Dimensions (L,W,H)	39	286	39	100	100	14	100	24
Interior Wall	Quantity (Integer)	4	4	4	100	100	100	100	100
Interior Wall	Dimensions (L,W,H)	12	12	12	100	100	100	100	100
Interior Wall	Material (String)	20	20	20	100	100	100	100	100
Interior Wall	Rough Opening	14	14	14	100	100	100	100	100

	Quantity (Integer)								
Interior Wall	Rough Opening Dimensions (L,W,H)	41	41	41	100	100	100	100	100
Floor	Quantity (Integer)	3	3	3	100	100	100	100	100
Floor	Dimensions (L,W,H)	9	9	9	100	100	100	100	100
Floor	Material (String)	3	3	3	100	100	100	100	100
Floor	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Floor	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Roof	Quantity (Integer)	2	2	2	100	100	100	100	100
Roof	Dimensions (L,W,H)	6	6	6	100	100	100	100	100
Roof	Material (String)	4	4	4	100	100	100	100	100

Roof	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Roof	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Column	Quantity (Integer)	67	67	67	100	100	100	100	100
Column	Dimensions (L,W,H)	202	201	202	100	100	100	100	100
Column	Material (String)	67	67	67	100	100	100	100	100
Beam	Quantity (Integer)	61	76	61	100	98	80	100	89
Beam	Dimensions (L,W,H)	183	170	126	69	78	74	69	71
Beam	Material (String)	61	76	61	100	100	80	100	89
Total Instances		839	1180	782	93.2	93.2	66.3	93.2	77.5

A-5 RAP Result for the Testing Model 5

Component/Element	**Item**	**No. of Relevant RBF**	**No. of Extracted RBF**	**No. of Correctly Extracted RBF**	***Sn.* %**	**Sp. %**	**Pr. %**	**Acc. %**	**F1m. %**
Exterior Wall	Quantity (Integer)	138	138	138	100	100	100	100	100
Exterior Wall	Dimensions (L,W,H)	414	198	198	48	48	100	48	65
Exterior Wall	Material (String)	552	558	552	100	100	99	100	99
Exterior Wall	Rough Opening Quantity (Integer)	180	180	180	100	100	100	100	100
Exterior Wall	Rough Opening Dimensions (L,W,H)	540	528	528	98	98	100	98	99
Interior Wall	Quantity (Integer)	126	124	124	98	98	100	98	99
Interior Wall	Dimensions (L,W,H)	378	366	366	97	97	100	97	98
Interior Wall	Material (String)	378	372	372	98	98	100	98	99
Interior Wall	Rough Opening	60	60	60	100	100	100	100	100

	Quantity (Integer)								
Interior Wall	Rough Opening Dimensions (L,W,H)	180	180	180	100	100	100	100	100
Floor	Quantity (Integer)	8	8	8	100	100	100	100	100
Floor	Dimensions (L,W,H)	24	24	24	100	100	100	100	100
Floor	Material (String)	24	24	24	100	100	100	100	100
Floor	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Floor	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Roof	Quantity (Integer)	32	18	18	56	56	100	56	72
Roof	Dimensions (L,W,H)	96	54	54	56	56	100	56	72
Roof	Material (String)	68	54	54	79	79	100	79	89

Roof	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Roof	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Column	Quantity (Integer)	16	16	16	100	100	100	100	100
Column	Dimensions (L,W,H)	48	48	48	100	100	100	100	100
Column	Material (String)	16	16	16	100	100	100	100	100
Beam	Quantity (Integer)	18	18	18	100	100	100	100	100
Beam	Dimensions (L,W,H)	54	54	54	100	100	100	100	100
Beam	Material (String)	18	18	18	100	100	100	100	100
Total Instances		3368	3056	3050	91	91	100	91	95

A-6 RAP Result for the Testing Model 6

Component/Element	**Item**	**No. of Relevant RBF**	**No. of Extracted RBF**	**No. of Correctly Extracted RBF**	***Sn.* %**	**Sp. %**	**Pr. %**	**Acc. %**	**F1m. %**
Exterior Wall	Quantity (Integer)	48	48	48	100	100	100	100	100
Exterior Wall	Dimensions (L,W,H)	145	145	145	100	100	100	100	100
Exterior Wall	Material (String)	213	213	213	100	100	100	100	100
Exterior Wall	Rough Opening Quantity (Integer)	29	29	29	100	100	100	100	100
Exterior Wall	Rough Opening Dimensions (L,W,H)	87	87	87	100	100	100	100	100
Interior Wall	Quantity (Integer)	80	74	80	93	93	93	93	93
Interior Wall	Dimensions (L,W,H)	172	168	168	98	98	100	98	99
Interior Wall	Material (String)	172	172	172	100	100	100	100	100
Interior Wall	Rough Opening	16	16	16	100	100	100	100	100

	Quantity (Integer)								
Interior Wall	Rough Opening Dimensions (L,W,H)	49	49	49	100	100	100	100	100
Floor	Quantity (Integer)	6	6	6	100	100	100	100	100
Floor	Dimensions (L,W,H)	18	18	18	100	100	100	100	100
Floor	Material (String)	20	20	20	100	100	100	100	100
Floor	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Floor	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Roof	Quantity (Integer)	2	2	2	100	100	100	100	100
Roof	Dimensions (L,W,H)	6	6	6	100	100	100	100	100
Roof	Material (String)	2	2	2	100	100	100	100	100

Roof	Rough Opening Quantity (Integer)	-	-	-	-	-	-	-	-
Roof	Rough Opening Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Column	Quantity (Integer)	13	13	13	100	100	100	100	100
Column	Dimensions (L,W,H)	28	28	28	100	100	100	100	100
Column	Material (String)	8	8	8	100	100	100	100	100
Beam	Quantity (Integer)	-	-	-	-	-	-	-	-
Beam	Dimensions (L,W,H)	-	-	-	-	-	-	-	-
Beam	Material (String)	-	-	-	-	-	-	-	-
Total Instances		1114	1104	1110	99.6	99.6	100	99.6	100