Visual Navigation for Autonomous Material Deposition Systems Using Remote Sensing

by

Soroush Maleki

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

Master of Science

Department of Mechanical Engineering

University of Alberta

Abstract

In this work, a novel visual navigation method is proposed to estimate the state of mobile and fixed cold-spray material deposition systems using a stereo-camera sensor installed in the workspace. Unlike other visual localization algorithms that exploit costly onboard sensors such as LiDARs or fully rely on distinct visual cues on the robot and grid markers in the environment, our method significantly reduces the cost and complexity of the sensory setup by utilizing a cost-effective remote stereo vision system. This allows for the localization of the target system regardless of its appearance or the environment and enables scalability for tracking and operation of multiple mobile material deposition systems at the same time. To achieve this aim, deep neural networks, kinematic constraints, and learning-aided state observers are employed to detect and estimate the location and orientation of the deposition system. A physical model of the system is fused with a remote visual sensing module and is proposed. This accounts for frames in which depth estimation accuracy is reduced due to perceptually degraded conditions in the cold spraying context. The visual state estimation algorithm is evaluated on a fixed and mobile setup that demonstrates the accuracy and reliability of the proposed method.

Moreover, a model predictive controller is formulated, designed, and implemented to enable the task of autonomous mobile robot trajectory tracking. The architecture of the model predictive controller incorporates essential kinematic constraints, input bounds, and control input smoothness, thereby ensuring the generation of a feasible input for the autonomous mobile material

deposition system to seamlessly trace a predefined trajectory. A comprehensive comparative analysis is conducted between this model predictive controller and its PID counterpart, encompassing a rigorous evaluation through a series of simulated and real-world tests, aiming to elucidate their respective performances and characteristics.

Keywords: Visual-based state estimation, deep learning, intelligent manufacturing systems, material deposition, mobile material deposition, state estimation.

In loving memory of the lives tragically lost in the Ukrainian PS752 flight, this thesis is dedicated to their eternal spirit and the enduring impact they leave in our hearts and minds.

Acknowledgements

I would like to thank the Automated Diagnostics for Energy and Environmental Systems lab (directed by Prof. Michael Lipsett) for providing the mobile robot platform for this research. I would also like to express my heartfelt gratitude to my esteemed supervisors, Dr. Ehsan Hashemi, director of the NODE lab, and Dr. André McDonald, director of the AHSTS lab and vice president of strategic research initiatives and performance at the University of Alberta. I extend my gratitude to Dr. Ophelia Jarligo, the research manager of the AHTST lab who has made a significant effort in managing research-related sessions and lab tests. Their unwavering support, guidance, and expertise have been instrumental in shaping my research journey and successful completion of my MSc. Their mentorship and encouragement have inspired me to excel in my academic pursuits.

I am also thankful to the entire research groups at the NODE lab and AHTST lab for fostering a collaborative and stimulating research environment. Their camaraderie and support have been invaluable throughout this endeavor. Lastly, I extend my appreciation to my family and friends for their constant encouragement and understanding, which has been my pillar of strength through the hardships of my master's program.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The integration of autonomous robotic systems has played a pivotal role in driving automation across various industrial applications [1]. These systems have had a profound impact on enhancing safety, efficiency, and productivity in sectors such as warehouse storage automation, manufacturing, medicine, and construction [2, 3]. The utilization of autonomous robotic systems has yielded a remarkable reduction in accidents associated with manual labor while simultaneously improving time and cost efficiency [4]. The growing global demand for these systems has spurred extensive research in robotics and automation, focusing on the development of innovative software and hardware solutions to augment the performance of autonomous systems in existing use cases while also exploring new avenues for their integration into everyday human activities.

The deployment of autonomous robotic systems for automated material deposition and maintenance tasks holds substantial promise for various industries. The ability to automate these processes not only increases efficiency but also offers significant advantages in terms of precision, consistency, and safety. Automated material deposition, for instance, finds applications in diverse domains such as coating, painting, and surface treatment. By leveraging autonomous robotic systems, manufacturers can streamline their production processes, achieve high-quality outputs, and reduce waste [5].

In the context of maintenance, the use of autonomous robotic systems allows for timely and accurate inspections, repairs, and maintenance operations in challenging environments that may be hazardous or difficult to access for

human workers. This includes tasks such as structural inspections, pipeline maintenance, and equipment servicing. The adoption of autonomous systems for maintenance activities brings about improvements in efficiency, reliability, and safety while minimizing downtime and reducing costs [6].

To enable the successful implementation of automated material deposition and maintenance tasks using autonomous robotic systems, two crucial components must be addressed: state estimation and control algorithms. In state estimation, the goal is accurately determining the position, orientation, and other relevant states of the robot. Precise state estimation is essential for reliable trajectory planning, obstacle avoidance, and interaction with the surrounding environment. Furthermore, control algorithms are required to ensure that the robot accurately follows the desired trajectories, maintains stability, and achieves the desired performance metrics.

In light of these considerations, the present research aims to contribute to the advancement of state estimation and control algorithms for automated material deposition and maintenance using fixed or mobile robotic systems. The development of such algorithms in these areas will enhance and enable robotic platforms to perform complex tasks with precision and reliability. By introducing novel approaches to state estimation and control that can be applied to a wide range of robotic systems, this research aims to propel the adoption of autonomous robotic systems in various industrial domains, thereby promoting increased efficiency and improved safety.

1.1 Motivation

In the context of surface treatment and fabrication, both in industrial and academic settings, the automation of primary manufacturing tasks assumes paramount importance. A precise and repeatable material deposition is a critical requirement in various industries, which can decrease variability and inconsistency, leading to compromised product quality. The controlled movement of material tools along pre-defined paths, with specific characteristics such as stand-off distance, number of passes, angle of attack, and tempera-

ture and pressure parameters, necessitates the use of automated systems to achieve precise and consistent material deposition. Moreover, numerous surface treatment applications are situated in environments inaccessible to human workers due to hazardous conditions or challenging terrain. Examples include the treatment of corroded pipelines traversing deserts or the maintenance of radioactive nuclear power plants.

The implementation of automated material deposition systems brings substantial efficiency and speed to surface treatment tasks, resulting in cost and time savings, and reducing the likelihood of human errors. These automated systems are highly adaptable, requiring minimal hardware and software changes when dealing with varying target surface sizes and shapes. This adaptability empowers the production of tailor-made products, catering to specific customer demands and enhancing manufacturing flexibility. The integration of such systems aligns with sustainable manufacturing practices, optimizing material usage and minimizing waste, contributing to resource efficiency and environmental preservation. Furthermore, the pursuit of automation in surface treatment and material deposition opens up avenues for interdisciplinary collaborations and innovative advancements in fields such as robotics, material science, computer vision, and artificial intelligence, driving educational and research development.

In the proposed approach, we take advantage of a static sensory unit namely, a camera that is fixed on a tripod in the working environment. The utilization of a stationary/remote visual sensing system is strategically advantageous for countering perceptually degraded conditions caused by factors such as dust, dynamic object movement, and varying lighting conditions. This approach leverages the stability of the stationary setup and its elevated position in the 3D space, which enables us to identify dynamic objects in the framework. Moreover, in contrast to an onboard sensory system, an external sensory node offers a comprehensive view of the target system, enhancing its resilience against issues like dust, material debris, and variations in lighting conditions.

The potential applications of material deposition extend far beyond sur-

face treatment. It encompasses a diverse range of tasks, such as autonomous painting, welding, coating, surface finishing, 3D printing, and additive manufacturing. The development of a general framework for autonomous control and navigation of robotic agents is a central motivation in this research endeavor. By exploring the frontiers of automation in material deposition, this work aims to revolutionize manufacturing processes, providing not only improved efficiency and productivity but also safer and more sustainable solutions for industries worldwide. As automated material deposition technologies continue to evolve and advance, their transformative impact will extend to various sectors, enhancing industrial capabilities and fostering innovation in cutting-edge technologies.

1.2 Robotic maintenance and deposition systems

The endeavors made by robotic engineers and scientists in developing autonomous robotic systems with the application of material deposition and surface treatment, maintenance, and manufacturing can be summarized in two main categories. The first category of such systems capable of enhancing automation for industrial purposes encompasses mobile robotic systems that can navigate a complex environment. Furthermore, these robotic systems can be further modified to enable better control over the workspace through manipulators. The figure below demonstrates two examples of such mobile robotic systems with the application of surface treatment.

Moreover, by employing fixed robotic systems, manufacturers can achieve consistent and high-quality results while improving productivity and reducing costs when performing surface treatment in the industrial context. These systems are designed while ensuring the safety of workers while leveraging the precision and efficiency of robotic automation. Similar to their mobile counterparts, these robotic systems can also be equipped with necessary maintenance tools for the task of autonomous manufacturing, welding, material deposition, and much more. The application of such systems is through the following

(a) (b)

Figure 1.1: Two prominent examples of mobile surface treatment systems. (a) is an autonomous mobile robotic system designed for enhancing the surface durability of ships by depositing anti-corrosion materials made by AMBPR© [7]. (b) showcases a mobile material deposition (cold spray) system, engineered by students at the University of Alberta.

figures.

In summary, the mobile and fixed maintenance and material deposition systems can improve efficiency, precision, and reliability all enabled by innovative algorithm development in state estimation and controls. Having enumerated and expanded on autonomous material deposition systems and why they play a significant part in automation, some of the ongoing research is presented in two principal components of their autonomy stack, namely the state estimation and controls modules as the main focus of this thesis in the next few sections.

1.3 Problem statement

The aim of this research is to incorporate autonomous material deposition and surface treatment systems performed by fixed and mobile robotic systems in order to increase safety and product consistency. However, in order to introduce automation, the focus is on the primary modules in an autonomous surface treatment system, namely state estimation and controller design. This provides the opportunity to localize and control autonomous material deposi-

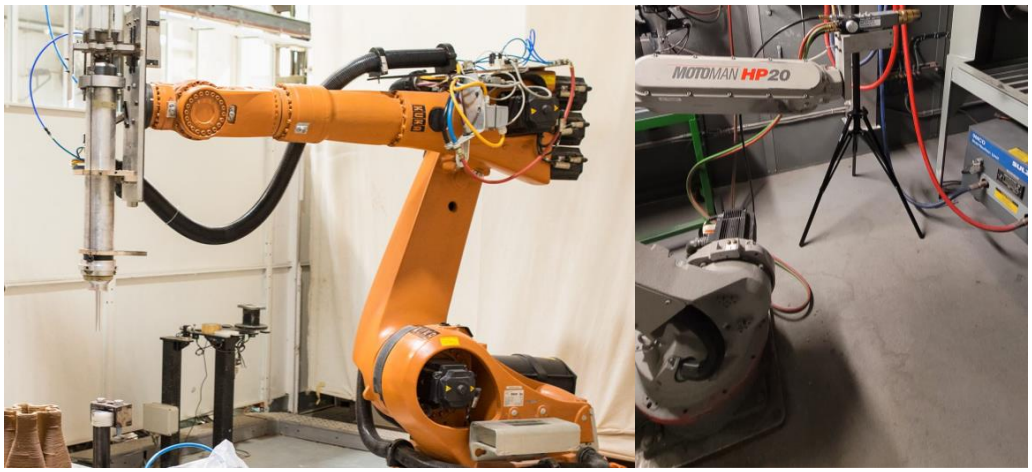

Figure 1.2: Two examples above demonstrate the application of fixed material deposition systems in industrial applications. Picture (a) exemplifies a DDSPLM additive manufacturing unit [8] with a deposition nozzle installed while (b) demonstrates the fixed manipulator used for the task of autonomous material deposition at the AHTST lab at the University of Alberta.

tion systems even in remote areas where accessibility is not feasible.

The problem addressed in this thesis research is twofold; The first part involves estimating the state of the fixed or mobile material deposition system through a fixed camera in the environment. Having attained the states (position and orientation) of the autonomous agent in the face of uncertainties, the second part involves the implementation of PID and MPC controllers to help navigate the autonomous agent in the environment.

Traditional control methods, such as PID controllers, may suffer from deficiencies, including agnostic handling of system constraints, occasional overshoots, and undershoots. Hence, there is a need to develop more advanced control algorithms to optimize material deposition and surface treatment procedures. To overcome these challenges, more advanced control algorithms, such as model predictive control (MPC), will be developed and implemented for trajectory tracking of robotic systems in the context of autonomous material deposition and surface treatment. MPC has shown significant promise in providing greater flexibility in control design by incorporating performance metrics and hard constraints on input and state variables. By predicting fu-

ture system behavior over a finite horizon and optimizing control inputs accordingly, MPC can lead to stable and accurate trajectory tracking, ensuring precise material deposition and surface treatment in real-world scenarios. Enhancing state estimation accuracy will be explored through the integration of fixed visual sensors and external landmarks to remotely estimate the robot's position and orientation. Developing efficient state estimation algorithms will enhance the robotic system's situational awareness, leading to improved path following and precise material deposition and surface treatment.

Additionally, the research will investigate the impact of noise in state estimation and control commands, which can arise from uncertainties in the environment and sensing modules. Evaluating the the developed MPC algorithms against existing noise will be crucial in ensuring the stability and reliability of the system in real-life scenarios.

1.4 Autonomous manufacturing and material deposition

The autonomous material deposition process is a high throughput additive manufacturing technique that combines and utilizes the speed and heat of projectile material particulates with high fluid momentum loads to deposit a layer of free-form structures and coatings. Such thermal spray processes enable the manufacturing of a wide range of material deposition layers on substrates of different geometries such as planar, cylindrical, and even irregularly-shaped surfaces with large dimensions [9]. Such particles usually come in a variety of different sizes from 1 to 50 μm in diameter. The particles are accelerated to high velocities using a combustion flame or a cold spray deposition unit [10]. The successive inter bonding among the splats causes a build-up of deposition materials, leading to a well-bounded layer with a thickness of more than 10 μm [10].

Integrating principles in automation and advantages caused by them in manufacturing-related areas such as surface treatment has been the focus of many studies up to date. A select number of such scientific works related to

the area of integrating automation of material deposition and surface manufacturing were explored in this section. For instance, in a recent study by Jiangzhuo et al. [11], the need for optimizing the path for achieving desired coating thickness in thermal spraying techniques using robot manipulators and handling systems was addressed. A parametric coating thickness prediction model was utilized, and the Nelder-Mead method was employed to determine the optimal kinematic parameters of a zigzag path for uniform coating with the desired thickness. The developed prototype system integrated these functions and provided an optimal path directly, validated through a case implemented with a homemade spray system and a robot system. Additionally, Nylen and Edberg introduced a simulation method to optimize robot trajectories for achieving uniform coating thickness and predicting transient coating temperatures on complex geometries [12]. The trajectory was optimized to ensure a constant spraying distance, normal orientation to the surface, and adherence to non-collision requirements, while the plasma was represented using a simplified turbulence model and discrete particle model. As another instance of work done on multiaxis robots for autonomous material deposition, Candel and Gadow [13] emphasized the importance of high-accuracy robot systems for efficient and reproducible thermal spray processes. Specialized software tools were developed to simulate, generate, and implement torch trajectories, considering their impact on coating deposition. Case studies demonstrated practical applications, showcasing how such tools optimized robot trajectories for autonomous material deposition and surface treatment. This work was continued by Bolot et al. [9], which focused on offline robot trajectory generation for thermal spray applications, integrating it with thermal history computation to analyze thermal stresses during the spray operation, considering impinging plasma jet and molten particle jet contributions.

In addition to the studies above, scientific approaches were developed for optimizing robotic trajectories for performing thermal coating deposition to increase the thermal and thickness efficiency of deposition. In a study by Z. Cai et al. [9], the authors proposed a mesh-based trajectory generation approach for thermal barrier coatings, focusing on optimizing the robot trajectory to

achieve coating thickness homogeneity and uniform heat transfer distribution on the coating surface. They demonstrated that by doing so, they increased the effectiveness of deposition by increasing the uniformness of the temperature field. In another study by Hegels et al., post-optimization was utilized to increase the efficiency of a designed path by making the thickness of the deposition uniform across the deposition substrate [14]. The method used a high-quality GPGPU-based simulation of the spray process to evaluate coating thickness error and kinematic path quality, efficiently delivering improved paths that reduced the coating error on real free-form surfaces by up to 3.5% of the original value in every case study. The post-processing optimization aimed to reduce thickness error caused by imprecise design, adapt the path to changes in spray gun or technology, accommodate slight incremental changes in workpiece geometry, and improve path execution by the robot.

In general, the reviewed papers on the usage of robotics in automation demonstrate significant advancements in various fields, ranging from industrial manufacturing to surface treatment and material deposition. These studies have explored the integration of robot systems for tasks such as trajectory generation and optimization, path planning, and state estimation, leading to improved process reproducibility, coating quality, and efficiency. However, despite the progress made in these areas, some limitations persist, including challenges related to handling deposition environments in which a fixed material deposition unit may not be able to operate, as well as addressing the state estimation part, which may not be possible to perform using the conventional localization techniques due to the existence of visually degraded environments in the material deposition context. In the next section, some of the existing studies that address these challenges in state estimation and controls are summarized, providing further insights regarding the necessity of this work.

1.5 Visual based state estimation

The task of visual robotic state estimation has been extensively researched in the field of robotics. As highlighted in the preceding discussion, the implemen-

tation of autonomous robot navigation and operation in indoor environments has led to the automation of diverse tasks, ranging from warehousing operations to transportation services and automated maintenance [15, 16]. At the core of enabling such complex tasks lies the crucial step of acquiring environmental information to accurately estimate the state of the robot.

In the field of visual state estimation, challenges arise when static landmarks are solely relied upon [17], where identifiable features continuously vary in location over time. Consequently, the performance of such methods may degrade, and accurate state estimation becomes more challenging. Additionally, errors due to wheel slip can lead to state estimation drift over time [18] when IMU sensory data is integrated with wheel odometry. These limitations call for more sophisticated state estimation approaches that account for dynamic environments and mitigate drift issues.

Moreover, conventional multi-modal methods for state estimation that rely on onboard sensory setups, such as cameras, LiDAR, and Global Navigation Satellite System (GNSS), may encounter difficulties in specific environments. In cases where visual cues are lacking, such as environments with limited textures for visual node [19], or environments lacking 3D distinct features for LiDARs [20], the performance of visual and LiDAR-based state estimation methods may be compromised. Similarly, in confined environments or regions with obstructed GNSS signals, traditional GNSS-based methods may face challenges [21].

One possible solution to the state estimation problem is to use fixed sensors in the environment and estimate the states of the system remotely. This becomes exceptionally vital in scenarios where a network of robots operates to carry out a certain task, where the cost associated with replicating the sensory system on individual robots may become prohibitive to implement in real-life [22]. The integration of fixed visual sensors with a limited field of view on the robot's workspace, as well as active landmarks with a specific design to determine the position and orientation of the robot clearly, was proposed by Jankovic, et. al [23]. The performance of this method can be affected by improper illumination and the landmark's unique design, as well as color

thresholding of the landmark for image processing.

The proposed method by Shim and Cho [24] relied on installed indoor surveillance systems using several neighboring cameras, constructing two-dimensional maps from multiple image planes. However, this method lacked support for multiple robots and assumed prior knowledge of the environment's structure, necessitating an extensive calibration process for the camera network. In a similar study by Babinec et al., localization was performed by utilizing markers developed for augmented reality, installed in the environment [25]. The accuracy of this method is influenced by the number of markers in the environment. State estimation was achieved for automated guided vehicles by utilizing data fusion of an externally installed camera and internal laser range scanner to reduce estimation uncertainty [26]. The pose of an RGB-D camera was estimated by Fei et al. [27] by incorporating convolutional neural networks (CNN) for extracting image features and long short-term memory (LSTM) units to consider the temporal data of the image frames. In another work by Song and Chang [28], a multi-camera mobile robot poses estimation was proposed, where the pose of a mobile robot was estimated by utilizing a marker installed on the robot, and the occlusion problem was addressed by switching between the cameras as the robot left the field of view of the ego camera. Lastly, a two-stage localization and instance identification technique was proposed to estimate the pose of a mobile robot [29]. The first stage provided the robot bounding boxes for the orientation estimation convolutional neural network. However, this method was highly dependent on the positioning of the camera, environment lighting, the robot's shape and structure, and the color of the ground on which the robot was operating.

Similarly, Feng et al. [30] considered the three-dimensional localization problem of a mobile robot and used a network of wirelessly connected visual sensors to perform the recursive least squares localization method. This approach also relied on the color information of the mobile robot, and the stability of the network communication influenced the robustness of this method. The mobile robot's drift challenge over time was addressed by Flögel et al. [31], where a slip-aware kinematic model of the robot's motion was utilized to

forecast the motion of the robot, and this model's prediction was fused with an object detection-based visual tracking state observer output for the location and orientation of the robot. A similar approach was implemented in [32] where the depth map of each frame was obtained through a monocular disparity estimation neural network, and the 2D bounding box on image frames was applied to obtain the position of the mobile robot.

In other works addressing the problem of visual 3D pose estimation of objects, complex neural networks were designed and developed to learn visual features in the input images and output the pose of the target object in an end-to-end manner. For instance, Hu et al. [33] introduced a novel 3D object detection method for mobile autonomous units, combining monocular image input with cascade geometric constraints to achieve robust detection. The framework consisted of two stages: the first stage used CenterNet with an additional branch to regress orientation, dimension, and center projection of the bottom face. In the second stage, cascade geometric constraints filtered out imprecise 2D bounding boxes, leading to improved 3D box output. The proposed method did not rely on external sources and could be trained end-to-end. As an additional example, the method proposed in [34] addressed the problem of occlusion and improved monocular 3D object detection in autonomous driving by considering the relationship of paired samples, encoding spatial constraints for partially-occluded objects from their adjacent neighbors, and jointly optimizing uncertainty-aware predictions for object locations and 3D distances through nonlinear least squares. To exemplify methods that relied on deep learning to extract 3D information of mobile agents, [35] introduced a novel pipeline for 6D object pose estimation from RGB-D images using raw point clouds, demonstrating robustness against object variations and occlusions and outperforming state-of-the-art methods on the well-known LINEMOD and Occlusion LINEMOD datasets.

There are several critical restrictions that prevent the application of such end-to-end deep learning-based methods for 6 degrees of freedom state estimation. Firstly, the development of such a method requires extensive manually labeled 3D datasets [36, 37] that cover a variety of cases for target object

localization. Furthermore, gathering and labeling such vast datasets can be time-consuming and costly, requiring many hours of work and manual actions. In addition, the kinematic constraints and physical nature of the problem seem to be overlooked in the end-to-end learning methods. Moreover, one may face a trade-off between accuracy and real-time performance when heavily relying on end-to-end machine learning (ML) based methods. These rely on complex networks, which prohibit their use in real-world applications with limited computing power in which creating and labeling the datasets is not feasible. Finally, it seems that there is a gap in the literature in finding the pose of arbitrary objects, with a restricted number of training instances. When looked at as a whole, the 3D object detection community mostly focuses on autonomous driving applications, and the existing works on detecting and tracking the pose of objects outside of this application are limited, which will be addressed in this work.

1.6 Motion planning and controls using stationary sensors

The challenge of achieving autonomous maneuvering for mobile robots has been extensively explored in prior research. Numerous conventional approaches have evolved into optimal control-based methodologies, wherein control inputs are computed at each time step [38], considering both historical and predictive error information. This focus on model predictive controllers stems from their effectiveness in trajectory-tracking tasks, which makes them a compelling choice for our investigation [39].

The trajectory-tracking challenges in autonomous mobile robots are typically addressed by devising control laws that enable the agents to follow predetermined, feasible trajectories. However, this approach encounters limitations due to the complexities of the vehicles' dynamics [40], which entail nonlinear terms and significant uncertainties, making the computation of feasible trajectories a formidable task. Moreover, in the presence of tracking errors, controllers may attempt to synchronize outputs with time-parameterized de-

sired outputs, leading to difficulties in closed-loop performance and generating excessively large control signals [41].

Path-following problems revolve around the design of control laws guiding an object to reach and adhere to a geometric path, while simultaneously satisfying additional dynamic specifications. In this study, the application of model predictive control (MPC) theory to path tracking for mobile robots, particularly in the context of nonholonomic mobile robots, is focused on. The investigation encompasses both linear and nonlinear approaches, addressing the primary research concerns related to MPC and nonlinear model predictive control (NMPC) in mobile robotics. Instead of taking a global perspective on MPC theory, this literature review centers on its specific application to mobile robotics. Given the high dynamism of mobile robot systems, the challenges posed by path tracking are noteworthy and warrant comprehensive investigation.

As a prominent example, a learning-based nonlinear model predictive control (LBNMPC) algorithm was introduced by Ostafew et al. [42] to enhance path-tracking accuracy for autonomous mobile robots during repeated traversals along a reference path. The LB-NMPC algorithm leveraged a straightforward a priori vehicle model and incorporated an adaptive disturbance model. To account for disturbances, they employed a Gaussian process (GP) based on data collected from prior traversals, considering the system state, input, and other pertinent variables. Similarly, Lim et al. [43] proposed a nonlinear model predictive tracking control scheme tailored for a nonholonomic unmanned ground vehicle (UGV). Their approach combined high-level guidance control utilizing kinematic approximation for UGV motion with an NMPC algorithm to address trajectory planning and optimal control problems. Furthermore, an updated investigation by Ostafew et al. [44] focused on outdoor mobile robots. In contrast, indoor mobile robots were often preferred for research due to their controlled environments, which allowed for more in-depth exploration of specific issues. Many researchers focused on obtaining feedback laws that ensured the asymptotically stable equilibrium of the closed-loop system [45]. As a result, MPC theory emerged as a potential solution to address

this control problem. MPC aimed to solve optimization problems by predicting the system's behavior over a specific horizon, considering input and state constraints. Ensuring system stability with the chosen prediction horizon became a major concern, and studies showed that an infinite predictive horizon could guarantee stability for a system [46]. Nonetheless, the choice of an infinite predictive horizon might not be feasible for practical nonlinear systems. Several approaches to optimal control were introduced, including MPC [47–49]. Although MPC methods vary in model representation, noise representation, and cost function minimization techniques, they shared a similar structure and offered the necessary degrees of freedom to address the control challenges effectively. As Findeisen and Allgöwer [50] explained, the model predictive control (MPC) problem entailed solving an online, finite horizon open-loop optimal control problem, taking into account system dynamics and constraints concerning states and controls. Leveraging the system model and measurements obtained at time t, the MPC controller predicted the future dynamic behavior of the system over a prediction horizon and subsequently determined the input that optimized a predetermined open-loop performance objective function. In ideal scenarios without disturbances or model-plant mismatch, and under the condition that the optimization problem could be solved for infinite horizons, this enabled a coherent and consistent control strategy for the entire system operation. Indeed, NMPC implementations showed promise, but they came with challenges related to nonconvex optimization[51, 52]. Consequently, the tuning of controller parameters became crucial to achieving satisfactory performance. To ensure asymptotic stability, NMPC typically required terminal state constraints [53].

Researchers have made significant efforts to address the challenges posed by online optimization in NMPC and achieve asymptotic convergence of tracking error. For instance, Hedjar et al. [54] propose the use of Taylor series approximation in the prediction model to tackle the online optimization issue. With advancements in computational power, experiments have been performed using more complex robot models [55–57]. Moreover, NMPC has been extended to handle obstacle avoidance in scenarios involving nonholonomic robots [58–

61]. Recent research efforts have also focused on the stabilization of nonholonomic mobile robotic systems [62]. Furthermore, Abbas et al. [63] conducted a study to assess the performance of NMPC controllers concerning the look-ahead horizon. They compared two different obstacle avoidance methods and tested NMPC in a range of simulated but realistic tracking scenarios that involved static obstacles on constrained roadways.

The problem of stability has also been well-addressed in the context of applying MPC in mobile robots [50, 64]. However, in the context of inherently nonlinear systems, particularly in highly dynamic environments like mobile robotics, linear models often fall short of accurately describing the system dynamics. As a result, nonlinear models become necessary, leading to the motivation for employing nonlinear model predictive control (NMPC) techniques [50, 64]. To accommodate the control of time-varying nonlinear systems with input constraints, Fontes proposed a novel NMPC framework in 2001 [65]. This approach introduced a set of design parameters that enable a priori verification of the stabilizing properties of the considered control strategies. By relaxing traditional assumptions on the continuity of the optimal controls and the stabilization of the linearized system, the class of addressable nonlinear systems was significantly expanded, including certain nonholonomic systems that could be stabilized effectively by NMPC [65]. Gu and Hu delved into the application of receding horizon (RH) control for nonholonomic mobile robots, aiming at regulation and tracking control [66, 67]. Stability in RH control was ensured by incorporating a terminal-state penalty in the cost function and constraining the terminal state within a specific region in the optimization constraints. Their work revealed that the RH tracking control had the remarkable capability of simultaneously achieving both tracking and regulation objectives.

Furthermore, Yang et al. addressed the stability issue in MPC from the perspective of formation control and obstacle avoidance in a group of nonholonomic mobile robots [68]. They introduced two control algorithms, formulated to solve the optimal control problem while considering cost functions coupled with the dynamics of each interacting robot. The incorporation of a poten-

tial function defined the terminal-state penalty term, and a corresponding terminal-state region was added to the optimization constraints.

These studies represent significant contributions to the development of NMPC for nonlinear systems, particularly in the context of nonholonomic mobile robotics, and their findings pave the way for more sophisticated and robust control strategies in dynamic and uncertain environments. The continued exploration of NMPC techniques holds the potential to revolutionize the field of robotics and enable the safe and efficient operation of autonomous systems in a wide range of applications, including path planning, formation control, obstacle avoidance, and beyond.

1.7 Research objectives and contributions

As highlighted in the previous sections, accurate and reliable localization, control, and navigation of autonomous mobile robots using fixed low-cost sensors in the autonomous material deposition context remain an open challenge. Solving this challenge can lead to increased accuracy, repeatability, and cost-effectiveness of the robotic surface treatment systems, therefore, decreasing human labor. In this study, a unified novel visual localization and navigation framework is proposed to estimate the state of a mobile cold spray material deposition system and to control the autonomous material deposition unit on a predefined path. The main goals and contributions to this work are:

- Incorporating *deep learning* into the framework while reducing the number of training examples needed to implement the framework, by addressing depth estimation and object detection in separate modules.
- Developing a learning-aided *object localization and orientation estimation* framework, independent of the physical structure of the environment or the robot.
- Designing and developing a *physical motion model* of the system consisting of a material deposition head and a mobile platform.

- Fusion of the physical motion model of the system with the remote sensing module's output to account for perceptually degraded conditions.
- Incorporating optimization to impose *kinematic constraints* on the visual state-observer.
- Developing and implementing a *model predictive controller* for the task of trajectory tracking, while taking into account the kinematic constraints of the material deposition agent.
- Unifying the state estimation and control algorithms in a single framework called "infrastructure-aided visual navigation", allowing the surface treatment scientists to incorporate automation to increase reliability, and deposition accuracy in their material deposition tasks.

1.8 Thesis organization

In this thesis, a comprehensive exploration of incorporating autonomy into the surface treatment industry with a specific focus on autonomous material deposition using fixed and mobile robotic systems is undertaken. The essential need for autonomous systems in surface maintenance is emphasized, highlighting the significance of developing necessary modules for material deposition. The organization and structure of the subsequent chapters are laid out, collectively contributing to achieving the research objectives.

The methodology adopted for state estimation of the autonomous agent, a crucial aspect in ensuring accurate and reliable performance, is delved into in the ensuing chapters. The experimental setup, encompassing both fixed and mobile manipulators, is elaborated, followed by an in-depth explanation of the visual state estimation framework. The subsequent chapter extensively elucidates the model predictive algorithm employed for the precise control of the robot along predefined trajectories. The research then presents the results, showcasing a thorough and comprehensive investigation of the state estimation framework and the model predictive control algorithm. This analysis encompasses both quantitative and qualitative aspects, allowing for a comprehensive

evaluation of the proposed methodology in comparison to the traditional PID counterpart.

The conclusion section serves as the final summation, consolidating the findings derived from the various research chapters. By reiterating the core outcomes and contributions of the thesis, it underlines how the proposed methodology effectively addresses the challenges associated with visual navigation in the domain of autonomous material deposition. Through this organization, the thesis aims to provide a systematic and in-depth exploration of the potential of autonomous material deposition systems, contributing to the advancement of autonomous surface maintenance applications.

Chapter 2

Visual state estimation for material deposition systems

In this chapter, the proposed methodology for estimating the state of an autonomous material disposition agent in a working environment will be explored. The real system setup used for experimentation will be described, providing essential details of the visual state estimation framework. Emphasis will be given to possible choices for 2D detection, a crucial step in identifying the target agent for localization.

The investigation will continue to cover 3D projection, point cloud processing, filtering, and optimization steps, all essential in determining a 3D bounding box for the robotic agent. Additionally, the implementation of a learning-aided state observer will be studied, enabling the production of a smooth and consistent estimation of the system's state.

It's crucial to highlight that our visual state estimation framework is designed to effectively address the various forms of noise inherent in sensory data. Ensuring swift convergence to accurate states demands meticulous noise management. To achieve this, we introduce statistical filtering techniques for refining dense point cloud representations, along with a constant acceleration motion model to counteract visual sensor-related noise, particularly tackling occlusion-related noise. Additionally, we integrate adaptive covariance matrices to model noise across visual and encoder sensory nodes, a key element in the Kalman filter fusion process.

By presenting the step-by-step process of the proposed methodology, this

chapter aims to provide a comprehensive understanding of how autonomous material disposition agents' state estimation is achieved in real-world working environments. The utilization of visual state estimation, along with 2D and 3D processing techniques and learning-based state observers, forms a framework for accurate and reliable state estimation in this context.

2.1 System setup

This section is dedicated to illustrating the experimental setups utilized to generate a proof of concept for our methodology. To examine the generalizability and scalability of our framework, we will consider the case of a mobile and fixed material deposition system.

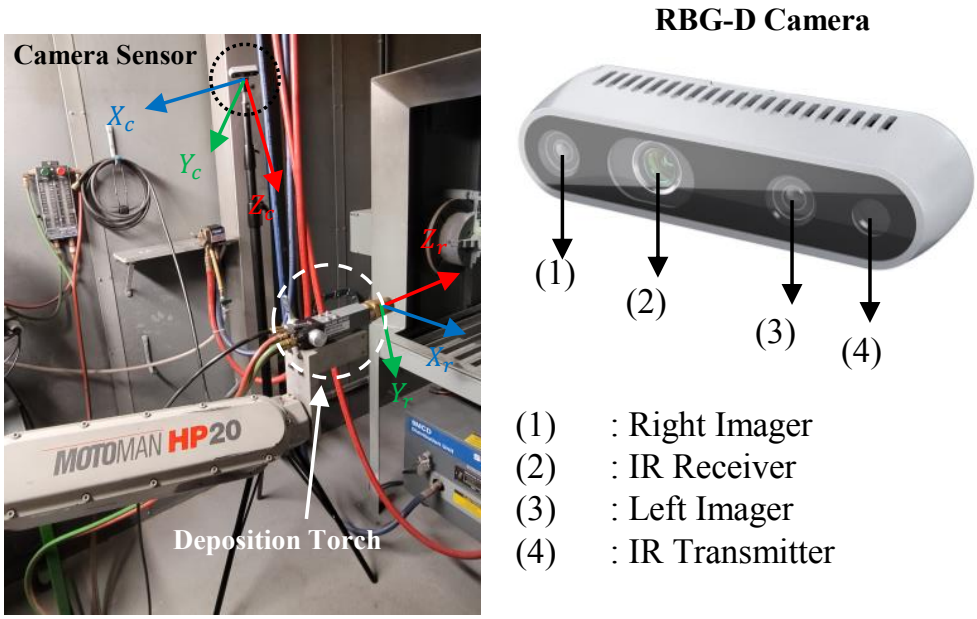

Figure 2.1: System setup including the 6 dof MOTOMAN HP20 manipulator, the deposition torch (dashed white circle), RGB-D camera shown on right (dotted black circle), Position (three-dimensional) and Orientation (three-dimensional) of the torch is estimated using the camera sensor.

The experimental setup for visual state estimation involved setting up a stereo camera sensor on an elevated sensor holder device, as depicted in Figure 2.1. The main components of the experimental assembly included a flame spray

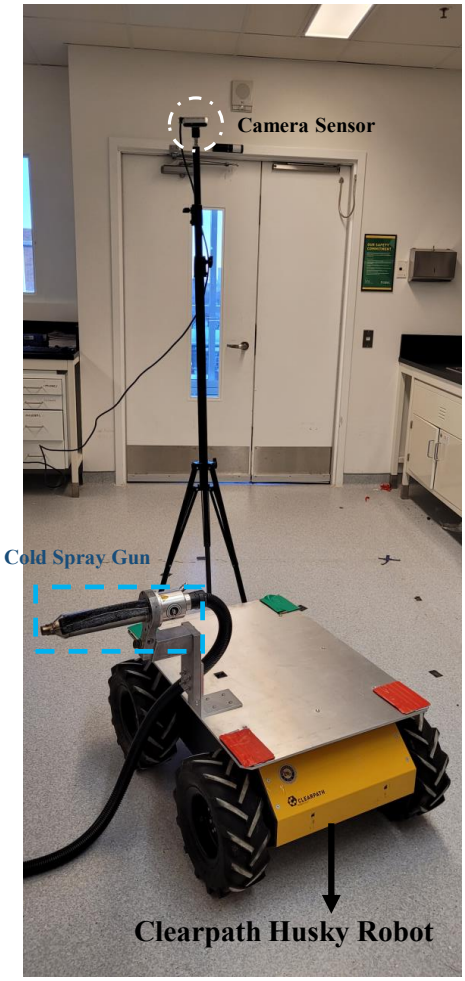

Figure 2.2: System setup including the Husky Robot made by Clearpath, cold spray gun, and camera sensor are shown. The goal is to estimate the position (two-dimensional) and orientation (one-dimensional) pose of the mobile robot.

material deposition system (6PII, Oerlikon Metco, Westbury, NY, USA) used as the test object.

The algorithm's performance was tested and quantitatively evaluated on two cold spray material deposition systems for visual state estimation. The first system was the MOTOMAN HP20, a 6-degree-of-freedom manipulator from Yaskawa America, IL, USA, used for automated material deposition tasks in indoor environments. The setup of this system can be seen in Figures 2.1 and 2.2, which illustrate the deposition torch, the camera sensor, and the body-fixed coordinate systems of both the deposition torch and the camera.

The mobile deposition system, on the other hand, consisted of a 4-wheeled

Intrinsic Parameters		
Description	**Property**	**Value**
Focal Length in X direction	Fx (pixels)	382.73678
Focal Length in Y direction	Fy (pixels)	382.73678
Principle Point X position	Cx (pixels)	321.24
Principle Point Y position	Cy (pixels)	239.32
Baseline	B (mm)	50
Camera Skew Parameter	s	1

Table 2.1: Table of stereo camera sensor properties and values

mobile robot, specifically the Clearpath Husky robot from Clearpath Robotics, Kitchener, Ontario, Canada, along with the camera sensor and the cold spray deposition gun. The dimensions of the manipulator's end-effector, which was the target for localization, were 0.1m, 0.057m, and 0.25m. The robot was controlled using the NX100 controller with standard Ethernet connectivity to personal computers.

The second test case involved the mobile robot, Husky, which is designed for heavy-duty applications in both indoor and outdoor environments. It had dimensions of 0.99m, 0.67m, and 0.39m. A holder was specially designed and manufactured to install the cold-spray nozzle on this robot for remote operation.

The stereo camera sensor used in the experiments was the Real Sense depth Camera D435i from Intel, California, US. It comprised two grayscale cameras (known as imagers), one RGB camera capable of recording color information, and one infrared projector. Although the sensor could generate RGB-D images, we decided to use the grayscale cameras for depth estimation using the CoEx network. The camera sensor captured 30 frames per second, with a horizontal and vertical field of view of $87^o \times 58^o$ and a frame size of 480×720 pixels.

The camera sensor properties and intrinsic parameters were obtained through calibration. Notably, the stereo camera sensor described in Table 2.1 was capable of rectifying the images and accounting for lens distortions internally. Hence, the sensor's output did not require further operations before being used in the subsequent steps of the visual state estimation framework.

2.2 An overall look at framework

This section outlines the research methodology employed in this work, with a high-level perspective. We start by illustrating the contents of Figure 2.3. In the first step, after receiving the left and right images from the camera sensor, a learning-based method is used to generate a dense point cloud representation of the target object. Confining our analysis to this point cloud instead of the full environment contributes to the efficiency of our algorithm. After this step, a 3D bounding box of known size is fit to this bounding box using an optimization process which will be described in a future section. Knowing the location and orientation of this bounding box, we use the predictions of the physical motion model of the system to refine the estimated state through an optimal variance filter, leading us to estimate the object's pose. In the next sections, we will discuss each stage in greater depth.

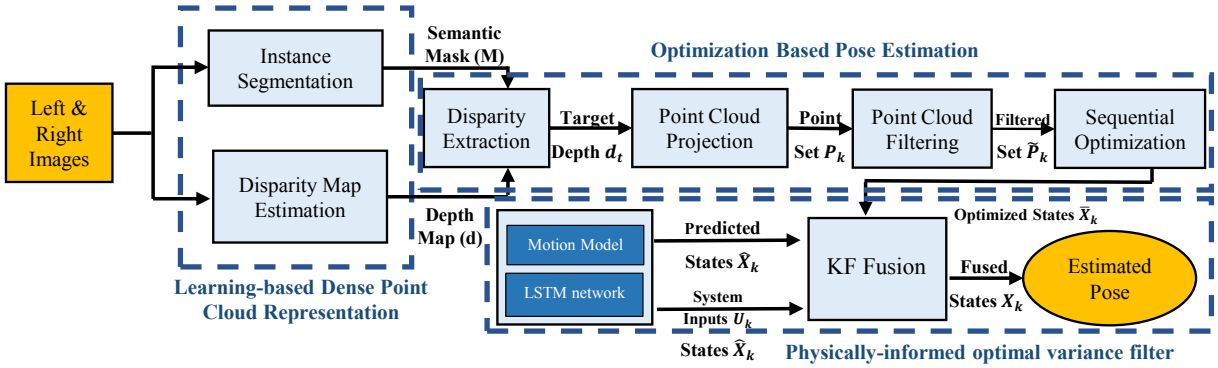

Figure 2.3: Visual navigation and remote sensing framework for the mobile material deposition systems

2.3 Detection methods

This section focuses on the localization of the target object in one of the two gray-scale 2D images, with the left image chosen arbitrarily for this purpose. There are various approaches for detecting objects in images, each dependent on the specific application. In this context, we will describe three existing methods and explain why we opt for our chosen approach.

2.3.1 Two-dimensional Object Detection

When dealing with two-dimensional object detection, the goal is to find a rectangular bounding box that encompasses the intended target object(s). This is achieved by training complex convolutional neural networks capable of learning various patterns from a training dataset.

The process of two-dimensional object detection comprises two main steps. The first step involves detecting the target object by estimating the x and y coordinates of the top-left corner and bottom-right corner of the bounding box. This information is crucial for the subsequent task. The second step is object classification, where the detected objects are categorized into different classes. The rectangular bounding boxes provide regions of interest (ROI) that contain the objects.

The example in Figure 2.4 demonstrates the output of a widely used 2D bounding box detector known as YOLO V5 [69]. This particular detector was trained on a dataset of approximately 300 images. Although two-dimensional object detection offers advantages in various scenarios, it may face limitations in providing a tight and fine-grained bounding box for a target object. Consequently, when dealing with sufficiently large target objects, the bounding box might encompass other obstructing objects, leading to imprecise localization.

Given this limitation, it becomes imperative to explore alternative 2D localization techniques that are better suited for our specific application. The goal is to find a method that can offer higher precision and accuracy in localizing the target object, particularly in environments where precise and detailed object localization is crucial for the success of autonomous material disposition tasks.

2.3.2 Instance segmentation

Instance segmentation is a widely adopted computer vision technique for detecting objects with pixel-level accuracy. Unlike other segmentation methods that group an image into regions or objects, instance segmentation goes beyond mere identification by tracking multiple instances of the same object

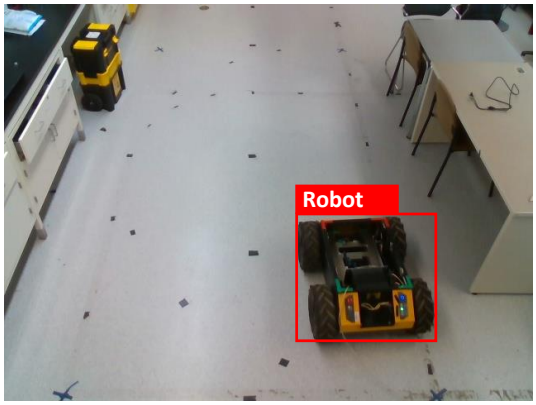

Figure 2.4: Detecting the mobile robot using the YOLO V5 framework in the lab environment.

within an image. This capability proves particularly advantageous in scenarios involving networked robots, where detecting and tracking each robot in the image becomes crucial for state estimation.

By accurately locating and segmenting individual objects in an image, instance segmentation assigns a unique label to each pixel belonging to an object instance. Additionally, instance segmentation networks can differentiate between overlapping and occluding objects, a critical challenge in infrastructure-based visual state estimation. Considering these benefits, we have chosen instance segmentation as the primary component of our 2D detection module.

In our work, we employ OrienMask [70], a real-time instance segmentation framework that addresses the challenge of designing highly accurate image segmentors. Built around YOLO V3, this framework incorporates a mask head that predicts orientation maps, serving as displacement vectors to differentiate between foreground and background in an image. The discrimination ability of these orientation maps eliminates the need for additional foreground segmentation, allowing for efficient mask recovery. Moreover, instances with the same anchor size share a common orientation map, reducing memory usage without sacrificing mask granularity. The framework constructs instance masks concurrently from the corresponding orientation maps using the surviving box predictions after non-maximum suppression (NMS), resulting in low complexity. The performance examples of this framework can be observed in

the figures below.

Figure 2.5: Accurately detecting the bounding boxes for the robots using OrienMask. The pixel contents of each bounding box are labeled using the segmentation framework, providing rich information on the 2D position of the target object in the image.

As can be observed from 2.5, the results of the instance segmentation framework can be relied on, specifically, the problem of identifying accurate object detection boundaries is addressed by selecting this approach and the information is ready for the next steps of the algorithm.

2.4 Disparity maps

This section addresses the challenge of depth estimation using a stereo camera sensor. This problem has been extensively explored in the literature for both monocular and stereo camera sensors, employing a variety of solutions. However, the common aspect among most of these methods is the initial estimation of "disparity" for pixels, followed by converting the disparity information into depth maps.

Estimating disparity is a fundamental problem that aims to determine the amount of displacement occurring to a pixel, corresponding to the same point in space when comparing the left and right images from a stereo camera sensor. Typically, this involves matching patches of pixels or features in the left and right image pairs, which is accomplished through techniques like pixel block-matching [71], graph cuts [72], and other optimization-based methods. While

these methods take advantage of stereo geometry and image characteristics to estimate the disparity maps, they can face significant challenges in dealing with occluded, texture-less, and noisy image pairs.

An alternative approach to traditional block-matching methods is to leverage the power of artificial neural networks, which offers several compelling advantages for this application. Neural networks excel at learning spatial and geometric features that establish links to depth information, enabling them to understand complex correspondences between pixel pairs in stereo images. When trained on extensive datasets, neural networks can handle ambiguities and ill-posed problems caused by noisy images in stereo matching. Their ability to model contextual information and incorporate local cues for depth understanding, along with their adaptability and flexibility for end-to-end depth estimation, make them an ideal choice for generating disparity maps in our application.

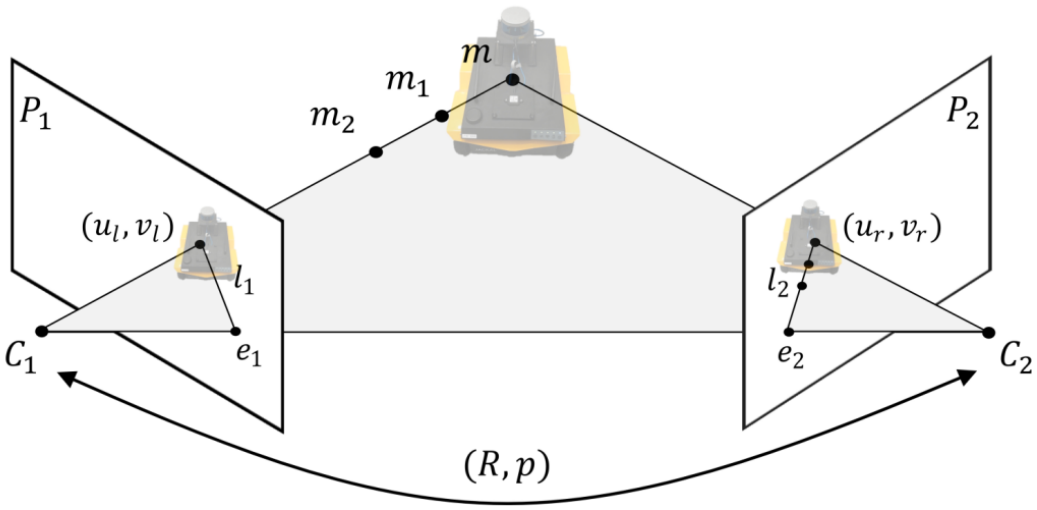

Figure 2.6: Depth estimation epipolar constraints [31].

The ultimate goal of employing a disparity estimation module in our system is to estimate the 3D position of points located on the body of the target object. This is realized through the 'epipolar' constraint, which ensures that the vectors going through the center of the cameras shown as C_1m and C_2m and the vector connecting the two cameras C_1C_2 shown in picture 2.6 lies on

the same line. One of the real-time open-source and available neural networks for the task of depth estimation is called "CoEx" [73] standing for "Correlate and Excite". In this work, the authors present a new method to address the challenges due to complex and computationally expensive spatially varying operations and introduce a concept called cost volume excitation. Additionally, they propose using top-k selection before the final disparity estimation for further enhancement of the accuracy. This algorithm was tested both on the training dataset and in our case and the generated disparity information is presented in the figures below. The areas with brighter colors represent higher disparities (closer objects) and areas with darker colors represent lower disparities (farther objects).

Figure 2.7: Estimating disparity on the real setup. (A) shows the training image and (B) demonstrates disparity inference.

As can be seen from figure2.7, a discrepancy in the performance of the algorithm can be noticed going from the training image to the test image. It is worth noting that the neural network was trained on the "Flying Things" dataset, a synthetic set of images with automatically generated labels used for depth estimation and optical flow. Considering the performance of the algorithm seen in figure 2.7, the flaws in estimating depth can be noticed at different points of the image. Specifically, one can see the top left corner of the disparity image and find the bright spot which is in fact supposed to be dark. For this specific reason, we have to limit our analysis only to the target object and enhance the estimate of the depth by filtering out parts of the disparity

information that are incorrect in reality.

2.5 Disparity extraction and 3D projection

In this section, the focus is on extracting the depth information for the target object to localize using the information from the last two steps. Specifically, it will be shown how we the information from the generated 2D mask by instance segmentation described in section 2.3 and the depth estimation method that was illustrated in section can be uesd2.4.

The first step in this module is to extract part of the disparity information that corresponds to the target object from the disparity image. This is done by generating a binary mask of the target object by the instance segmentation network and applying this mask to the disparity image. Figure 2.8 illustrates this process in a clear manner.

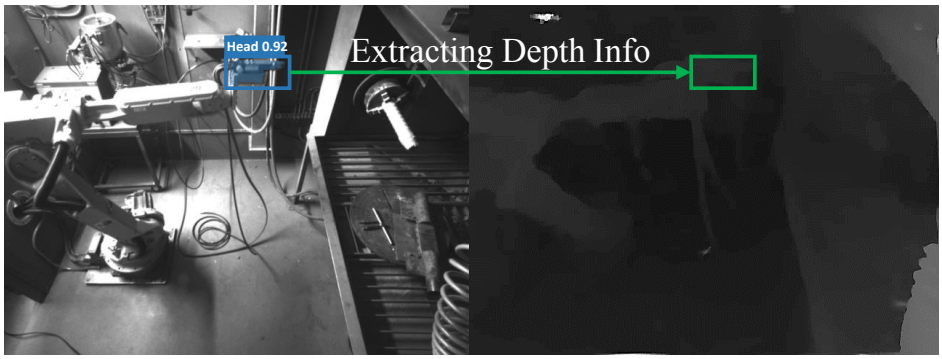

Figure 2.8: Extracting disparity info from the disparity image. This is done by applying the binary mask from the instance segmentation module to the depth image.

After extracting the disparity information, the next step involves projecting each pixel in the disparity image to a three-dimensional point cloud format. This process can be accomplished using the following set of equations. First, a projection matrix Q is defined, which can be derived from the intrinsic parameters of the stereo camera.

The extracted disparity information can then be converted into a 3D point in space with respect to the camera frame, enabling point cloud processing.

The equations required for this step are as follows.

$$Z = \frac{f.B}{d} \tag{2.1a}$$

$$X = \frac{x_c.Z}{f} \tag{2.1b}$$

$$Y = \frac{y_c.Z}{f} \tag{2.1c}$$

Where $(X, Y, Z)^T$ is a tuple of numbers representing each point in 3D space, B and f are the baseline and focal length of the camera (in pixels) as described in previous sections, and $(x_c, y_c)^T$ are the pixel-level coordinates of each projected point. Despite the fact that the above equations provide a way of projecting the feature-matched pixels to 3D, for real-time implementation, these equations mush be converted to vectorized form to allow fast and efficient point cloud projection. Such calculations are made possible by introducing a projection matrix called Q, which is used to encompass equations 2.1a 2.1b 2.1c. This matrix is defined as:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/B & \frac{C_x - C'_x}{B} \end{bmatrix} \tag{2.2}$$

Upon the introduction of this matrix, one can project a set of 2D points $P_s = \{(x_c^i, y_c^i)^T | 0 < x_c^i, y_c^i < W, H\}$ using the following equations given that W, H are the width and height of the image frame in pixels.

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} \tag{2.3}$$

Which facilitates fast computation and allows for finding the homogeneous coordinates of each 3D point using the intrinsic parameters of the camera.

2.6 Point cloud filtering and outlying point rejection

In this step, the filtering process for improving the point cloud reliability and eliminating the outliers in the point cloud is described. In this method, a statistical outlier removal filter is relied on for eliminating outlying points that are a result of errors in disparity calculation. It is beneficial to note that such points usually exist on the edges of objects, badly illuminated areas in the image, and areas lacking texture.

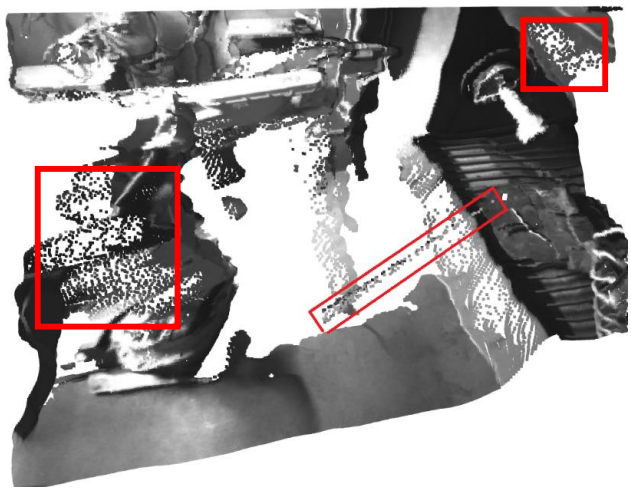

Figure 2.9: Error in disparity calculation in a glance. The areas in the picture that are marked with a red box are points in the point cloud that lack the required accuracy for reliable pose estimation. We strive to omit these points from further calculations.

As can be seen from 2.9, some of the resulting points in the point cloud seem to be a result of inaccurate disparity calculation, especially on the edges of objects. The filtering process in our methodology happens in 2 steps. The first step is done in the two-dimensional binary mask image generated by the instance segmentation network. After receiving each mask, the erosion morphological operation is performed on the binary image. This operator computes a local minimum over the area of the given kernel and by moving the kernel throughout the binary mask image, we replace the minimum value of the kernel with all of the pixels in the area. The second step of the filtration

process happens after projecting the disparity map to the 3-dimensional point cloud. The key to finding the outlier points is to exclude the ones that are far from their neighbors given a threshold that can be manually tuned. The mathematical expression of the filter is:

$$P_k = \{p_{i,k}\} \quad i \in 1, ..., m \ , N_{i,k} = \{n_{j,k} \mid norm(n_{j,k} - p_{i,k}) \leq t\} \tag{2.4}$$

Where P_k is the set of the unfiltered point cloud in the k'th frame, N_{i_k} is the i'th point in the filtered point cloud. Furthermore, t denotes the distance threshold based on which the filtering happens. If the size of points in the neighborhood of the point in question is less than the threshold, the point is marked as an outlier and is rejected from further analysis.

2.7 Sequential optimization-based pose estimation

Following our discussion on extracting and filtering the point cloud corresponding to the target object, we would like to estimate the pose of the target system. Specifically, having estimated a pointset P_k at time step k, our goal is to calculate the position and orientation of the robot in question. The following figure illustrates this fundamental problem.

Figure 2.10 illustrates the concept of pose estimation. At each time step, we would like to find the position of the body-fixed frame denoted by l with respect to the camera. Concretely, the vector $C = [X_c,\ Y_c,\ Z_c]^T$ is desired to be found as the vector that connects the origin of the body fixed frame to the origin of the world frame, defined in the world frame coordinates. Furthermore, the world frame itself is defined arbitrarily in a fixed position in the room.

The other, perhaps more important and challenging, part of pose estimation is estimating the orientation of the target object in 3D space. In particular, the rotation matrix $R \in SO(3)$ is calculated which defines the relative orientation of frame l with respect to frame w. Finally, the pose of the target object in the world frame will be defined as:

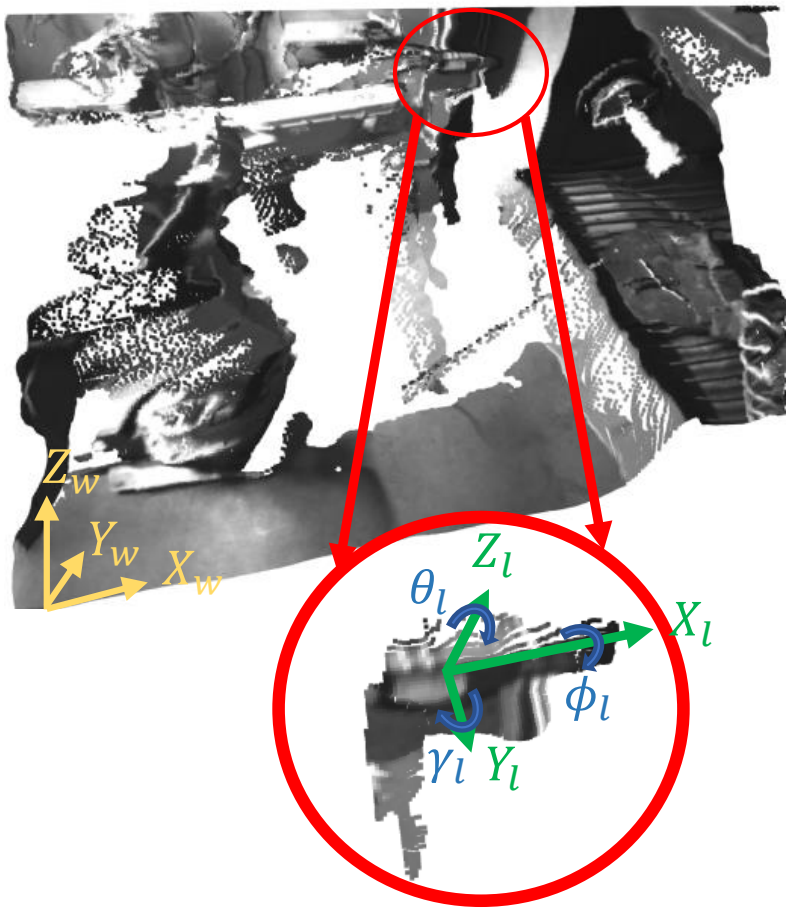

Figure 2.10: Estimating the pose of a target object, namely the material deposition torch, in 3D space using a visual sensor. The point cloud is extracted and the body-fixed frame as well as the world frame are defined.

$$T_w^l = \begin{bmatrix} R & C \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{2.5}$$

In this work, the pose estimation problem is defined as a nonlinear program (optimization cost function). Specifically, a cost function is defined such that is a measure of the closeness of the estimated bounding box with predefined dimensions to the filtered point set, attained from the last steps of the algorithm. This optimization program is defined as:

$$T_w^l = \underset{R,C}{\operatorname{argmin}} \sum_{i\,=\,1}^{m} \max_{1\leq j\leq 6}[|\{p_i\} - S_j|] \tag{2.6}$$

The optimization problem presented above entails the evaluation of the maximum Euclidean distance, with respect to each point within the point cloud, originating from any of the six surfaces associated with the proposed bounding box denoted as S_j. The bounding box, characterized by the R, C pair, is defined by these surfaces. The objective is to compute the aggregated sum of these distances across all points in the point cloud with respect to the bounding box's surfaces, and subsequently optimize the parameters in R and C to minimize this cumulative sum. In short, the aim is to ensure that the bounding box is positioned as closely as feasibly possible to the points it is intended to encompass.

The optimization program defined by 2.6 is a computationally expensive way of finding the bounding box that best fits the filtered point cloud. This is mainly due to the fact that in each frame, the optimization cost function requires the program to go through each point and compute the relative distance from the surfaces of the point cloud using the following equation, which finds the distance of a point $P = [X,\ Y,\ Z]$ from surface defined by the equation $ax + by + cz + d = 0$

$$h = \frac{aX + bY + cZ + d}{\sqrt{a^2 + b^2 + c^2}} \tag{2.7}$$

Our proposed method includes warm-starting the optimization problem, with an initial estimate of the pose that is derived by performing the **principal component analysis** (PCA). This method allows us to reduce the computational weight of our optimization problem significantly, allowing real-time computation of pose for the next steps.

The first step in finding an initial estimate of the pose (position and orientation) is to find the initial estimate of the position of the body-fixed coordinate

system with respect to the world coordinate system. Since the input information to our pose estimation framework is a stochastic set of points, scanned through depth estimation on the body of the target object, the position of the center of the body-fixed coordinate system is estimated by calculating its expected value. This is conveniently done by finding the physical centroid of the points, essentially averaging over the 3 dimensions that define the point set as the average is an unbiased estimator for the expected value in our case. The initial estimate of position C^O is found as:

$$C^O = \frac{\sum_{i=1}^{m} p^i}{m} \tag{2.8}$$

Where k represents the index of the point in the point set, and m is the total number of points in the point cloud.
In the next step, the points are normalized to have a similar scale in 3D dimensions to ease the process of covariance calculation. Specifically, each dimension of the points is fed through the following transformation:

$$\tilde{p}_k = \frac{p_k - C^O}{std(p_k)} \tag{2.9}$$

$\tilde{p}_i$ shows the transformed point and std is the standard deviation operation. In the next step, our aim is to find how each dimension of the point could vary in 3D space with respect to the mean. In order to do so, a 3×3 covariance matrix is defined such that has entries associated with pairs of point cloud dimensions (X, Y, Z). This matrix is of the form:

$$COV(\tilde{p}_k) = \begin{bmatrix} COV(X,X) & COV(X,Y) & COV(X,Z) \\ COV(Y,X) & COV(Y,Y) & COV(Y,Z) \\ COV(Z,X) & COV(Z,Y) & COV(Z,Z) \end{bmatrix} \tag{2.10}$$

The main diagonals of the covariance matrix defined by 2.10 are the variances of each dimension and the covariance matrix is symmetric.

As the next step, to find the principal components of the point cloud, the 3 eigenvalue-eigenvector Paris are found. Denoting these sets of values with

$V = [v_1, v_2, v_3]$ and $\Lambda = [\lambda_1, \lambda_2, \lambda_3]$, we sort the eigenvalues in ascending order and find the principal eigenvectors corresponding to each eigenvalue. This is done through the following equation.

$$\beta = \quad \Lambda \Longrightarrow V^* = V(\beta) \tag{2.11}$$

The sorted 3×3 matrix V^* serves as the initial estimate for the orientation of the target object. In this context, we can link the first column of this matrix with the largest dimension of the point cloud's bounding box, utilizing prior knowledge. Similarly, we associate the other columns accordingly. Consequently, the initial pose for the optimization problem can be obtained as follows:

$$T_{wO}^{l} = [V^* | C^O] \tag{2.12}$$

Where $V^* = R^O$ is the initial estimate of the orientation in 3D space. The following figure provides a visual of how the PCA-based initial estimate of the pose can help in the refinement of the pose through optimization.

Having defined our optimization cost function as well as an efficient way of deriving a warm-starting strategy for real-time computation, the last step of solving the optimization problem to find T_w^l for each time frame is defining a reasonable bound for the search of optimal values.

The intuition behind this approach is that the optimal pose for time step k should reasonably lie within an interval close to the optimal solution for T_w^l in time step k-1. To achieve this, a hard constraint can be imposed on a vector containing the optimization parameters, ensuring that the optimized values are in proximity to the results obtained in the previous time step. The vector ζ represents the 3D features of the bounding box, encompassing its 3D position in space as well as its roll, pitch, and yaw values:

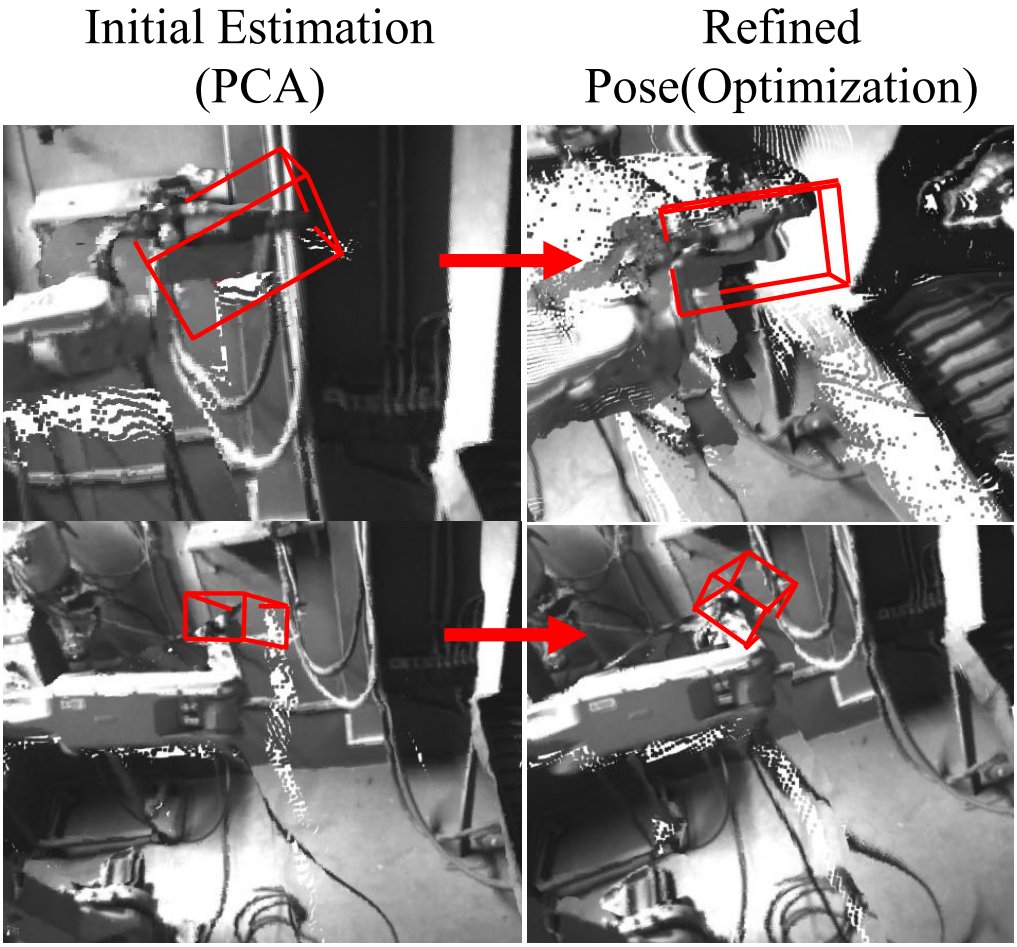

Figure 2.11: Using PCA and refining the pose through warm-started optimization. The example results show improvement in pose estimation using the PCA step.

$$\zeta_k^{6\times1} = \begin{bmatrix} C_k^{3\times1} \\ \phi_k \\ \theta_k \\ \gamma_k \end{bmatrix} \tag{2.13}$$

Then the hard constraint on the bounds for ζ would be defined as:

$$|\zeta_k - \zeta_{k-1}| \leq \alpha \tag{2.14}$$

Where α is a manually tuned hyperparameter of the optimization program.

As a reference for the reader, the Euler roll-pitch-yaw angles are defined in a Z-Y-X coordinate system and are derived using the following equations from the rotation matrix R^k for time step k where R_{ij} denotes the (i, j) element in the rotation matrix:

$$\phi = arctan(R_{21}/R_{11}) \tag{2.15}$$

$$\theta = arctan(\frac{-R_{31}}{\sqrt{{R_{32}}^2 + {R_{33}}^2}}) \tag{2.16}$$

$$\gamma = arctan(R_{32}/R_{33}) \tag{2.17}$$

2.8 Visual based state observer

In this section, the incorporation of physical understanding and kinematic constraints of the rigid body, whose pose is being estimated, will be elucidated. This integration aims to enhance the algorithm's performance in handling uncertainties. The approach offers several benefits, including but not limited to:

- **Estimation Error and Convergence**: The performance of the system heavily relies on the visual pose estimator module, as described in the previous section. In scenarios where the system is not visible to the camera due to occlusion or faces degraded visual conditions, such as dust and material splash during deposition, it is crucial that our algorithm continues to estimate the pose of the target object without failure. By leveraging our knowledge of the physical constraints of the system, we can create mathematical models that describe its motion in 3D space. This incorporation of information from such models significantly improves the algorithm's performance, leading to more accurate pose estimation of the target object. We can confidently estimate the position and orientation of the system visually in 3D space, considering the consistency of its motion.

  Moreover, onboard sensors within the robot's structure and design, such as wheel encoders that measure the robot's motion, provide essential in-

formation for the system. Through the use of a visual state observer, we introduce flexibility into the modeling process, developing mathematical representations of the sensor-level uncertainties. As a result, if new sensors are integrated into the system, their uncertainties can be modeled and seamlessly included in our algorithm. This adaptability ensures the continued accuracy of the pose estimation, even with the incorporation of new sensors.

- **Computational Cost**: In the upcoming sections, a physical modeling approach is proposed, utilizing a linear Kalman filter. The Kalman filter can recursively incorporate new measurements, enabling continuous state updates, making it suitable for real-time and efficient implementation. Additionally, the core principles of the Kalman filter are fairly straightforward and easy to implement in practical applications, contributing to the algorithm's efficiency.

  By leveraging the Kalman filter, the algorithm's development can be notably enhanced, and its efficiency in terms of real-time performance increased. The filter's capability to handle continuous updates ensures accurate pose estimation and responsiveness to changes in system dynamics and sensor measurements. As a result, the algorithm meets the real-time requirements of the autonomous material disposition system.

The next few sections will discuss how the kinematic constraints, sensor models, and recursive fusion can be taken advantage of in order to achieve a consistent estimate of the states of the system. The basic structure of a Kalman filter is illustrated in the figure below. At first, the estimation process starts with an initial rough estimate of the states of the system. This rough estimate has high uncertainty, therefore the values on the main diagonal of the covariance matrix P_0 are large numerically. At each time step, the filter goes through two main steps, namely the extrapolation step and the update step. In the extrapolation step, the states and uncertainty of the system are predicted for the next time step using a simple physical model of the system.

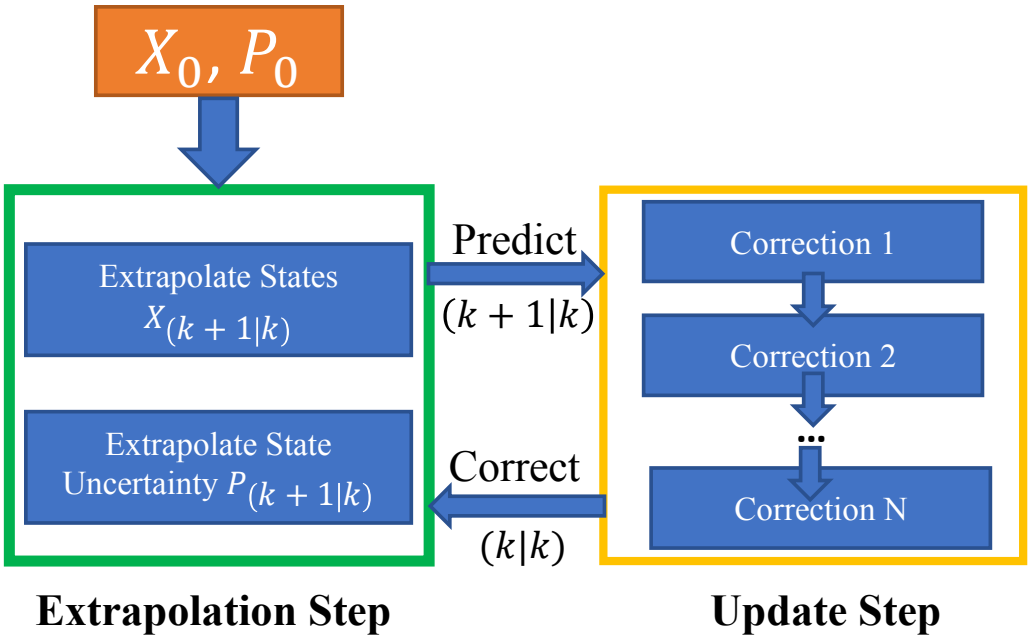

Figure 2.12: The linear Kalman filter framework.

In the correction step, noisy measurements from N sensors (whether external or internal) are incorporated to enhance the accuracy of the prediction and narrow down the uncertainties associated with it. This process continues for the duration of the filter's operation and in the occasion that n sensors don't provide any information in the current time step, they are simply skipped in the update step. In the next sections, we will discuss our formulation of sensor uncertainties, states, and the equations implemented in order to estimate the states of our fixed and mobile material deposition systems.

2.8.1 The general constant acceleration motion model

As the first step of the Kalman filter algorithm, a general motion model of the system is introduced, enabling the prediction of the rigid body's movement through 3D space. The states serve as the primary variables characterizing the pose and speed of the target object. Considering that all objects in 3D space have 6 degrees of freedom, the state vector comprises 12 variables, with

6 for position and orientation and 6 for the corresponding speeds. The state vector $X^k_{12\times1}$ is defined as follows:

$$X_k^{12\times1} = \begin{bmatrix} \zeta_k^{6\times1} \\ \dot{\zeta}_k^{6\times1} \end{bmatrix} \tag{2.18}$$

Where ζ is defined as the bounding box pose vector, characterized by equation 2.13. The next step concerns the definition of the prediction model. Since the samples through time are taken at tightly spaced intervals, it is safe to assume that the forces exerted on the system are equal for that interval. Therefore, according to Newton's second law, acceleration is constant as well. This allows us to predict the i'th state of the system for time step k represented by $x^i_{k+1|k}$ can be predicted as:

$$x^i_{k+1|k} = \boldsymbol{f}(x^i_k, u_k) = \frac{1}{2}\ddot{x}^i_k\Delta t^2 + \dot{x}^i_k\Delta t + x^i_k \tag{2.19}$$

Where $x^i_{k+1|k}$ is the state's prediction for the next time step, made at the current time step.

The general linear motion model of our system requires a matrix implementation to introduce control inputs, namely **system's acceleration**. This implementation is indicated via the following equations.

$$X_{k+1|k} = AX_{k|k} + BU_k + w \tag{2.20a}$$

$$z_k = HX_{k|k} + v \tag{2.20b}$$

Where $w \sim \mathcal{N}(0,\ \sigma_w^2)$ and $v \sim \mathcal{N}(0,\ \sigma_v^2)$ represent the random normally distributed noise in state prediction. Matrices A and B state transition and input matrices and H represents the measurement matrix, which allows us to choose the first 6 elements of the state matrix as our measurements.

Due to the large size of A and B matrices, here they are illustrated in a compact form. Concretely, for the state transition matrix:

$$f = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \ , \ A = [a_{ij}] \tag{2.21a}$$

$$\{a_{ij} = f : i \ = \ j; a_{ij} = 0_{2\times 2} : i \ \neq \ j\} \mid i, j = 1, \ ..., \ 6 \tag{2.21b}$$

The compact definition above suggests that the 2 by 2 elements on the main diagonal of matrix a are equal to matrix "f" and all other elements in matrix A are zeros. Similarly for matrix B:

$$b = \begin{bmatrix} 1/2\Delta T^2 & 0 \\ 1 & 0 \end{bmatrix} \ , \ B = [b_{ij}] \tag{2.22a}$$

$$\{b_{ij} = b : i \ = \ j; b_{ij} = 0_{2\times 2} : i \ \neq \ j\} \mid i = 1, \ ..., \ 6, j = 1, \ ..., \ 3 \tag{2.22b}$$

Finally, the inputs to the system are defined as accelerations in positional and orientational states, denoted as $U = \ddot{\zeta}^k$. Gaussian distributed noises for w and v are assumed to have bounded values, with their covariances being uncorrelated and represented by Q^k and R^k for time step k. The observation matrix H is an identity matrix (i.e., $H = I_{12\times 12}$) since our measurements are directly equal to our states..

2.8.2 Observerability analysis and observer design

With the aim of verifying that the system is observable for designing a bounded certainty state observer, one must check if the observability condition is satisfied for the system for which the observer is designed. The observability criterion below is a sufficient condition for the implementation of the Kalman filter.

$$O = [H, \ HA, \ HA^2, \ ..., \ HA^n] \longrightarrow rank(O) = n \tag{2.23}$$

To verify this condition, the observability matrix O is constructed in a similar compact format as previously illustrated. Since the H matrix is an identity matrix in our case, the observability matrix can also be calculated from $O = [I, \ A, A^2, ..., \ A^{12}]$. By expanding this equation for the observability

matrix, it can be demonstrated that the system is indeed observable. Due to the unique shape and structure of the matrix A, a closed-form solution for A^k can be calculated in a compact form as:

$$f^* = \begin{bmatrix} 1 & k\Delta T \\ 0 & 1 \end{bmatrix} \ , \ A^k = [a^*_{ij}] \tag{2.24a}$$

$$\{a^*_{ij} = f^* : i \ = \ j; a^*_{ij} = 0_{2\times 2} : i \ \neq \ j\} \mid i, j \ = \ 1, \ ..., \ 6 \tag{2.24b}$$

It is beneficial to note that ΔT represents the sampling time in equations 2.21a through 2.24a.

To update the estimation of acceleration, a learning-based input estimation network is designed. The recurrent long short-term memory network shown in figure 2.13 is trained to estimate the input to the system in the next time step, using the short- and long-term memories ξ_k and δ_k of the previous measurements Z_k. Furthermore, this network performs corrections on current measurements Z_k with the objective of improving depth estimation accuracy and achieving consistent estimation for distant objects. This correction is made to yield $\tilde{Z}_k$. In every instance of using deep-learning-based methods, one of the most critical challenges is to attain the **ground truth** for the target variable to estimate. This is due to the fact that deep-learning methods, at heart, rely on estimating a black-box object from observations made in the real world, and acquiring the clean, usable data corresponding to these observations is the most challenging segment of estimation methods that rely on machine learning.

For estimating the input acceleration to the state observer, an accurate estimation of the acceleration of the system using an independent method is required, serving as the ground truth for the supervised learning algorithm. In the case of the fixed manipulator use case, the accurate end-effector pose derived by the encoder sensors of the robot can be used in the forward kinematic equations of the robot. Using the following equation, which characterizes the forward kinematics of the fixed manipulator, we can find the pose of the final link attached to the material deposition gun.

$$T_e^o = \prod_{i=1}^{6} e^{\varrho_i \theta_i} T_o \tag{2.25}$$

Where T_e^o and $T_o \in \mathrm{SE}(3)$ are the initial link and end-effector link poses of the robot, ϱ_i is the twist vector of the i'th joint and θ_i is the i'th joint angle.

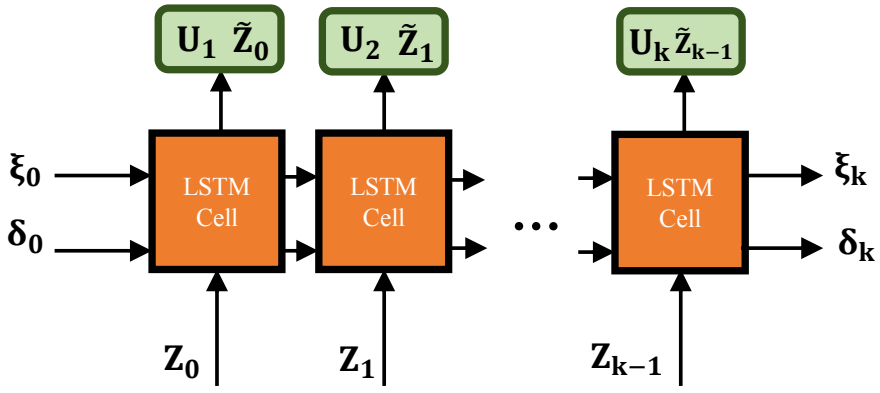

Figure 2.13: Input estimation and measurement correction LSTM. Inputs to the system are learned via a long-short term memory network, allowing for learning time-dependent variables such as inputs.

In the case of the mobile material deposition platform, the ground truth estimation job is slightly more convolved. In this case, we take advantage of 4 unique markers incorporated on the robot's top surface which are used in identifying the corners of the robot. Using a bird-eye-view (BEV) frame generated from RGB images, a color threshold is used to identify the markers in the BEV image. After localizing the corners of the robot, the center is found by calculating the centroid of the markers, and orientation is found by deriving the slope of the line fit to the 2D location of the markers. The following figure demonstrates this method for estimating the ground truth for the case of the mobile robot.

Leveraging the information by the bird-eye view generated by the last step (figure 2.14), one can determine the ground truth position and orientation of the mobile robot. This is done by using the three simple equations below:

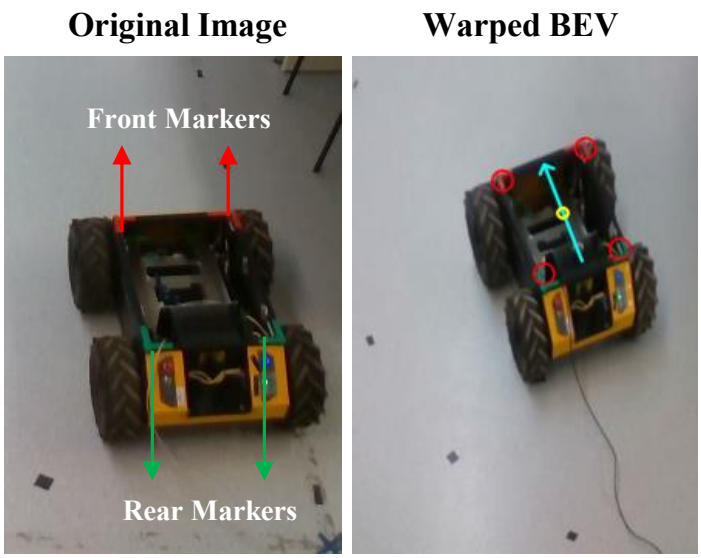

Figure 2.14: Estimating the ground-truth position for the mobile robot. The corners are identified using a fixed color threshold to extract pixels that have green and red color tones which correspond to either end of the robot.

$$\boldsymbol{X_{gt}} = \frac{X_f + X_b}{2} \tag{2.26a}$$

$$\boldsymbol{Y_{gt}} = \frac{Y_f + Y_b}{2} \tag{2.26b}$$

$$\boldsymbol{\gamma} = \text{atan2}(Y_f - Y_b, X_f - X_b) \tag{2.26c}$$

Where the subscripts gt, f, and b stand for ground truth, front and back respectively.

So far in the discussion of the recursive linear Kalman filter, employing the input estimator network, definition of states, and the linear model defined by 2.18 through 2.26a to extrapolate the current state and make a prediction regarding the expected value of the states for the next time step is discussed. In addition to predicting this mean value, we must also make a prediction about the uncertainty of the states in the next time step. Specifically, if our current best estimate of the covariance matrix $P_{k|k}$ representing the uncertainty of states in the current state is available, we are striving to make a prediction on this covariance matrix by finding $P_{k+1|k}$. The following equation is implemented to yield an estimate of this uncertainty matrix.

$$P_{k+1|k} = AP_{k|k}A^T + Q_k \tag{2.27}$$

Where $P_{k+1|k}$ represents the extrapolated uncertainty matrix of the states, with dimensions 12×12, A denotes the state transition matrix as defined by 2.21a, and Q signifies the motion model's uncertainty matrix. By examining 2.27, it is evident that relying solely on the motion model leads to a continuous increase in the uncertainty of the states. This is attributed to the presence of Q, which is added to the state covariance term each time the motion model is utilized. To obtain an estimate of the uncertainty matrix of the motion model Q, the following equation is employed.

$$Q_k = \boldsymbol{E}(w.w^T) \tag{2.28}$$

Where E represents the expected value (mean) of the argument given. In The next section, we will dive into calculating the sensory noise covariance matrix R.

2.8.3 Sensory noise formulation

The crux of the linear Kalman filter entails the ability to model the sensors in the system and their noise based on statistical methods, domain knowledge, or other estimation techniques. To correct for the uncertainty accumulated by the motion model, a measure of the certainty of our real-world measurements is formulated. Specifically, a mathematical representation of the accuracy of the sensors employed in the system is proposed, considering their known weaknesses and strengths.

The first sensor under consideration is the camera sensor, renowned for its rich and dense information and visual measurements. As described in the previous sections, the estimation of the states of the target robotic system relies on the camera sensor by incorporating a bounding box and estimating the associated 6 degrees of freedom. However, this sensor has a significant downside – the reliability of the measurements rapidly decreases as the robot's relative

distance from the camera increases. Additionally, whenever the variance of the measurements increases, the trust in the camera sensor must be decreased. In other words, if recent measurements significantly differ, the reliance on the camera sensor must be reduced, and the estimates of the states should be obtained from other sensory or model sources. The proposed equation for adjusting the camera covariance matrix is as follows:

$$R_k^c = I_{n\times n}(\alpha + \beta\sqrt{|\tilde{z}_{k+1} - \tilde{z}_k|} + \frac{\gamma}{1 + e^{-(|\tilde{z}_k - z_t|)}}) \tag{2.29}$$

The equation above presents the adjustment process for the camera sensor's covariance matrix, denoted as R_k^c. In this equation, α, β, and γ are three manually tuned coefficients, z_k represents the measurement vector, and z_t is a depth threshold representing the maximum distance from the camera beyond which the depth measurement module's reliability diminishes.

The first term, α, accounts for the inherent noise present in our visual-based pose estimator. The second term captures the variance in the measurements, which increases with greater normal distances between two measurements, potentially leading to reduced accuracy. Consequently, the main diagonal values of the covariance matrix are increased. The third term incorporates the depth factor, which is governed by the depth threshold z_t. By tuning this threshold, we impose a limit on the maximum depth value for which the camera sensor remains accurate. Beyond this threshold, the measurement noise in this context increases, and there is an upper limit γ that represents this increment when the target robot is significantly beyond the depth threshold z_t.

The next step involves leveraging the information provided by wheel encoders to enhance robot localization accuracy. Wheel encoders allow for incremental measurement of wheel rotation and speed, which can be converted into vehicle speed in body and world coordinates. However, this conversion necessitates a comprehensive understanding of the robot's locomotion mechanisms. For skid-steering mobile robots like the Husky Clearpath, a slip-aware model is developed to convert left and right wheel velocities to vehicle speeds using a skid-steering motion mechanism.

Skid-steering robots, such as the Husky Clearpath, possess versatile control over the wheels, enabling differential speed and direction control. This mechanism allows the robot to execute tight turns and turns without moving by adjusting wheel speeds or driving them in opposite directions. As a result, the robot exhibits agile navigation in confined spaces and reduced system complexity due to its straightforward nature, enabling easy traversal of its environment. The overall view of this locomotion mechanism is depicted in the following figure:

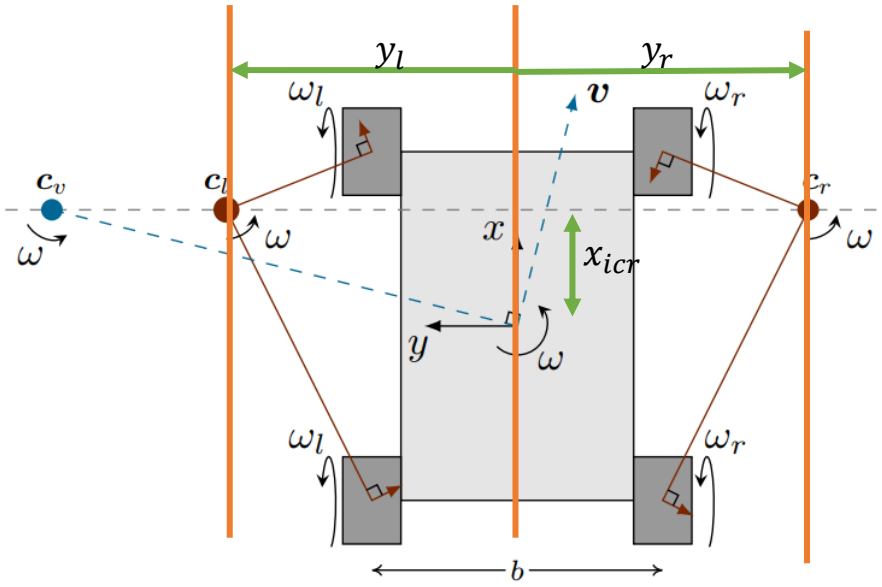

Figure 2.15: The skid-steering locomotion mechanism. In this part, we are striving to model the mobile robot's motion by introducing slip in the right and left wheel pairs.

Figure 2.15 provides a visualization of various parameters involved in modeling the locomotion of a skid-steering mobile robot. The parameters c_v, c_r, c_l represent the instantaneous centers of rotation (ICR) for the vehicle, left wheel, and right wheel, respectively. Additionally, y_l and y_r depict the horizontal distances between the vehicle's symmetry axis and the ICRs, while x_{icr} represents the vertical distance between the vehicle's horizontal axis of symmetry and the line on which all three ICRs are situated. The body coordinate system of the

robot is centered at its midpoint, with $\mathbf{x}$ pointing forward and $\mathbf{y}$ pointing to the left. The primary objective is to convert the wheel rotational velocities $\omega_{l,r}$ into vehicle velocities in the body coordinate system, denoted as $\mathbf{v_x}, \mathbf{v_y}, \omega$.

To achieve this conversion, a slip-aware motion model is utilized, which can be expressed by the following equation:

$$\begin{bmatrix} \mathbf{v_x} \\ \mathbf{v_y} \\ \boldsymbol{\omega} \end{bmatrix} = \mathcal{G}(\Lambda) \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \tag{2.30}$$

In which $\mathcal{G}$ encodes the linear, slip-aware motion model in which the parameters inside figure 2.15 are used. Such parameters are stored in Λ and will be described shortly. This entity is found using the following equation.

$$\mathcal{G}(\Lambda) = \frac{1}{y_l - y_r} \begin{bmatrix} -y_r\alpha_l & y_l\alpha_r \\ -x_{icr}\alpha_l & x_{icr}\alpha_r \\ -\alpha_l & \alpha_r \end{bmatrix} \tag{2.31}$$

The slip parameters $\alpha_{r,l}$ encapsulate various factors, including wheel friction, terrain characteristics, wheel pressure level, and other unknown parameters related to the wheels. To determine the slip parameter vector Λ, we propose a method that involves minimizing the normal distance between the ground truth obtained through the BEV method described earlier and the results derived solely from wheel odometry using encoder sensors. This optimization process seeks to find the optimal parameter vector Λ^* by minimizing the following cost function:

$$\mathcal{J}(\boldsymbol{\Lambda}) = \frac{1}{T} \sum_{i=1}^{T} |\zeta_i^* - \hat{\zeta}_i| \longrightarrow \Lambda^* = \underset{\Lambda}{\operatorname{argmin}} \quad \mathcal{J}(\Lambda) \tag{2.32}$$

Where ζ_i^* represents the ground truth state vector, $\hat{\zeta}$ the state vector obtained by wheel odometry, and T is the total number of gathered data points to optimize $\Lambda = [x_{icr}, y_l, y_r, \alpha_l, \alpha_r]$.

After finding the optimized Λ^*, equation 3.4 can be used to convert wheel velocities to body frame velocity vector $V_{B_k} = [v_x, v_y, \omega]$. The last step of this pipeline is to use this body velocity vector to obtain the velocity in the world coordinate system $\{w\}$. This is done so by first constructing the body velocity

twist ξ_b and then performing the matrix exponential operation to find vehicle pose in the world coordinate system $T^w_{B_k}$.

$$\xi_b = \begin{bmatrix} 0 \\ 0 \\ \omega_r \\ v_x \\ v_y \\ 0 \end{bmatrix} \Longrightarrow T^{B_{k+1}}_{B_k} = \mathbf{exp}(\xi_b) \tag{2.33}$$

This leads us to find $T^w_{B_k}$ by simply multiplying the odometry pose transformation $T^{B_{k+1}}_{B_k}$ by the global pose of the robot at time step k. Finally, the encoder sensor's uncertainty covariance matrix is introduced, whose diagonals increase if the distance traveled between the two consecutive time steps is large. This is a simple representation of the fact that wheel odometry-based localization suffers a loss of accuracy by the accumulation of errors.

$$\mathbf{R^e_k} = I_{n\times n}\{\eta_1 \; tanh(\eta_2 \; \Delta D)\} \tag{2.34}$$

Where $\mathbf{R^e_k}$ is the encoder uncertainty matrix, $\eta_{\{.\}}$ is a tunable parameter representing the properties of the covariance matrix, and ΔD is the distance traveled since the last time step. In the next section, incorporating the measurement noise covariance matrices $\mathbf{R^c}$ and $\mathbf{R^e}$ is discussed to update the predictions from the general motion model with corrections made in the linear state observer.

2.8.4 Uncertainty-aware sensor fusion

In this section, the mathematical tools to incorporate the main sources of information for the visual sensory fusion framework are discussed. As previously explained, the Kalman filter methodology involves two main steps in each time step. The first step is the prediction step, where the system's mean state and covariance matrices are forecasted for the next step using the motion model detailed in previous chapters. The next step is the update step, where the physical motion model's prediction is corrected using measurements from on and off-board sensors, including the camera and wheel encoders. This update

occurs whenever a new measurement from the environment is available. By incorporating information from the real world, the accuracy and consistency of our physical understanding are enhanced. However, it should be noted that this information is uncertain and comes with its own sources of noise and uncertainty. Hence, the Kalman gain is employed to facilitate a smooth "shift of trust" between our sources of information (i.e., motion model, camera, encoder) in case our reliance on each of the sensors is compromised. The first and foremost equation used to accomplish this step is the computation of the Kalman gain for the sensor "i" in time step k, for which a measurement is available.

$$\mathcal{L}_k^i = P_{k|k-1}\ H^T(HP_{k|k-1}H^T + R_k^i)^{-1} \tag{2.35}$$

Where $\mathcal{L}_k^i$ is the Kalman gain for the i'th sensor in time step k and the rest of the parameters are already defined. There are 3 notable points regarding this equation. First, the Kalman gain for sensor i is directly dependent on the sensory covariance matrix for the same sensor, which allows for the Kalman gain to be adjusted accordingly. Second, it could be easily shown that if R_k^i is zero, the Kalman gain will be 1, and if R_k^i is a large number then, $\mathcal{L}_k^i$ will be close to 0. Finally, computing the Kalman gain is a computationally expensive process as one has to compute the inverse of a large matrix. However, with the advent of high-end processors, this issue can be resolved as well.
The next step would be to correct state measurements to obtain $X_{k|k}$. This process is the heart of the recursive Kalman filter and is performed via the following equation:

$$X_{k|k} = X_{k|k-1} + \mathcal{L}_k^i(z_k - HX_{k|k-1}) \tag{2.36}$$

The term multiplied by the Kalman filter is called *innovation*. The reason for this name is that it is due to this term that the discrepancy between the measurement and previous prediction of state is measured and based on the level of trust in the innovation (i.e. the magnitude of $\mathcal{L}$), this change is applied to the prediction $X_{k|k-1}$ to obtain $X_{k|k}$. This explains how the Kalman gain

governs the update in the prediction. If the Kalman gain is 0, there is no change of state in the update step. However, if the Kalman gain is 1, the previous prediction of the states is simply *forgotten* and $X_{k|k}$ will be directly equal to the measurement z_k. The final step would be to correct the covariance matrix, predicted in the last step. This is done through the following equation.

$$P_{k|k} = (I - \mathcal{L}_k^i H) P_{k|k-1} (I - \mathcal{L}_k^i H)^T + \mathcal{L}_k^i R_k^i \mathcal{L}_k^{i\,T} \tag{2.37}$$

Moving to the next step, the Kalman filtering process is repeated. This involves obtaining the prediction of state and covariance matrices, calculating the Kalman gain, and finally, updating the states and uncertainty matrices. This iterative process ensures that the estimation continuously incorporates new measurements and refines the state estimates as new data becomes available. By doing so, the Kalman filter adapts to changes in the environment and the reliability of different sensors, providing a more accurate and consistent estimation of the system's state over time.

2.9 Summary

In this chapter, a visual-based physically informed state estimation framework was proposed. This method addresses the problem of degraded visual conditions in the context of thermal spray, a problem that is prevalent in state estimation frameworks in which the camera is not fixed in the environment and also it highly reduces the sensory setup cost for mobile and fixed material deposition systems.

In our framework, depth estimation with object instance segmentation is incorporated to generate a dense 3D representation of the object using a stereo camera sensor. The quality of this representation is improved by performing 3D point cloud filtering. By utilizing prior knowledge of the dimensions of the target object, an optimal bounding box is estimated based on the normal distance of the points from the surfaces of the bounding box. Furthermore, physical kinematic constraints of the target object are taken into account in the optimization process to minimize pose estimation errors caused by inaccu-

racies in the depth estimation step. In addition, a motion model of the system is developed based on the piecewise constant acceleration assumption. Uncertainty in the motion model and the visual estimation is characterized through empirical expressions, resulting in an algorithm that is more tolerant of errors in the face of visual estimation errors. In the next chapter, the methodology for using the currently available estimated state of the robot to calculate the control inputs is discussed, which is needed to drive the mobile robot on a predesigned trajectory.

Chapter 3

Controller design

In the previous chapter, the methodologies for designing a visual state observer were explained, enabling the estimation of the position, velocity, and acceleration of the mobile robot within its operating environment. However, after obtaining an accurate estimate of the robot's state, the next challenge is how to control it along a desired path to enable visual navigation. This is crucial for mobile robots, as it allows them to operate independently and autonomously in the target environments.

The controller determines the robot's behavior and motion to respond to changing conditions and accomplish its tasks. In this chapter, we will delve into the design of the mobile robot's controller, addressing the potential challenges and complexities involved in developing a well-designed controller to enable autonomous robot operation.

It is imperative to understand that the control module is intrinsically susceptible to diverse sources of noise. Initially, the estimated state of the robot may deviate from absolute accuracy. Moreover, the predictive law might not completely capture the forthcoming states and behaviors of the robot. Lastly, the execution of actuation commands on the robot's wheels may not be perfectly precise. Through the establishment of a comprehensive cost function pertaining to desired path tracking within the controller framework, a certain level of noise mitigation is attainable. However, for more severe disturbances, we suggest performing extensive robustness analysis in future endeavors.

One of the primary challenges in autonomous robotic navigation is dealing

with uncertainties in the system's model and states. A carefully designed controller ensures the robot operates safely and efficiently. This chapter builds on the previous discussions on state estimation using off-board cameras and focuses on the mathematical aspects and formulations of two types of controllers. It also considers the various factors and considerations involved in this crucial aspect of our mobile robotic system.

3.1 The feedback control structure

This section addresses the problem of feedback control and outlines the design and implementation of the controller, plant, and observer to enable autonomous operation of our system. The general outlook of our feedback control structure is depicted in Figure 3.1.

The first and foremost part which is worth mentioning here is the controller. The controller is the main component responsible for calculating and generating the control signals required to operate the system and keep it on the designed trajectory. It utilizes information such as the states of the system, previous control inputs, and kinematic constraints of the mobile robotic system and applies the appropriate control command based on previously implemented algorithms to regulate the robot's motion. The controller incorporates an objective function to minimize the error between the actual (observed) and desired outputs of the system to ensure precise control. The general control law regardless of the controller structure is defined by the following abstract equation.

$$U_k^* = \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{X}}_{k+N,k+N-1,\ ...,k|k}, \boldsymbol{U}_{k+N,k+N-1,\ ...,k-1|k},\ \boldsymbol{Y_k},\ ...) \tag{3.1}$$

Equation 3.1 defines the structure of the control law that will be implemented to govern the movement of the mobile robot along a desired trajectory. The general control law $\mathcal{F}(.)$ encompasses the function that relates the predicted observed states, inputs, and outputs of the system over a future time horizon N, thereby determining the optimal input U_k^* at the current time step

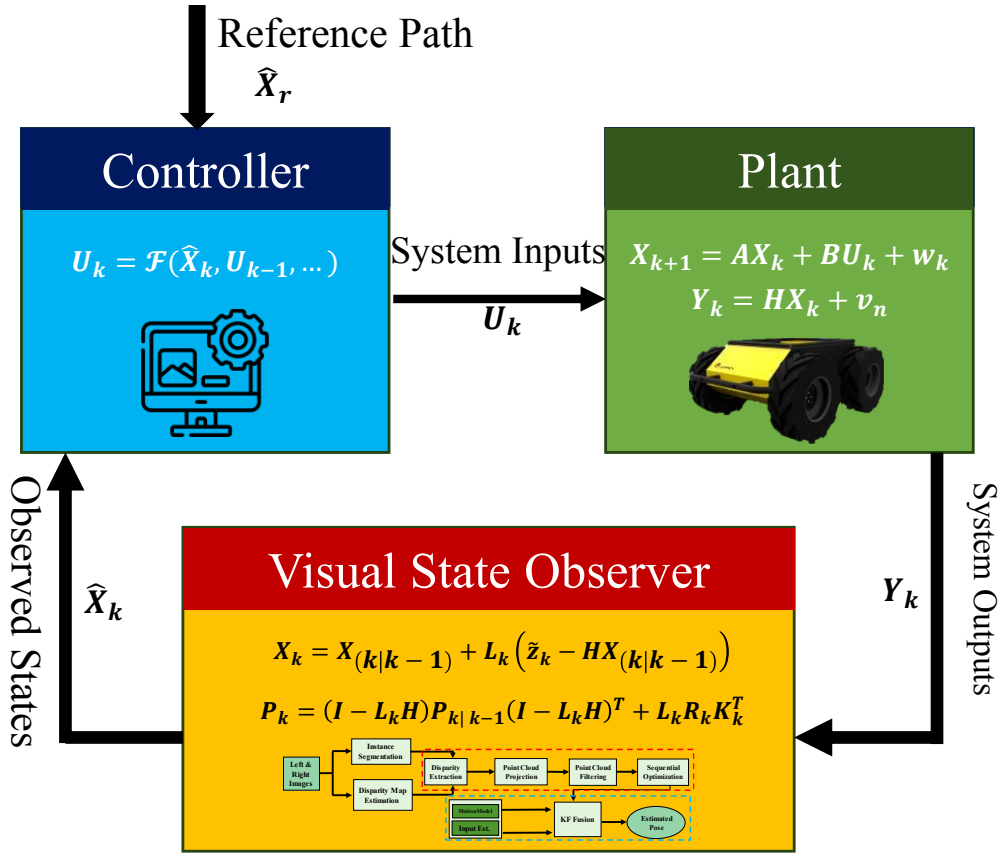

Figure 3.1: The general feedback control structure, involving the controller, plant, and the visual state observer.

k. This function may also consider the time derivatives and integrals of the mentioned components to minimize accumulated error and steady-state error. In the following sections, we will explore two potential approaches for designing these functions.

The next crucial element of the overall feedback control structure is the plant, which refers to the physical system of the mobile robot, including its kinematic constraints, mechanical structure, and actuators. It captures the dynamics and behavior of the robot, which may vary based on the control inputs provided to it. The physical representation of the plant, relating the control inputs (acceleration) and current states (position and orientation) to the states in the next time step, has been discussed in the previous chapter. This representation involves two linear equations that link the control inputs and current states to the states in the subsequent time step.

$$X_{k+1|k} = AX_{k|k} + BU_k + w \tag{3.2a}$$

$$Y_k = HX_{k|k} + v \tag{3.2b}$$

Where $w \sim \mathcal{N}(0,\ \sigma_w^2)$ and $v \sim \mathcal{N}(0,\ \sigma_v^2)$ represent the random normally distributed noise in state prediction. Matrices A and B state transition and input matrices and H represents the measurement matrix, which allows us to choose the first 6 elements of the state matrix as our outputs.

Finally comes the visual state observer. This component was the primary focus of the previous chapter and is an integral part of the feedback control structure. It estimates the unmeasured states of the mobile robot (such as velocities) through the available sensory data. This sensory data includes the visual measurements (camera images in the image frame) as well as the encoder inputs. The observer leverages the system's kinematic model as well as the constraints to provide an approximation of the current state by minimizing estimation errors through the Kalman filter framework. By incorporating the state observer, the control system can operate using an enhanced estimation of the states which allows for control accuracy.

The presence of each of the previously discussed components is essential. The controller generates the appropriate control signals to minimize the state error while the plant embodies the physical understanding of the system. Finally, the state observer enables estimation of the robot's states which in turn helps the control to make informed decisions based on observed and estimated states. Altogether, these components form a feedback control structure that is critical for achieving accurate control of the mobile robot.

3.2 Model description

Illustrated in the previous section, the plant consists of a mobile robot with a skid-steering locomotion mechanism. Skid-steering locomotion necessitates that the left and right wheel pairs are linked mechanically using a belt, restricting their respective speeds to be the same. The difference between the

wheel speeds allows the robot to rotate along a vertical axis going through its center.

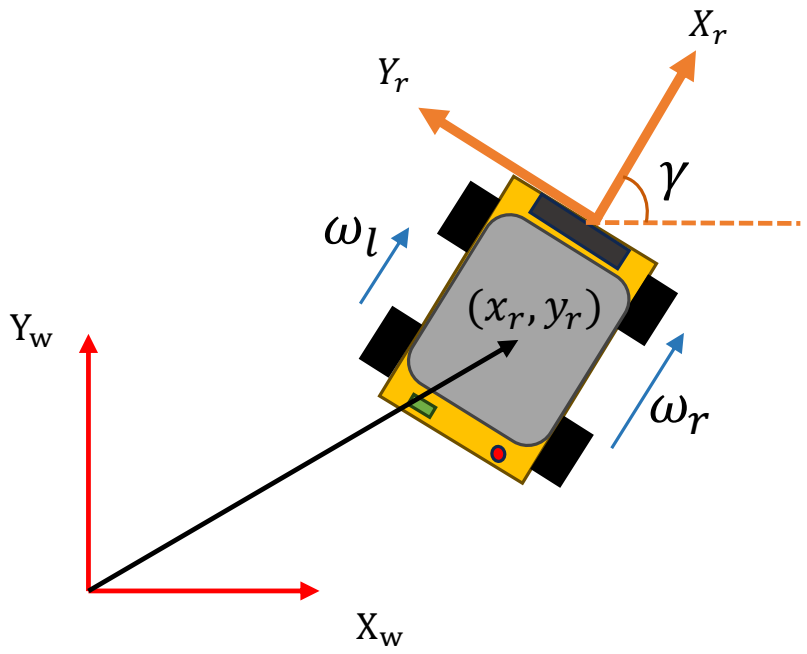

Figure 3.2: The skid-steering robot. Our plant's inputs and outputs are illustrated. Any given wheel velocity may cause a change in the position and orientation of the mobile robot in 3D space.

Understanding the concept of inputs and outputs to the system is a fundamental element of designing a control strategy that controls the robot autonomously on a given path. In the formulation used in this work, the inputs correspond to the velocity of the wheels and the outputs represent the position and orientation of the robot. In a skid-steering motion model, the robot's motion is primarily governed by adjusting the velocity of the wheels. The controller, by choosing appropriate values for ω_l and ω_r (the left and right rotational wheel velocities) can maneuver the robot to make rotations or move forward. Thus the robot's inputs will be:

$$U_k = \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \tag{3.3}$$

Please note that the input wheel rotational velocities ω_l and ω_r can readily be converted to the speed of the robot in the body frame B_r. This speed vector

is denoted as $\mathcal{V}^{B_r}$ and can be found using the following equation, previously explored in depth in the previous chapter.

$$\mathcal{V}^{B_r} = \begin{bmatrix} \mathbf{v_x} \\ \mathbf{v_y} \\ \boldsymbol{\omega} \end{bmatrix} = \mathcal{G} \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \tag{3.4}$$

Furthermore, please note that the robot's acceleration in the world frame can be found through the following equation, enabling us to predict the motion of the robot in the upcoming time steps.

$$\mathcal{A}_k^w = \frac{1}{\Delta T} \begin{bmatrix} Cos(\gamma) & -Sin(\gamma) \\ Sin(\gamma) & Cos(\gamma) \end{bmatrix} (\mathcal{V}_{k+1}^{B_r} - \mathcal{V}_k^{B_r}) \tag{3.5}$$

Where $\mathcal{A}_k^w$ is the acceleration of the robot in the world frame.
As previously elaborated, the outputs of the system Y_k are the position and orientation of the robot with respect to the world frame $W_{\{.\}}$ The relationship between the inputs and outputs can be represented through the kinematic equations which capture the nature of the movement of the robot, meaning that:

$$Y_k = \begin{bmatrix} \boldsymbol{x}_r \\ \boldsymbol{y}_r \\ \boldsymbol{\gamma} \end{bmatrix} \tag{3.6}$$

The states can, in turn, be related to the states of the system, through the linear state-output equations.

$$Y_k = HX_{k|k} + v \tag{3.7}$$

In the next section, the meaning of trajectory errors and their definition are discussed.

3.3 Trajectory (path) and error definition

This section concerns the problem of defining a reference path for our mobile robot to follow. In the context of mobile robot control, defining a path comprised of consecutive waypoints is essential, as it allows us to quantify the tracking errors for our controller which completes the feedback control loop.

Moreover, it can provide us with the necessary tools to evaluate the performance of our trajectory-tracking controller.

A trajectory fully defines the desired motion of the mobile robot by specifying the positions and orientations that must be achieved by the controlled system at different points in time and space. There are several choices to define the trajectory to be followed by the robot. The first possible choice is explicitly defining the mathematical equation of the line which must be followed by the robot. There is no necessary rule for this line to be straight, thus characterizing a general path using this method may involve the complex formulation of an abstract path which is not possible for all plausibly desired paths.

To address this issue, the desired trajectory is defined as tightly sampled points of the originally desired path, connected by straight lines. The following figure represents the defined trajectory.

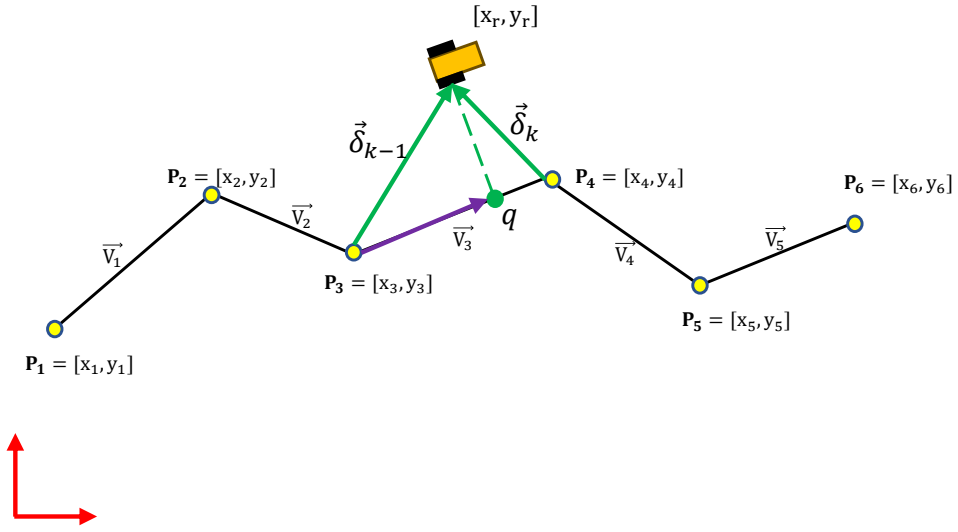

Figure 3.3: Defining the trajectory and error calculation process.

Mathematically, the trajectory is defined as the ordered point set $\mathbf{P}_r$ in the world frame, as well as the respective desired velocity profiles $\mathbf{V}_r$ (vectors)

connecting each consecutive point pair. Specifically:

$$\mathbf{P}_r = \begin{bmatrix} P_1 & P_2 & \cdots & P_n \end{bmatrix} \tag{3.8a}$$

$$\mathbf{V}_r = \begin{bmatrix} V_1 & V_2 & \cdots & V_{n-1} \end{bmatrix} \tag{3.8b}$$

After defining the trajectory profiles, our next goal is to compute the error vector E_k at time step k. This vector is comprised of two fundamental parts. The first part of the trajectory error vector encodes the minimum distance elements. The first part is the minimum distance between the robot and the trajectory and the second part is the relative orientation error of the robot with respect to the trajectory. These two error parts are denoted by e_k and ψ_k and are illustrated in the following figure.

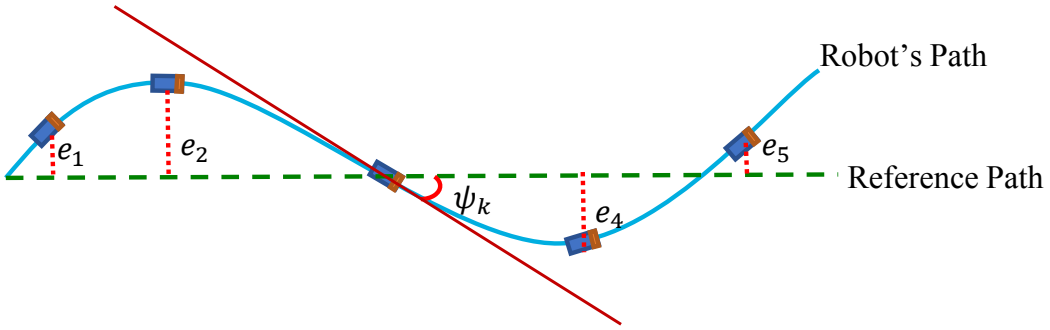

Figure 3.4: Defining the trajectory errors. The error vector E_k must be calculated by finding the minimum distance between the robot and the reference path. Also, relative orientation error should be calculated as well.

To define the trajectory tracking error, the two elements of tracking error e_k and ψ_k are aggregated in E_k:

$$E_k = \begin{bmatrix} e_k \\ \psi_k \end{bmatrix} \tag{3.9}$$

In order to calculate e_k namely the shortest distance between the robot and the desired trajectory, one must find the minimum distance of the robot with the line segments consisting of two consecutive way points. This vector is

denoted by $\overrightarrow{P_{k-1}P_k}$. This distance is found by projecting the vector $\overrightarrow{\delta_{k-1}}$ onto $\overrightarrow{P_{k-1}P_k}$, arriving at point q. There are two cases for the position of point q:

- **Case 1**: If the position of point q is between points P_{k-1} and P_k, then the minimum distance e_k is found by calculating the distance of the perpendicular projection element connecting point q and the robot. (Shown in dashed line in figure 3.3.
- **Case 2**: If point q is located after P_k (length of $\overrightarrow{P_{k-1}q}$ is less than length of $\overrightarrow{P_{k-1}P_k}$) then minimum distance e_k is found by calculating the distance of the robot with respect to point P_k.

Having computed e_k the next step of computing the error vector is to find ψ_k, namely the error in the orientation of the mobile robot. This is found through the simple equation below:

$$\psi_k = \gamma_k \ominus \gamma_k^r \tag{3.10}$$

Where $\ominus$ operator is defined as the difference between two angles that lies in the $[-\pi,\ \pi)$ interval and γ^r is the reference orientation angle found by calculating the slope of the line on which the point q lies. In the next section, the definitions given so far to derive a proportional-integral-derivative (PID) controller are used for the trajectory tracking of our mobile robot.

3.4 Proportional-integral-derivative controller

A Proportional-Integral-Derivative controller widely known as "PID" is perhaps the most widely used family of controllers in industrial and research applications. The vital role of the controller is to choose the appropriate steering (orientation) yaw rate for our mobile robot, such that it follows the designated path as closely as possible.

The PID controller consists of three main terms each responsible for a certain aspect of feedback control performance. In the following, we will go through each term and explain why it must exist to enable accurate path-following capability in our mobile robot.

- **Proportional(P)**: The proportional term in a PID controller is set to respond to the magnitude of existing error between the robot's observed position and the desired trajectory. It generates the element in the control signal which is "proportional" to the error E_k to reduce the error and bring the robot back on track. This real-time corrective action is based on the current (immediate) error which allows the robot to quickly respond to deviations from the path. If the proportional term is large, the robot may experience harsh steering commands resulting in overshoots in path following and an increased steady-state error. Therefore, we may need other terms in our controllers to account for this effect.

- **Integral (I)**: The integral term in a PID controller addresses steady-state errors resulting from the proportional term's action. By aggregating past errors through time, it applies corrective action to eliminate steady-state errors in the long run. This is crucial for robots with skid-steering motion as it helps to mitigate the resulting aggregated errors from wheel slippage, uneven terrain and wheel pressure, and imprecise kinematic formulation of robot constraints.

- **Derivative (D)**: The role of the derivative term in a PID controller is to "predict" future errors by measuring the rate of change in the error. It provides a damping effect, stabilizing the control signal and countering the effect of rapid proportional changes in the control signal. This results in a decrease in overshoots and oscillatory behavior and enables a smooth tracking of the desired path.

Having elaborated on the importance and meaning of each term in the PID controller, we can see the general form of the correcting action signal in the following equation.

$$U_k = \mathcal{K}_p E_k + \mathcal{K}_I \sum_{i\,=\,1}^{k} E_i \Delta T + \mathcal{K}_D \frac{(E_k - E_{k-1})}{\Delta T} \tag{3.11}$$

Where $\mathcal{K}_{\{.\}}$ represents the respective proportional, integral, and derivative coefficients. Such coefficients could be hand-tuned, however, it is possible to

geometrically determine them such that the tuning process is easier to address. The following figure is used to describe the process of deriving the controller gains using a look-ahead distance.

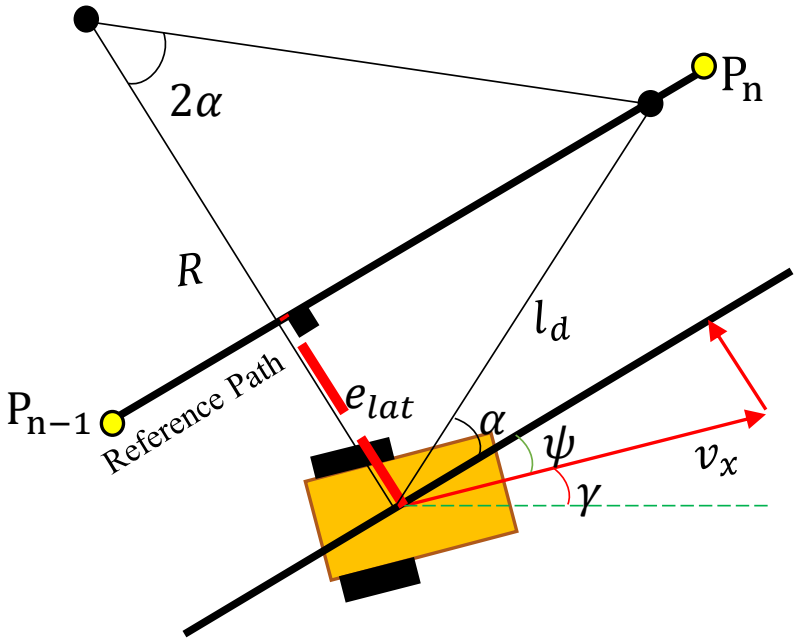

Figure 3.5: Deriving a PID controller for trajectory tracking.

Figure 3.5 shows the mobile robot along with two consecutive path way points P_{k-1} and P_k. Furthuremore, a look-ahead distance l_d is defined and the point on the path that has l_d distance from the robot is shown. The lateral distance error from the path e_{lat} is to be minimized through our control action. Moreover, the vehicle's body ICR is shown on the figure, having a radius of turn R at this moment. To find the rate of change of e_{lat}, namely $\dot{e}_{lat}$ we use the longitudinal speed of the robot and the orientation error ψ as the following:

$$\dot{e}_{lat} = v_x Sin(\psi) \tag{3.12}$$

By inspecting figure 3.5, it can be seen that $sin(\alpha) = \frac{e_{lat}}{l_d}$. In order to find the radius of turning, the sin rule is written in the triangle forming from the ICR, the robot and the look-ahead point. Concretely:

$$\frac{l_d}{Sin(2\alpha)} = \frac{R}{Sin(\pi/2 - \alpha)} \longrightarrow R = \frac{l_d}{2\ Cos(\alpha)} \tag{3.13}$$

The definition of the instantaneous center of rotation can be used as the only stationary point on the robot when performing maneuvers. This means that:

$$R = \frac{V_x\ Cos(\psi)}{\omega} \tag{3.14}$$

Combining equation 3.13 and 3.14, ω can be found(i.e. the rotational yaw rate) as:

$$\omega = \frac{2\ V_x Cos(\psi) Sin(\alpha)}{l_d} = \frac{2\ V_x Cos(\psi)}{l_d^2} e_{lat} \tag{3.15}$$

Where the adaptive coefficient $\frac{2\ V_x Cos(\psi)}{l_d^2}$ is the $\mathcal{K}_p$ we were striving to find. Having calculated the proportional and derivative coefficients through equations 3.15 and 3.12, the PID control is formulated law as:

$$\omega_k = \frac{2\ V_x Cos(\psi)}{l_d^2} E_k + \mathcal{K}_I \sum_{i\ =\ 1}^{k} E_i \Delta T + v_x Sin(\psi) \frac{(E_k - E_{k-1})}{\Delta T} \tag{3.16}$$

And V_x is chosen to be the same as the reference speed available from the designed reference path (figure fig:trajectory-definition).

While PID controllers are one of the most widely used controllers in industrial applications, there are certain shortcomings that inhibit us from implementing them for real-world scenarios and many safety-critical use cases. The first and foremost implementation detail of PID controllers is the number of adjustable parameters. Tuning each gain may be time-consuming and in many cases, may need long sessions of trial and error. Furthermore, PID is agnostic to systems kinematics constraints, choosing a possibly infeasible control command for the plant to execute. Such constraints may involve the bounds of control inputs, the constraints of smoothness and control effort of the control signals, and even the dynamics of the system. In addition to the aforementioned reasons, PID lacks the capability to adapt to changes in the plant under

different operational conditions, hindering its ability to handle complex control scenarios that require the needed control performance. Lastly, the PID is unable to handle path constraints such as existing obstacles in the way of the robot, reducing its reliability and safety.

To address the above limitations, a more sophisticated control law called "model predictive control" is developed. This advanced family of controllers has the capability of taking into account physical and kinematic constraints, enabling us to incorporate a desired behavior from the robot in an objective function to be optimized in each time step. Increasing the flexibility and reliability of our control comes with the cost of higher computation times, which is one limitation of model predictive controllers. In the next section, we will delve deep into the formulation of MPC and expand on our controller design for the mobile robot path following.

3.5 Model predictive controller

Model predictive controllers have proven to be a valuable tool in controlling material deposition systems [74, 75]. In this section of the controls chapter, the optimal control problem is introduced in a general format and elaborates on the basic definitions and concepts to control a general plant using a family of optimal controllers called the "model predictive controller". By considering the kinematic constraints of the system, the MPC has proven to be a valuable tool in addressing the challenges of trajectory tracking and control in complex environments [65–67]. The formulation of the MPC is investigated, where the performance metrics of the system are characterized through the cost function of the MPC.

The model predictive controller (MPC) has a predictive approach toward solving a control problem. Specifically, it defines a prediction horizon and calculates the aggregate sum of a cost function over this prediction horizon into the future. By optimizing the control inputs, it will be able to find the best immediate control action for the system at the current time step and applies it to the system [76].

3.5.1 The nonlinear programming problem of MPC

In the preceding chapter, we detailed how formulating the kinematics of the system enables us to predict the system's state for the next N time samples. This predictive window is referred to as the "horizon" of prediction, and it serves as the basis for defining the cost function of the control problem (see equation 3.2a). The performance objective function of the system, spanning from time step 0 to N, is defined as follows [76]:

$$\mathcal{J}_{0\longrightarrow N}(x_0, U_{0\longrightarrow N}) = p(x_N) + \sum_{i\,=\,1}^{N-1} q(x_k, u_k) \tag{3.17}$$

Where N represents the prediction horizon, and x_k denotes the state vector at time k obtained by starting from the initial state x_0. Additionally, we define the "vector of future inputs" $U_{0\longrightarrow N} = [u_0^T, ..., u_{N-1}^T]^T$. Lastly, the *stage cost* and *terminal cost*, denoted by $q(x_k, u_k)$ and $p(x_N)$ respectively, are defined to be positive definite.

$$p(x, u) > 0 \quad \forall x \neq 0 \quad u \neq 0 \tag{3.18a}$$

$$q(x, u) > 0 \quad \forall x \neq 0 \quad u \neq 0 \tag{3.18b}$$

Let us consider that the form of equation 3.19 is quite general and can be applied to a variety of control problems. To adapt it for the robot trajectory tracking problem, we need to further define the stage cost, terminal cost, and specify the constraints of the control problem. In this context, our control actions involve finding the optimal future control vector $U_{0\longrightarrow N}$ by determining the optimal value of $\mathcal{J}^*{}_{0\longrightarrow N}$.

$$\begin{gathered} \mathcal{J}^*{}_{0\longrightarrow N}(x_0) = Min\ _{\mathbf{U_0\longrightarrow N}} \mathcal{J}_{0\longrightarrow N}(x_0, U_{0\longrightarrow N}) \\ subj.\ to \quad x_{k+1} = \mathcal{F}(x_k, u_k) \\ h_i(x_k, u_k) \leq 0 \quad k = 0, ..., \ N-1 \end{gathered} \tag{3.19}$$

Where h_i denotes the i'th constraint of the system as a function of inputs and state (such as input bounds, inappropriate states, ...), and $\mathcal{F}$ is the physical kinematic constraints of the system.

3.5.2 The stage cost

In this section, the stage cost for the problem of mobile robot trajectory tracking will be defined. The stage cost encodes the feedback performance of the system in following a desired path, using the same definition of error as the PID controller in each time step.

Remark: The actuators in a real-world robotic systems are designed to take smooth control inputs with minimal necessary changes. That is why the in the stage cost, we incorporate 2 terms to encode control effort (term 2) and control smoothness (term 3). The details of these terms are explained shortly.

$$q(x_k, u_k) = E_k^T \; Q_E \; E_k + u_k^T \; R_u \; u_k + (u_k - u_{k-1})^T \rho (u_k - u_{k-1}) \tag{3.20}$$

The stage cost term is calculated for time steps 0 through N-1 for each optimization step. The coefficient matrices Q_E, R_u and ρ encode the level of importance of each performance metric for the optimization process. The **first term** in the stage cost characterizes how close the desired trajectory and orientation is being followed. If the lateral error and orientational error are large, this term will increase in magnitude. **The second** term, namely the control effort for the input $u_k = [v_x, \omega]_k$, is designed so that the controller chooses the most "energy efficient" inputs for the system, allowing it to comply with the efficiency requirements of a real-world system. Lastly, **the third** term of the stage cost is called the "proximity term". Since smooth control signals with minimized necessary changes are feasible in the context of physical actuators, it is intended to minimize the difference between two consecutive control signals such that sudden changes in the control signals are eliminated. If the difference between the two consecutive control signals is too large, this term also increases in magnitude and therefore the controller has a tendency to choose control signals that are close to the last time steps control signal.

This allows the controller to accommodate smoother trajectories and reduces sudden changes in control signals that can potentially damage the actuation system of the robot.

3.5.3 The terminal cost

The main reason to include a terminal cost term in the cost function is for the controller to choose control signals that cause the final state of the system to be closer to the desired state, at the end of the prediction horizon. Here, the intuition behind dedicating a terminal cost $p(x_N)$ is further expanded to the objective function of the model predictive controller. Let us assume a special case in which the mobile robot is placed exactly on the path it is supposed to follow like the following figure.

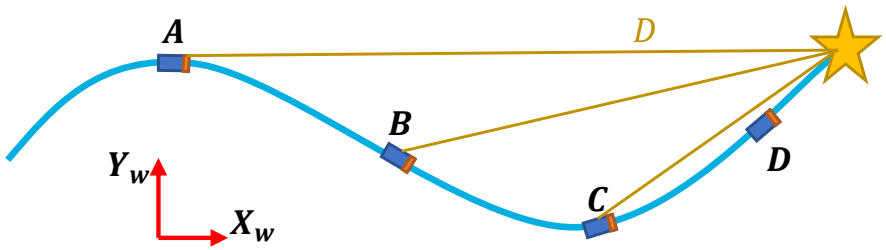

Figure 3.6: The intuition behind the terminal cost. The robot is incentivized to follow the path and reach the final goal.

The desired behavior from the robot is that in case it is precisely placed on the blue line (reference trajectory) shown in figure 3.6, it still follows the desired path and reaches the final goal, shown with a star. Now assuming that the terminal cost does not exist, the objective function of the equation 3.19, will always be zero. This is due to the fact that the lateral and orientational errors are zero and since the proximity term promotes input signals close to one another, an input signal of zero will be chosen as it has the least control effort. However, it is evident that such behavior is not desired, and our goal is to have the robot reach the endpoint of the desired trajectory. To achieve this, we include a terminal cost $p(x_N)$, which encodes the distance of the mobile robot from the final goal's location, using the following equation:

$$p(x_N)_k = D_k^T Q_f D_k \tag{3.21}$$

By incorporating the final term in the definition of the cost function, the optimal control input signal U_k is chosen at each time step and is fed to the control plant.

3.6 Summary

In this section, the problem of controlling a mobile material deposition robot with a skid-steering locomotion mechanism to follow a desired trajectory for thermal spray operation is investigated. Precise control of the mobile unit is essential to achieve uniform and controlled deposition thickness during material deposition, making trajectory-following accuracy a critical task.

Two control laws are discussed, and their mathematical equations are developed. The first control law, the proportional-integral-derivative (PID) controller, utilizes lateral and orientational errors as the primary feedback to keep the robot on track. Despite its simplicity, the PID controller lacks awareness of the robot's kinematic constraints and may require extensive tuning in practical scenarios.

To address these limitations, the model predictive controller (MPC) is introduced as an advanced optimization-based controller. The MPC incorporates system feedback errors, control effort, smoothness, and kinematic constraints into a single cost function, enabling it to characterize essential performance metrics for material deposition tasks. While the MPC is widely used in industry due to its capabilities, it comes with higher computational costs and complex implementation compared to the PID controller. However, its ability to handle system constraints makes it a suitable choice for autonomous material deposition applications.

Chapter 4

Experimental studies and discussion

In this chapter, the results of implementing the visual navigation framework, thoroughly explained in the preceding two chapters, are presented. These results encompass the evaluation of the framework's performance in two modules. The first module focuses on the state estimation, where the accuracy of localization and orientation estimation for both a mobile and a fixed robotic manipulator, designed for automated material deposition tasks, is assessed. Additionally, the performance of the controllers in tracking a predetermined trajectory using the mobile robotic platform is analyzed.

The results obtained from both simulation and real-world experiments provide evidence of the algorithm's reliable performance. The outcomes of these experiments quantitatively demonstrate the performance of the visual navigation framework. Furthermore, discussions on the controllers are conducted, and the experimental results are summarized, providing conclusive remarks on the overall system performance.

4.1 The visual state estimation algorithm

In this section, the performance of the visual state estimation algorithm, detailed in chapter 3, will be evaluated. The evaluation comprises assessing the accuracy of state estimation for both the fixed and mobile robots designed for autonomous material deposition. To begin, the results showcase the state

estimation framework's performance in the fixed robot under various spraying scenarios. Subsequently, a similar analysis will be conducted for the mobile robotic system.

4.1.1 Fixed manipulator state estimation

In this section, the performance of the localization framework is evaluated on the fixed manipulator. Our goal is to estimate the state of the end-effector of this robot. In this case, the camera sensor is placed on a tripod 2.7 m above ground level. The camera sensor's pitch angle is 45^o and the tripod is placed 1.2 m away from the robot's base. The performance of the algorithm is investigated in 3 different scenarios, in the following.

- **Investigating Position of End-Effector:** In this scenario, a rectangular movement of the end effector on a flat substrate is assumed for our application. In this case, all states of the end effector, except 2 positional states, remain constant and equal to zero. Figure 4.1 displays the trajectory of the end effector along with the estimated 2D position by the visual state estimator, and the filtered trajectory obtained after employing the state observer.

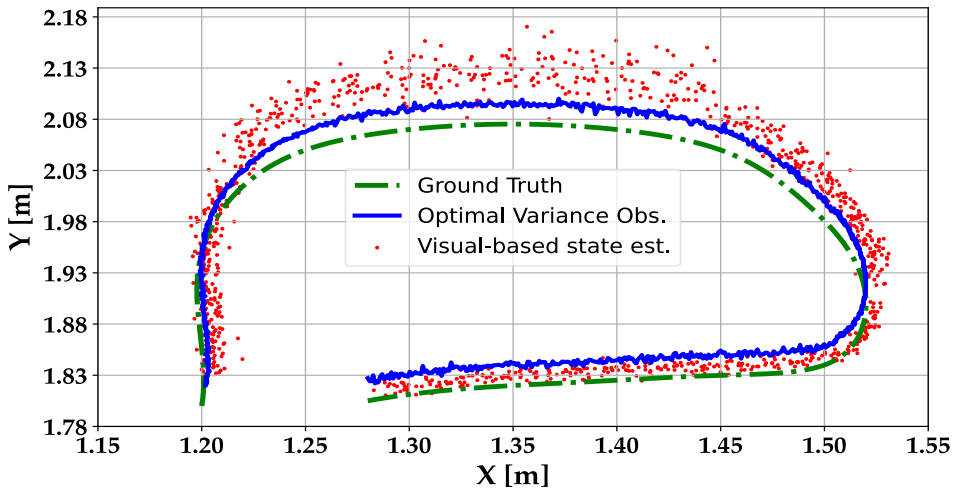

Figure 4.1: End-effector trajectory on the plate (Green), Noisy visual measurements (red), improved estimation by the motion model(blue).

It is evident from figure 4.1 that at the start of the movement (bottom

left), the visual state estimation is quite close to the actual trajectory. After a few centimeters of movement as the direction of the motion changes, there is a noticeable gap between the visual measurements and the actual trajectory due to errors in localization. However, the motion model is able to predict a different path from that of the visual observer, which results in an improvement of estimation, visible in the top horizontal part of the rectangle.

- **Investigating the yaw angle:** In this scenario, the application requires a cold spray operation on the interior surface of a cylindrical substrate, such as a pipe. The robot's control involves following a circular path in each pass, while other states, like the positional states, remain constant. The focus of this evaluation is on assessing the algorithm's performance in estimating the yaw angle of the robot's end-effector. The yaw angle is defined as the angle of rotation around the vertically perpendicular axis to the line of fire of the cold-spray nozzle.

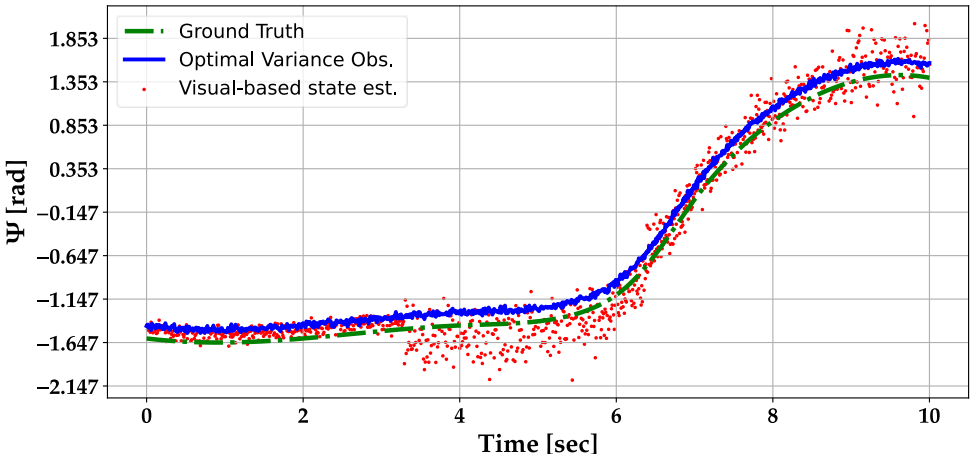

Figure 4.2: Yaw angle evaluation in the second scenario. The visual-based estimations (red dots) increase in variance between 4 and 6 seconds, compensated by the observer

It can be noticed from figure 4.2 that when the rate of change in yaw angle increases (after 4 seconds), the variance in localization increases for the visual-based estimator due to motion blur. However, the motion model fusion process is able to correct the loss of accuracy due to

perceptually degraded conditions.

- **Investigating the pitch angle:**

  The application requires that the end-effector of the robot is controlled to perform the cold-spray process on the exterior surface of a horizontally laid pipe. In this case, the end-effector is controlled such that the pitch angle, defined as the angle of rotation of the end-effector around the axis horizontally perpendicular to the line of fire, covers the angles in the range $[0, -\pi/2]$. Figure 4.3 demonstrates the actual pitch angle alongside the visually estimated pitch angle and the output of the optimal variance state observer.

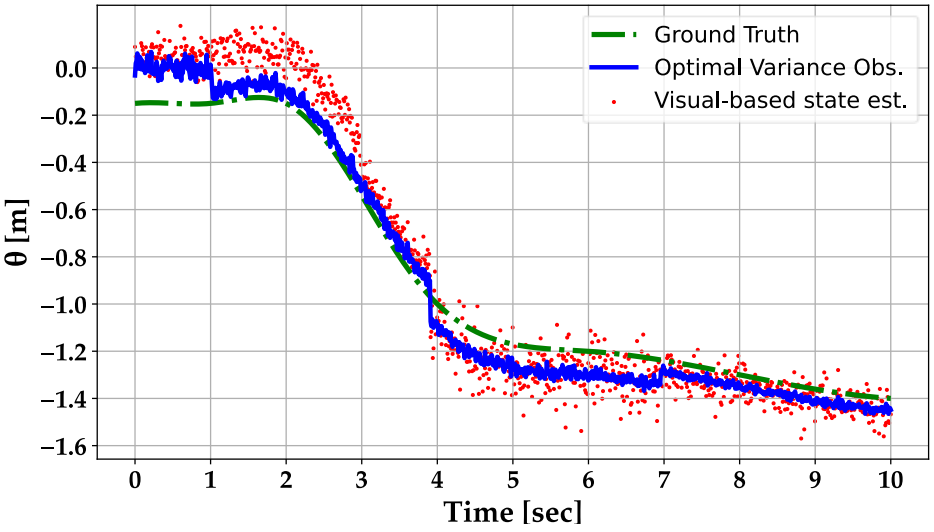

Figure 4.3: Pitch angle evaluation in the third scenario. Due to erroneous initialization, the initial error is larger compared to the error after KF-filter convergence.

As can be observed from 4.3, due to existing errors in initialization, the algorithm starts with considerable error with respect to the actual value. However, it is able to compensate for this error as the values of the visual state estimation grow large in disparity around 4 seconds after the start of the test. In this phase, the algorithm pays more attention to the predictions of the motion model, which increases the accuracy of the

prediction. Finally, figure 4.4 provides a few examples of 3D bounding boxes estimated for the case of the fixed manipulator.

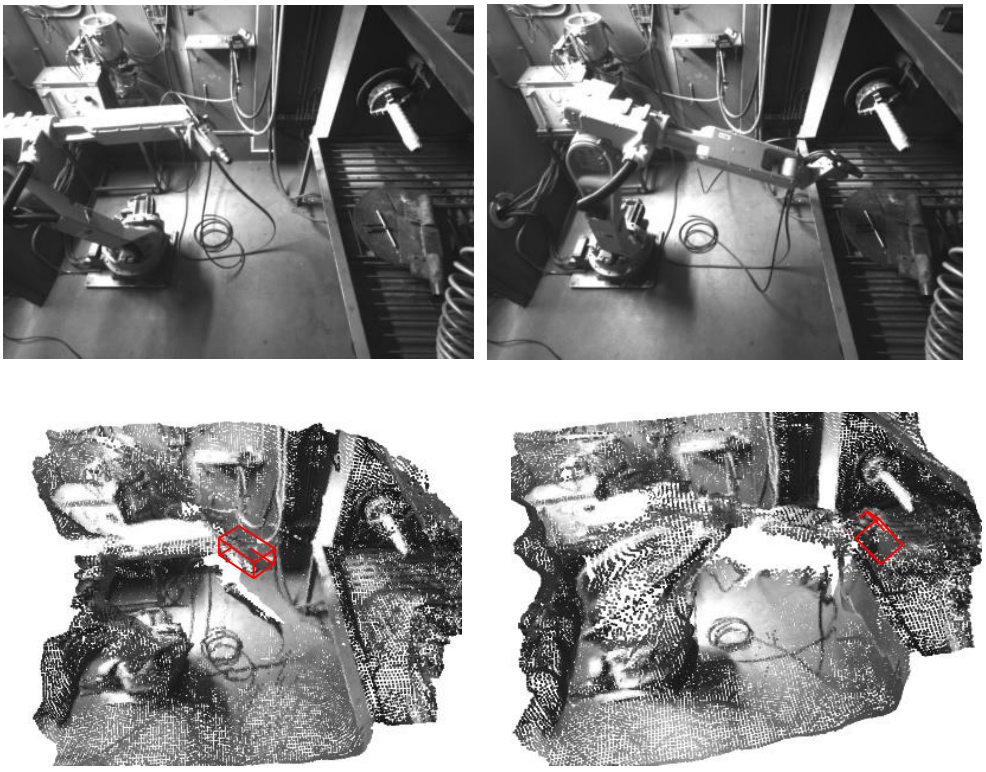

Figure 4.4: Dense point cloud representation of the fixed manipulator along with the estimated bounding box shown in red .

4.1.2 Mobile robot state estimation

In this section, the performance of the localization framework on the mobile cold-spray unit will be evaluated. The primary objective is to estimate the position and heading angle of the robot and the cold-spray unit, which is fixed on a mounting to the robot. For this evaluation, the camera sensor is positioned on a tripod at a height of 2.8 m above the ground level, with a pitch angle of 45^o. The algorithm's performance will be assessed as the robot moves along a straight and circular path. In both cases, the pitch and roll angles of the mobile unit are assumed to be zero.

- **Moving on a straight path:**

  In this scenario, the mobile robot moves in a straight line away from the camera. This experiment can be particularly useful when the mobile unit intends to spray along the length of a horizontal pipe. The camera is located above the origin of the coordinate system and the heading of the robot is constant.

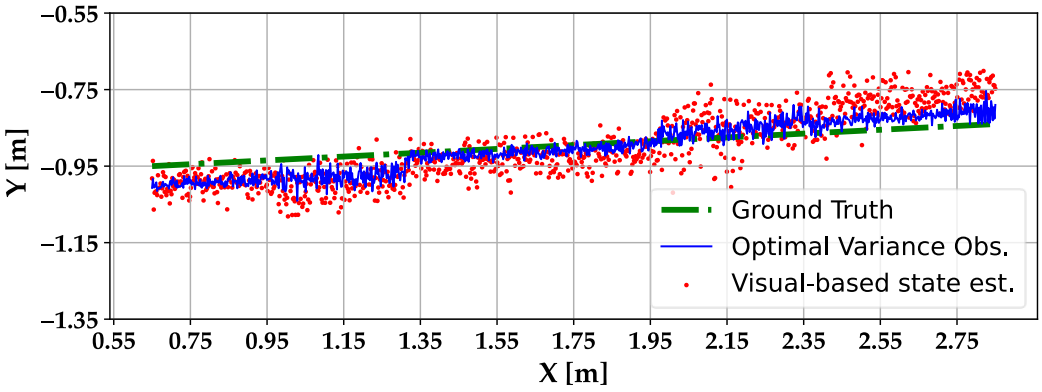

Figure 4.5: Mobile robot Trajectory in linear movement case. A larger variance at the start of motion can be observed. This is improved further on as the motion model fusion takes place.

As can be observed from figure 4.5, the state observer uses the visual measurements at the start of motion as the main element in position estimation. However, as the mobile unit moves away from the camera, the uncertainty in depth estimation increases and this affects the impact of visual measurements in the final filtered results by increasing the second term in the visual covariance matrix.

In the next case, the impact of occlusion on the mobile robot's state estimation performance will be investigated. The scenario involves the robot moving in a straight line while encountering an occluding object that blocks the camera's field of view. Consequently, the visual estimation is unable to contribute to the state estimation performance during this occluded period.

As can be seen from figure 4.6, there is an increase in visual state esti-

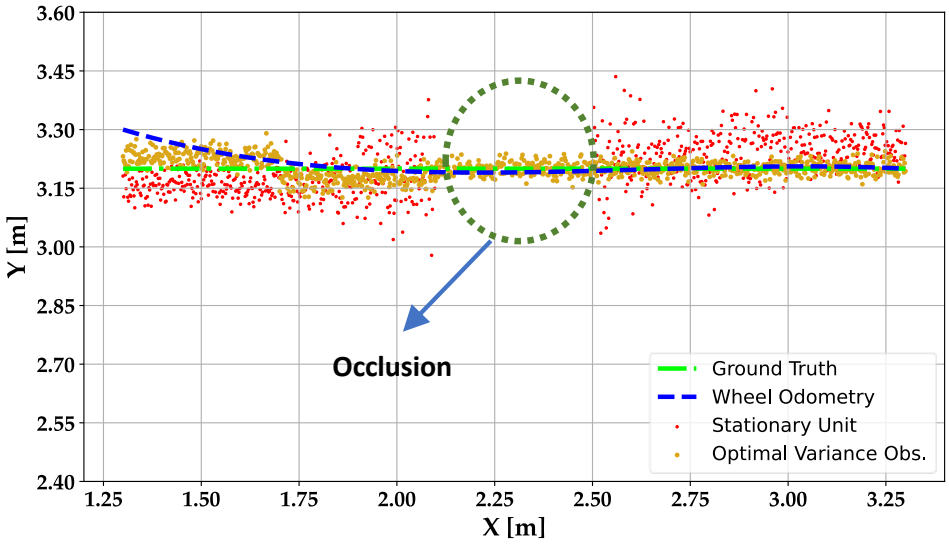

Figure 4.6: Mobile robot Trajectory in linear movement case. A larger variance at the start of motion can be observed. This is improved further on as the motion model fusion takes place.

mation variance around the area where the occlusion happens; leading to a loss of visual estimation and a shift of trust in the wheel odometry. This provides proof of accurate estimation of states in occluded cases.

- **Moving on a Circular Path:**

  In this scenario, the mobile unit performs a half-circle maneuver around the camera. The performance of position and heading angle estimation is investigated. From figure 4.7, an area in which the accuracy of the visual measurements (red dots) decreases is noticeable. This results in increased variance in localization and orientation estimation and a higher spread in visual-based estimations (red dots). Furthermore, from the position plot, a radial bias towards the center of the circle where the camera is fixed can be noticed. This offset is attributed to the fact that as the distance between the camera and the robot increases, the pixels whose depth is measured in each frame belong to one of the sides of the mobile robot. However, it can be noticed that this offset is also partially compensated by the motion model (blue line). The offset can

also be addressed by installing the camera at a greater height or by compensating for the error by adding an adaptive radial offset vector to the localization results. Finally, figure 4.4 provides a few examples of 3D bounding boxes estimated for the case of the mobile robot.

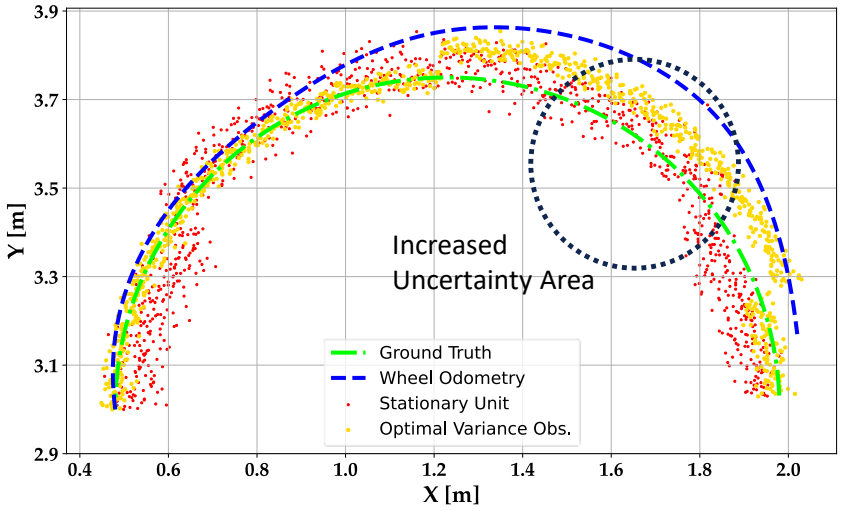

Figure 4.7: Mobile robot Trajectory in a circular movement.

4.1.3 Statistical error analysis

Error analysis in localization and orientation estimation for the case of the fixed manipulator was developed. This is done by considering the above-mentioned scenarios and by calculating the Mean absolute error (MAE), maximum error (ME), and root mean squared error (RMSE). The results of this analysis can be found in table 4.1.

Table of Errors			
Scenario	**MAE**	**ME**	**RMSE**
Flat Plate (x-y)	3.83 cm	4.37 cm	4.56 cm
Vertical Cylindrical Pipe (Yaw angle ψ)	0.18 rad	0.36 rad	0.23 rad
Horizontal Cylindrical Pipe (Pitch angle θ)	0.15 rad	0.27 rad	0.21 rad

Table 4.1: Fixed manipulator error analysis

From the above table, it can be noticed that the algorithm achieves less than 5 cm mean average error in localization of the target object and less than

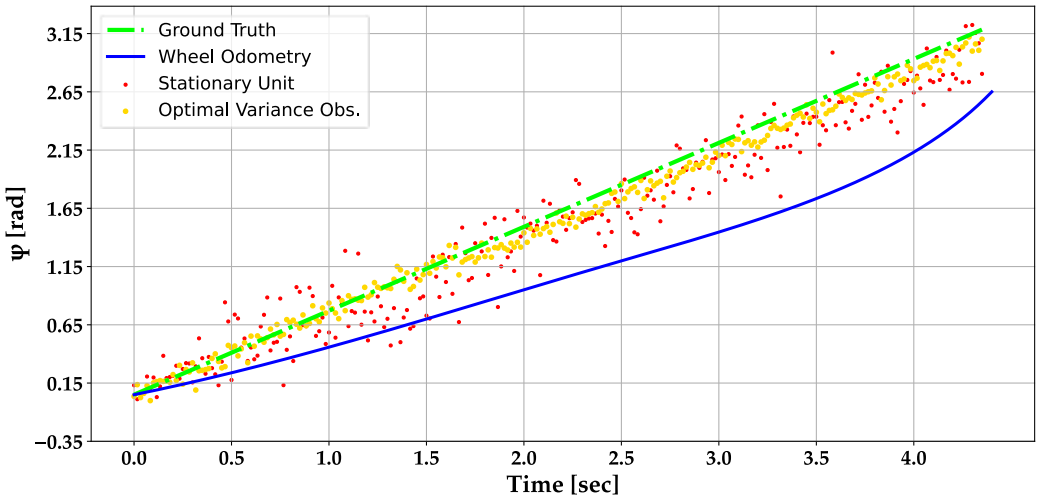

Figure 4.8: Mobile robot orientation estimation in a circular movement.

0.2 rad mean average error in orientation estimation. The roll angle is not investigated as it is not a principal state when performing material deposition using the fixed manipulator.

Next, the statistical error analysis for the mobile robot case will be discussed. Similar to the fixed robot case, the errors in each scenario will be presented and compared to the fixed robot case. The following table provides the statistical errors for each mentioned case in the mobile robot use case.

Table of Errors			
Scenario	**MAE**	**ME**	**RMSE**
Straight path 2D Position	8.05 cm	12.79 cm	9.13 cm
Circular path 2D Position	9.18 cm	11.49 cm	10.39 cm
Circular path Heading Estimation	0.37 rad	0.51 rad	0.44 rad

Table 4.2: Fixed manipulator error analysis

By comparing the results from table 4.1 and 4.2 for errors in position and heading angle estimation, it can be noticed that the errors for the mobile robot case are greater than those for the fixed manipulator case. This observation is due to the fact that the mobile robot operates in a larger workspace, making it possible for it to operate at greater distances from the camera, which in turn

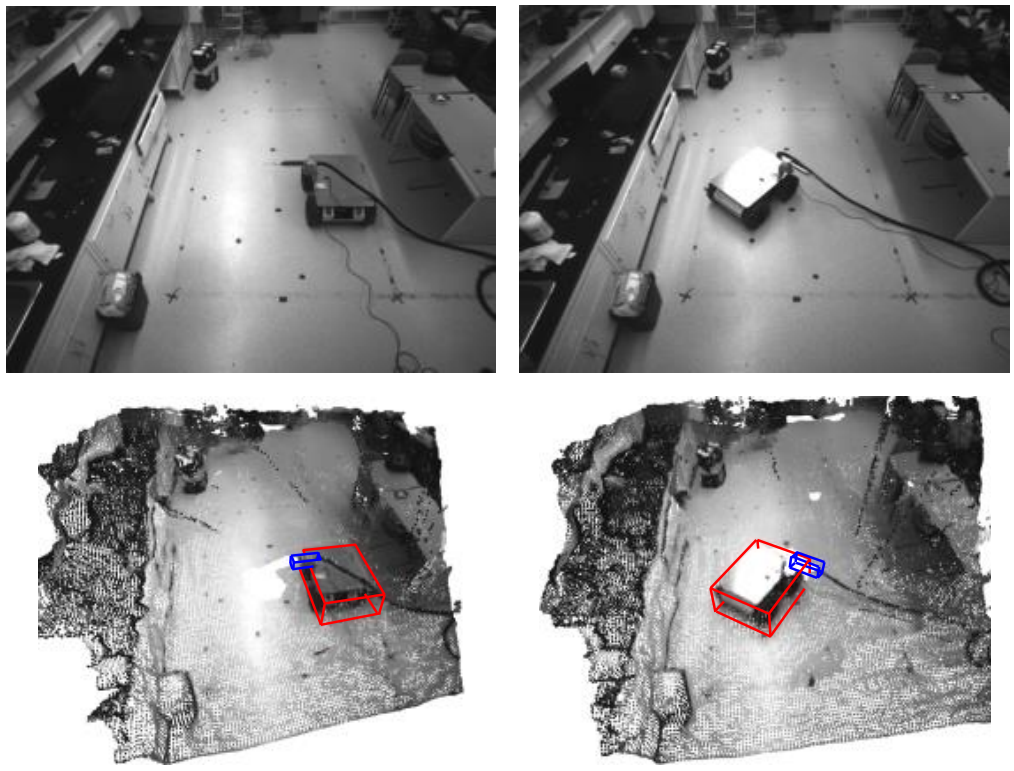

Figure 4.9: Mobile robot point cloud representation and estimated bounding boxes.

leads to an increase in depth estimation error. The depth estimation error could be significantly reduced if a camera with a larger baseline or better pixel resolution is used. Additionally, in future work, the effect of fusing localization results from multiple cameras installed in the environment could be studied to improve localization accuracy.

4.2 Controller results

This section aims to examine the efficacy of the developed proportional-Integral-Derivative (PID) and model predictive controllers (MPC), as elucidated in the preceding chapter, within the context of robot trajectory control. Our investigation encompasses an evaluation of controller performance through both simulation and real-life scenarios. The Gazebo Simulator is employed as the

simulation environment, while the real-life experimentation is constrained to the mobile robot configuration. The structure of this section commences with a comprehensive exposition of the MPC and PID controller performance in simulation, alongside a comparative analysis of their respective merits. Subsequently, quantitative and qualitative outcomes obtained from the real-life implementation using the Husky robot are presented.

4.2.1 Simulation results

The simulation results play a fundamental role in evaluating the performance of the designed PID and MPC controllers for the mobile robot trajectory control application. The "Gazebo Simulator," a widely adopted tool, is utilized to subject the controllers to diverse simulated scenarios, enabling the understanding of their capabilities, strengths, and limitations in handling various trajectory-tracking tasks. A comprehensive examination of key metrics, such as tracking accuracy, response time, and stability, provides valuable insights into the implementation of these two widely used controllers. Moreover, a comparative analysis between the two controllers is conducted to identify their respective areas of improvement. The simulation serves as a crucial step, establishing a solid foundation for further exploration and verification of the controllers in real-life setups. Figure 4.10 showcases the simulated husky robot in the gazebo environment.

The first test case involves instructing the controllers to follow a linear. The resulting position plots for the MPC and PID controller are as shown in Figure 4.11.

In this experiment, both controllers are initialized with a 20 (cm) lateral error and 1 (rad) orientation error. Analyzing the top figure, which represents the performance of the model predictive controller, it becomes evident that after 1 m of operation, the controller converges within close proximity of 2 cm to the desired trajectory. In contrast, the PID controller requires a significantly longer duration of operation, specifically 2.5 m, to attain a similar level of convergence within the designated trajectory. Moreover, the model predictive controller exhibits a remarkable attribute of smooth trajectory tracking,

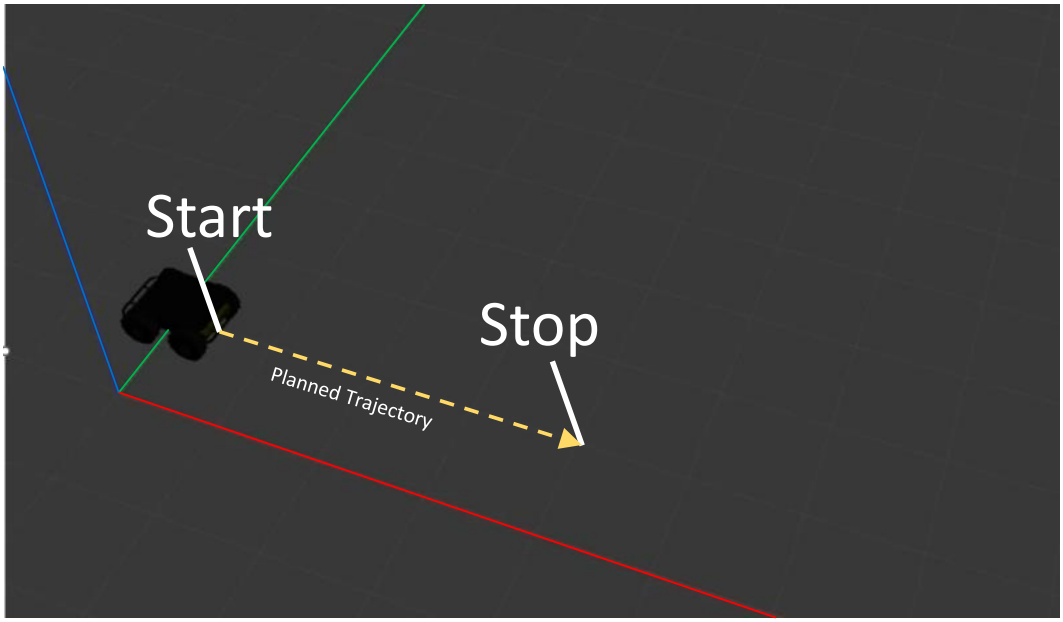

Figure 4.10: The husky mobile robot spawned in the Gazebo simulator for a simple line tracking case.

characterized by the absence of discernible overshoots and undershoots. Conversely, the PID controller displays two instances of overshoot, measuring 35 cm and 5 cm, respectively, along with an undershoot of 15 cm. The superior trajectory tracking performance of the model predictive controller can be attributed to its inherent ability to compute the most optimal linear and angular velocity solutions, coupled with the integration of a proximity term for enforcing soft constraints on the inputs. These factors collectively contribute to the enhanced precision and stability exhibited by the MPC controller in comparison to its PID counterpart.

The next plot worth discussing at this stage is the orientation plot of the MPC and PID controllers for this case. Figure **??** and figure **??** demonstrate the orientation plots of the PID and MPC controllers respectively.

The two depicted plots exhibit intriguing characteristics and notable distinctions in control performance. Upon close examination of figure 4.12, it becomes evident once again that the MPC controller demonstrates an ability to smoothly align the robot's orientation with the designated trajectory. No-

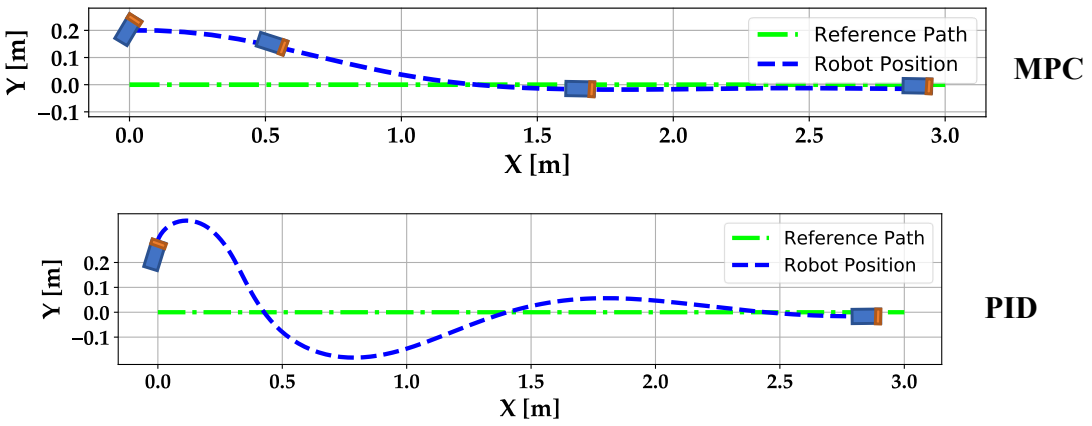

Figure 4.11: Simulated position plot of the MPC and the PID controllers. Optimal response time and smoother trajectory can be observed from the MPC plot in contrast to the PID plot.

tably, both plots commence with an initial orientation error of 1 rad. In the case of the PID controller, a rapid reduction in orientation error towards the target orientation of 0 rad is observed. However, due to the abrupt steering input, the PID controller fails to precisely attain the target orientation and exhibits an undershoot of 0.6 rad. Conversely, the model predictive controller showcases a superior performance, characterized by a significantly reduced undershoot and rapid convergence to the orientation goal of 0 rad within 7 seconds, in contrast to the 10 seconds required by the PID controller. Nevertheless, a momentary decrease in the yaw angle is observed in the trajectory of the MPC controller subsequent to the initial convergence. This behavior can be attributed to the intrinsic nature of the MPC controller, which relies on numerical optimization of the object cost function within the optimal control problem. Such transient instabilities are not uncommon and are typically followed by a period of stability, which, in our case, spans approximately 2 seconds and is accompanied by an increase in the trajectory tracking error term. However, in real-life scenarios involving safety-critical systems driven by such controllers, it is crucial to address and mitigate these instabilities by employing more complex cost functions.

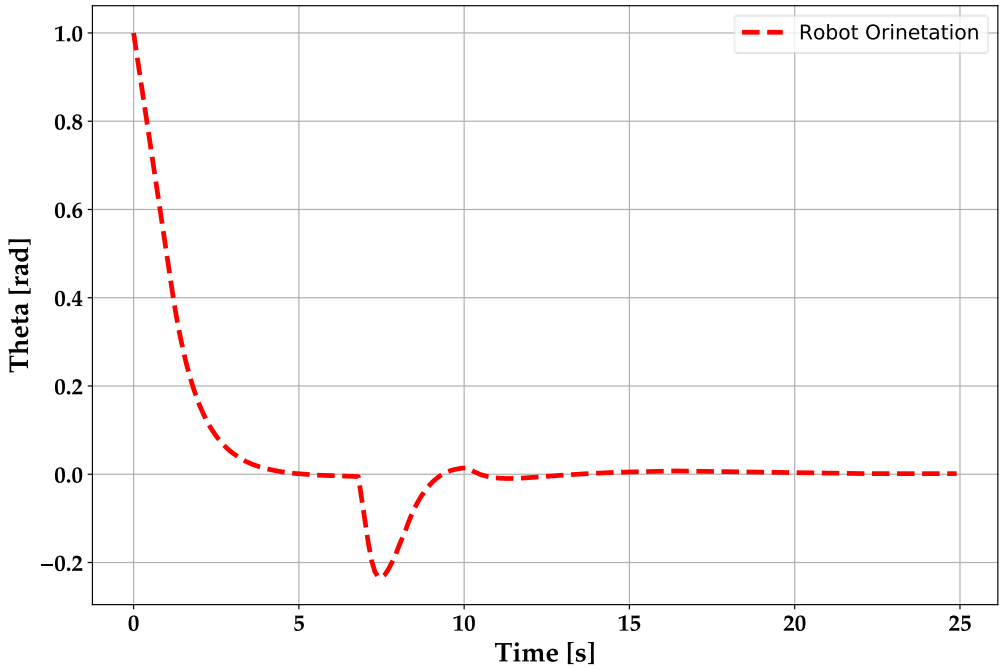

Figure 4.12: Model Predictive Controller's orientation plot for the simulated linear path following

Another significant aspect that requires discussion at this stage pertains to the stability characteristics of the two controllers. Interestingly, the MPC controller demonstrates the ability to maintain a steady state error near 0 rad once it has converged to the final orientation goal. In contrast, the PID controller fails to sustain this steady state error and directs the robot to initiate rotational motion, causing it to rotate around itself upon reaching the goal orientation. This behavior stems from the presence of the integral term in the PID controller, which accounts for the accumulation of errors over time. To circumvent such behaviors in real-world scenarios, it is imperative to adopt adaptive strategies for determining the PID controller's terms, thereby ensuring more stable performance.

Next, the input commands generated by the two controllers through the following pair of plots are demonstrated for the angular velocity.

The plot presented in Figure 4.14 provides valuable insights into the comparative performance of the Proportional-Integral-Derivative (PID) controller and the Model Predictive Controller (MPC), emphasizing distinctive charac-

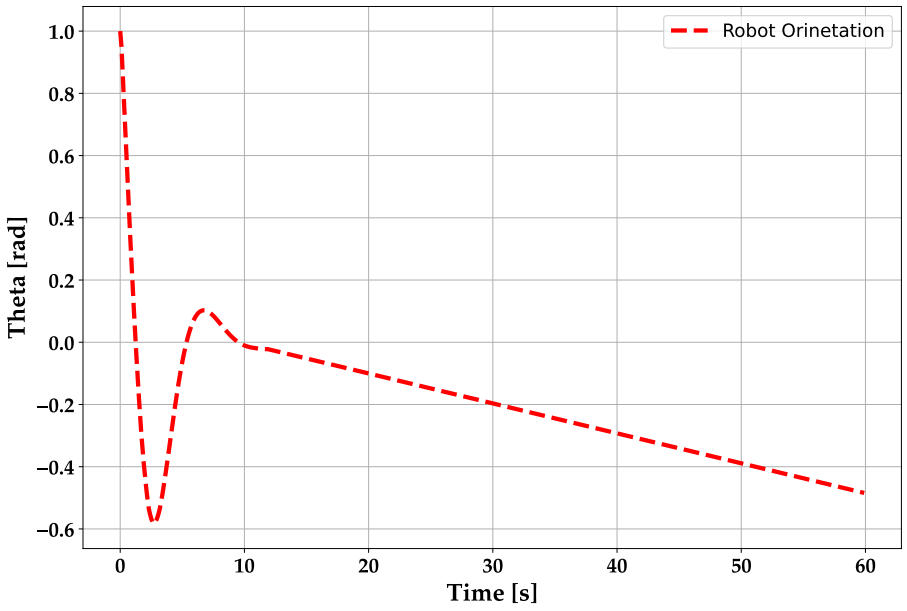

Figure 4.13: Proportional-Integral-Derivative controller's orientation plot for the simulated linear path following

teristics and notable advantages of the latter, particularly in generating angular velocity inputs. In the first plot, a series of abrupt changes in the input generated by the PID controller is observed, while the MPC controller exhibits superior performance. The PID controller demonstrates a sequence of undershoots and overshoots before eventually converging to a near-zero angular velocity control command. It is important to note that the steady-state value attained by the PID controller is not precisely zero, resulting in a minor rotational movement for the robot following the achievement of the goal.

Conversely, the MPC controller shows a significantly faster convergence to the desired input command of 0 rad/sec, achieving this milestone approximately 10 seconds after the operation's commencement. On the other hand, the PID controller requires approximately 30 seconds to achieve comparable results, highlighting the MPC's capacity for expedient convergence.

The last part of this comparative study involves the linear velocity plots generated by the two controllers.

In the present scenario, both controllers exhibit satisfactory efficacy in

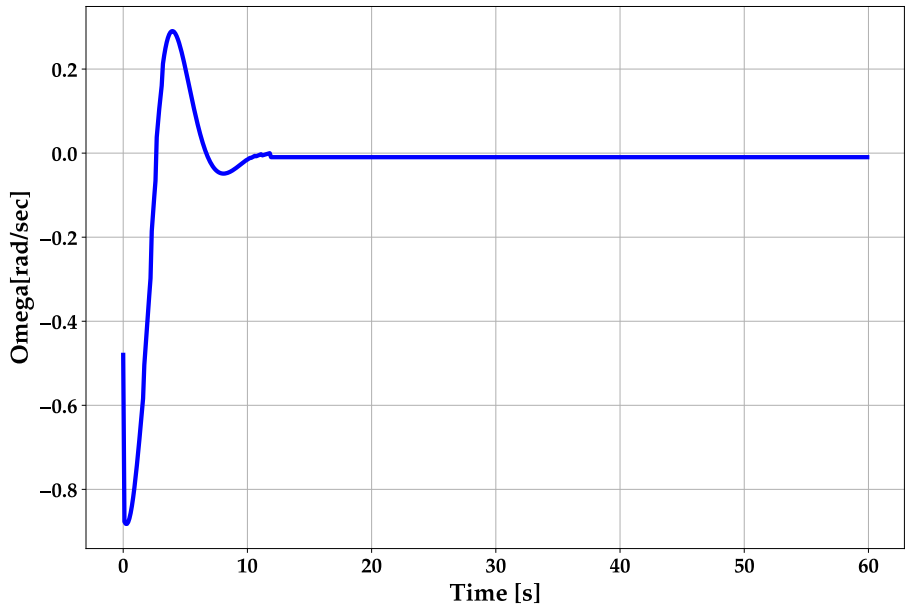

Figure 4.14: Model predictive controller's angular velocity command for the simulated linear path

attaining the desired controller input velocity of 0. Consistent with the earlier analysis of controller performance, the absence of abrupt overshoots and the presence of a smooth input velocity command in the control signal generated by the model predictive controller (MPC) is observed. Notably, the MPC demonstrates a convergence time of approximately 11 seconds, while the PID controller requires approximately 55 seconds to achieve the same outcome. This discrepancy in convergence time provides compelling evidence of the superior efficiency of the model predictive controller in propelling the robot toward its designated trajectory.

The table below summarizes our findings in controlling the robot on a linear path in simulation through the utilization of the MPC and PID controllers.

The analysis of the data presented in Table 4.3 sheds light on the comparative performance of the model predictive controller (MPC) and the proportional-Integral-Derivative (PID) controller. A careful examination reveals that the MPC exhibits a greater degree of stability in generating control commands, characterized by significantly reduced overshoots and undershoots. Moreover,

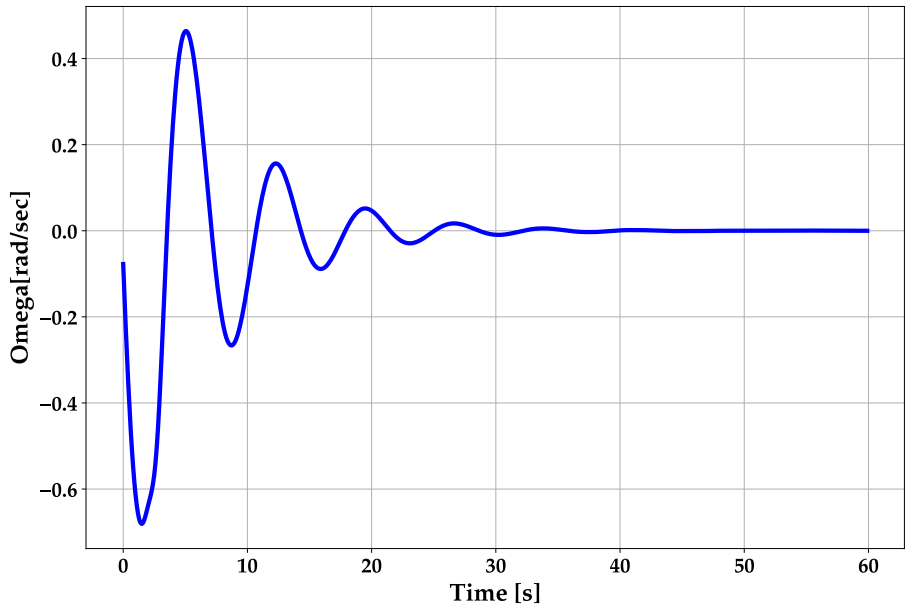

Figure 4.15: PID's angular velocity command for the simulated linear path

the MPC showcases faster settling times when compared to its PID counterpart. However, the PID controller holds a notable advantage in terms of control command calculation time, as it can compute the control command 20 times faster than the MPC. This attribute can prove advantageous in scenarios where the onboard computation device is constrained and unable to handle the computational demands typically required by the model predictive controller.

To conclude our comparative analysis in simulation, the designed controller is subjected to a more intricate testing scenario, involving the robot's navigation along a square-shaped path with two corner turns. At the onset, the robot was positioned 20 cm away from the desired trajectory, accompanied by a 45-degree orientation error angle. The outcomes of this rigorous test are illustrated in figure 4.18.

A thorough examination of figure 4.18 provides valuable insights into the performance of both the model predictive controller (MPC) and the proportional-Integral-Derivative (PID) controller in a challenging square path scenario. Specifically, in the case of the PID controller, a sharper initial steering com-

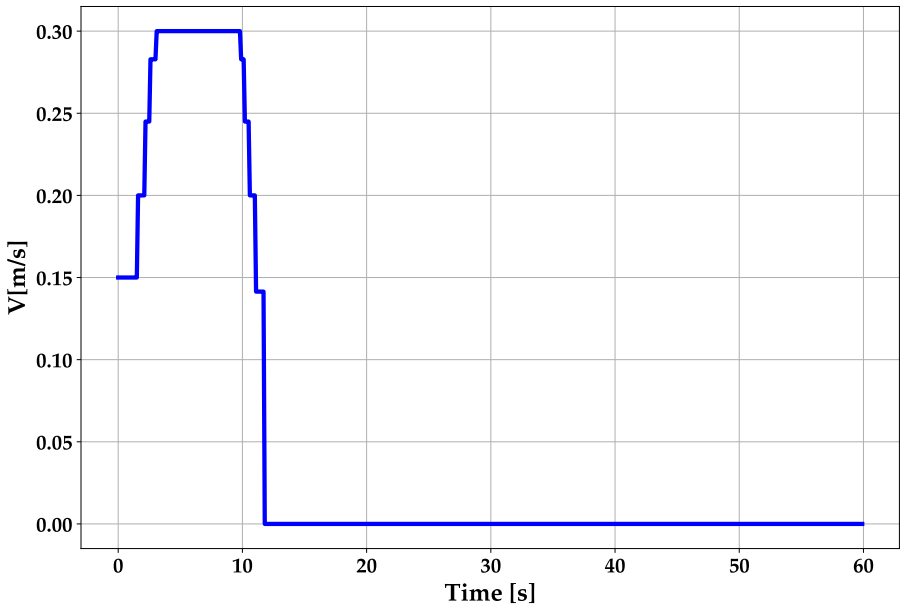

Figure 4.16: Model predictive controller's linear velocity command for the simulated linear path

mand is observed, resulting in an overshoot at the outset. Conversely, the model predictive controller demonstrates a more precise control strategy, avoiding any significant overshoot during the trajectory initialization. As the robot progresses along the path, the MPC exhibits an ability to closely follow the desired trajectory and execute turns with a high degree of accuracy. In contrast, the PID controller proves to be less efficient in turning scenarios, leading to an increase in lateral error when encountering the right turn. The accumulated error becomes apparent as the robot completes the second right turn in the case of the PID controller, where the lateral error persists as the robot moves vertically downwards towards the final position. This discrepancy highlights the superior performance of the MPC in maintaining trajectory precision and mitigating error accumulation during complex turning maneuvers. The table below summarizes the comparative performance test of the controllers in the square-shaped path following scenario:

Upon careful examination of Table 4.4, which provides a quantified summary of the comparative analysis between the PID and MPC controllers in

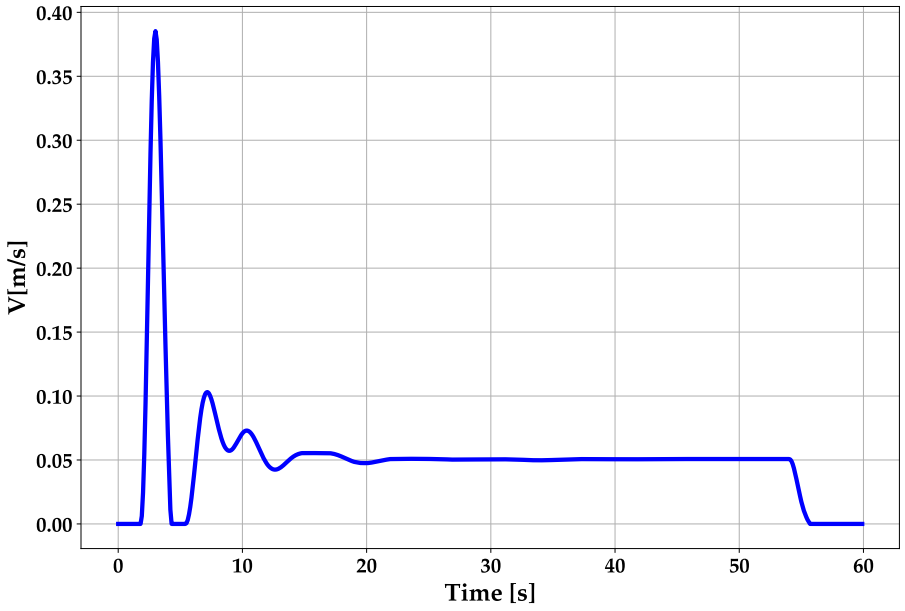

Figure 4.17: PID's angular velocity command for the simulated linear path

square-shaped path tracking, it becomes evident that the MPC controller exhibits superior tracking performance. Concretely, the results highlight the absence of visible overshoots in the MPC's trajectory, while the PID controller demonstrates a maximum overshoot of 26 cm, indicating a significant deviation from the desired path. This observation raises concerns about the suitability of the PID controller for applications that demand precise tracking, such as automated material deposition, where deviations can have adverse consequences on deposition performance and quality.

Given the aforementioned considerations, a decision was made to focus the real-life experimentation solely on the Model Predictive Controller (MPC), which has consistently exhibited efficient performance in both simulation and comparative studies. In the upcoming section, a comprehensive investigation of the MPC controller's capabilities will be conducted, subjecting the real Husky robot to various tracking scenarios. Through this in-depth examination, a deeper understanding of the controller's performance and its applicability in real-world settings is sought.

Table of comparative study in linear path tracking (Simulation)		
Metric	**MPC**	**PID**
Path Tracking Overshoot Count	0	2
Path Tracking Maximum Overshoot	0	34cm
Path Tracking Undershoot Count	0	1
Path Tracking Maximum Undershoot	0	17cm
Orientation response time	4.2sec	10sec
Orientation steady state Error	$\approxeq 0$	Unstable
Orientation Undershoot	0.2rad	0.55rad
Linear Velocity Settling time	11sec	55sec
Angular Velocity Settling time	11sec	30sec
Control Command Calculation Time	$\approxeq 20ms$	1ms

Table 4.3: Performance analysis table of MPC and PID controllers in linear path following

Table of comparative study in square-shaped path tracking (Simulation)		
Metric	**MPC**	**PID**
Mean lateral Error	$\leq 1cm$	15.2cm
Mean Orientation Error	$\leq 0.05rad$	0.27rad
Max Lateral Overshoot Magnitude	0	26cm
Time to Reach Goal	34sec	19sec

Table 4.4: Performance analysis table of MPC and PID controllers in square-shaped path following

4.3 Real setup results

The primary objective of this section is to conduct a real-life evaluation of the model predictive controller (MPC) using a mobile robot platform. To achieve this goal, a series of tests will be conducted wherein the robot will be exposed to diverse desired trajectories, and the corresponding estimated states and control inputs will be recorded. The performance of the MPC will be thoroughly assessed based on key metrics, including lateral and orientational tracking accuracy, control effort, and trajectory execution time. The obtained results will be analyzed, quantified, and summarized, leading us to draw final conclusions regarding the performance and suitability of the MPC in practical real-life scenarios.

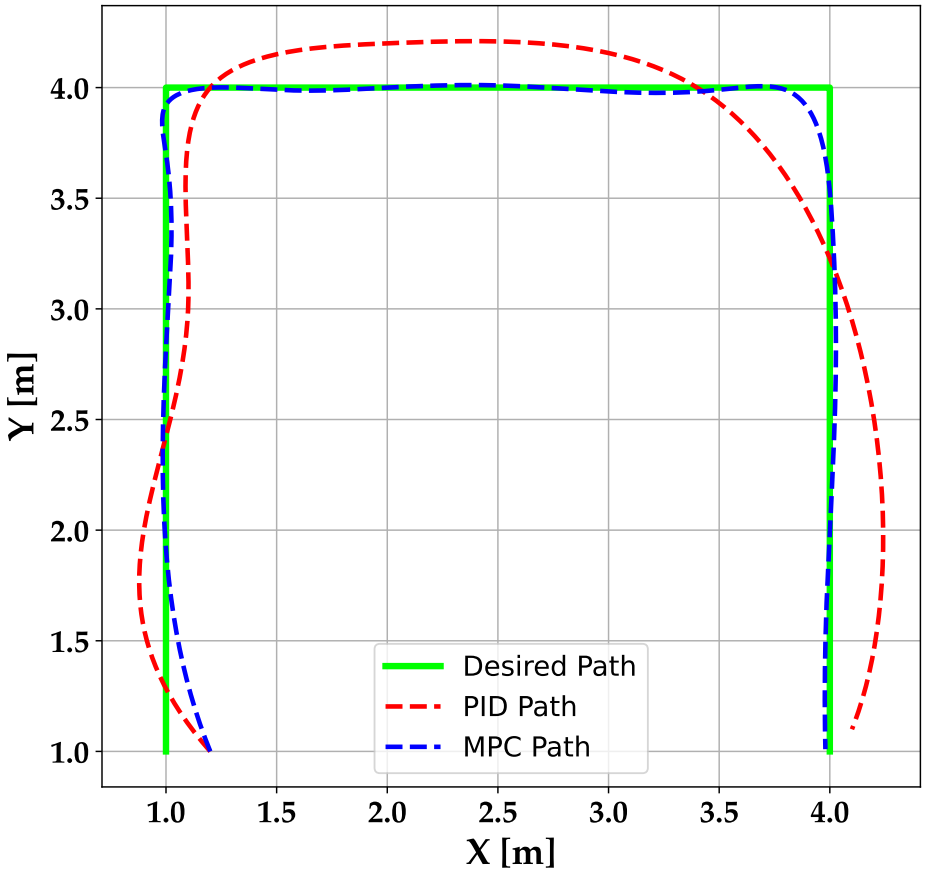

Figure 4.18: The square-shaped desired trajectory test case. The MPC outperforms PID in closely following the path and reaching the final goal.

4.3.1 Moving on a straight path

The initial scenario chosen to assess the performance of the model predictive controller (MPC) using the real setup involves the robot's motion along a linear, straight path. While this scenario may appear straightforward, it is essential to note that in practice, autonomous material deposition agents often traverse straight lines in a back-and-forth manner to achieve complete coverage of a designated deposition area. Moreover, despite its apparent simplicity, this fundamental test offers valuable insights into the controller's performance and stability in various real-life use cases. By evaluating the MPC's ability

to navigate the robot along a straight path with precision and stability, a deeper understanding of its suitability in practical applications can be gained. The following figure illustrates the husky mobile robot while undertaking the straight path maneuver.

Figure 4.19: The husky mobile robot, being controlled on a linear path. The robotic platform is able to execute this task proficiently while following a defined linear path.

Figure 4.20 demonstrates the angular velocity signal generated by the controller to drive the mobile robot on a straight path.

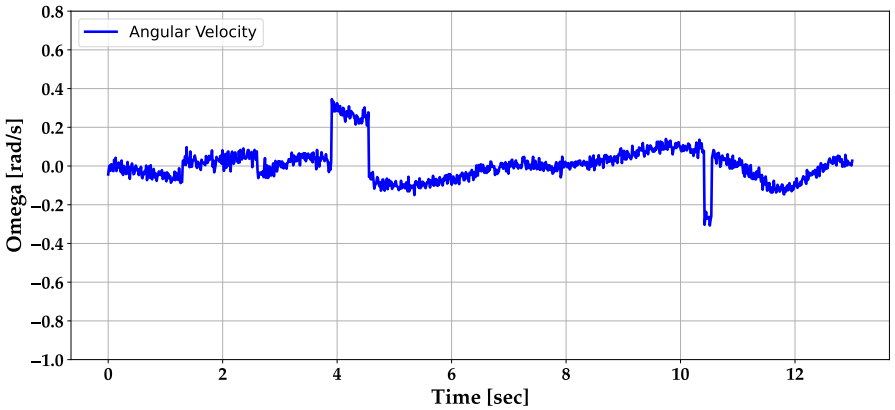

Figure 4.20: Angular velocity of the husky robot moving on a straight path. As expected, the generated control signal is close to zero for most moments of operation.

As can be observed from figure 4.20, it becomes evident that the controller executes angular velocities that are mostly close to zero for most moments of making the straight line maneuver. This matches our expectation as the most optimal control command sequence for angular velocity to move on a straight path is a command sequence that is close to zero. In spite of this fact, there

are moments in which the angular velocity has sudden changes in magnitude, which may be attributed to common local minimums in the task execution cost function of the optimal control problem which may lead to suboptimal solutions for the control signal in real-life scenarios. These sudden jumps are amplified as there are noises and inaccuracies in the state estimation module as well. In order to mitigate this issue, one can impose hard constraints on the optimization program in order to avoid generating control signals that do not match the real-life actuation capabilities of the mobile robot.

In addition to the above-described diagram, the linear velocity plot which will expose interesting characteristics of the MPC.

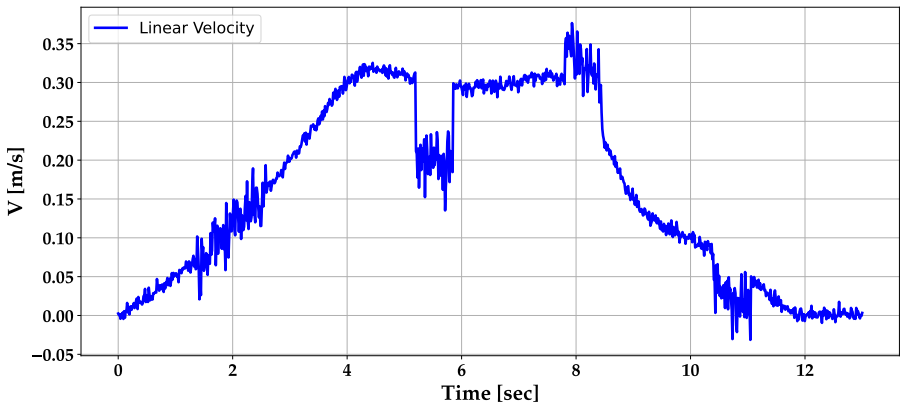

Figure 4.21: Linear velocity of the husky robot moving on a straight path. As expected, the generated control signal involves an area of initial increase, an area where it is mostly constant, and an area that lead to a complete stop.

Figure 4.21 reveals the occasional moments where there are sudden changes in the linear velocity. The plot exhibits three distinct areas: an initial 4-second-long increase in linear velocity, followed by a 2-second-long region of constant velocity, and finally, an area of decreased linear velocity. Consequently, the robot gains speed in the first few seconds, maintains that speed for a period, and eventually comes to a stop at the end of the path. Similarly observed jumps are present in the linear velocity, although their impact on control performance is not substantial, as the optimizer is constrained by hard limits on the generated control signals.

4.3.2 Making a right turn

The next test scenario employed to evaluate the performance of the model predictive controller (MPC) in a real-life setting involves the execution of a right turn. Figure 4.22 depicts four key steps of the robot undertaking this maneuver. The right turn represents a frequently encountered and extensively performed action in various applications, such as automated material deposition. For instance, in the context of cold spraying the outer surface of a curved pipe, an automated system may necessitate precise control during this particular maneuver. Therefore, this test scenario serves as a pertinent case study within our comprehensive analysis, enabling us to thoroughly assess the efficacy and suitability of the MPC controller in real-life applications.

Figure 4.22: The Husky mobile robot making a right turn using the model predictive controller.

This test case involves the robot navigating through an elbow-shaped path, comprising a vertical segment followed by a sharp right turn that leads to the final segment of the path. The robotic platform executes this operation, as demonstrated in Figure 4.22. This particular test case serves as a crucial assessment of the robot's ability to maneuver through complex trajectories involving both linear and angular movements.

To assess the performance of the automated navigation framework, the state estimation figure is investigated, involving the visual state estimation as well as the Kalman filter's output.

By investigating figure 4.23 it is possible to observe that the 2 main sensory sources (i.e. wheel odometry and stationary camera) work in tandem to estimate the state of the robotic platform in this maneuver. Specifically, while the robot is moving vertically toward the right turn, a drift in localization can

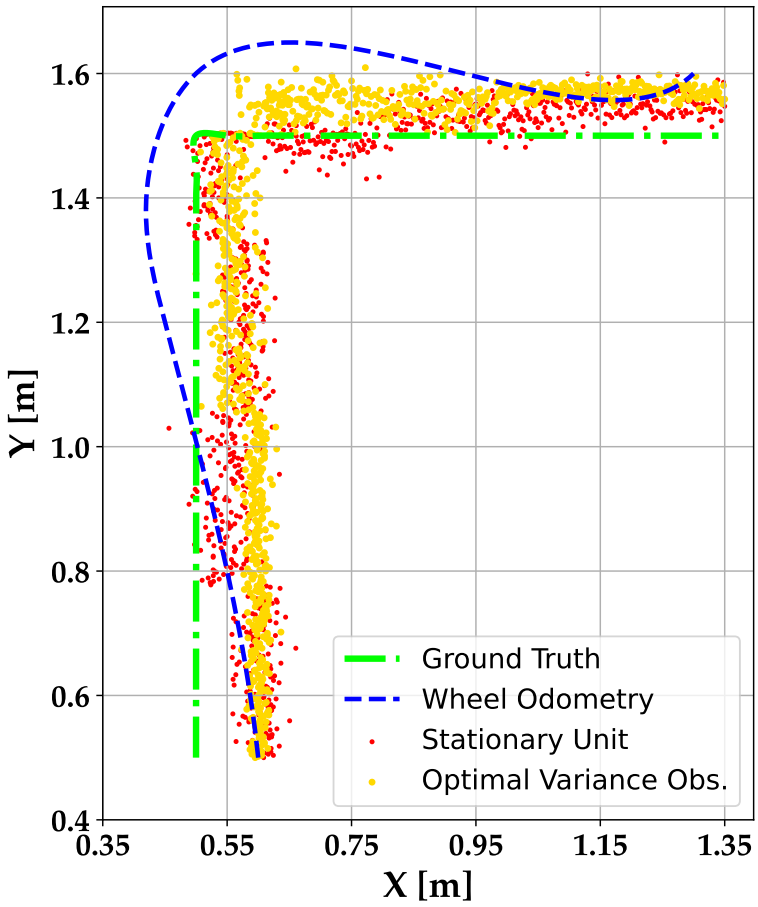

Figure 4.23: The 2D state estimation plot of the right turn executed by the model predictive controller.

be seen in both the odometry and the stationary camera. However, since the information from these two sensors is fused in the optimal variance observer, the resulting estimated position has a higher degree of accuracy compared to each individual sensory unit.

Additionally, the linear and angular velocity plots reveal further information regarding the real-life performance of the MPC controller.

In Figure 4.24, the log of the linear velocity command generated by the model predictive controller (MPC) during the right-turn scenario is plotted. A detailed analysis of the plot reveals two distinct flat peaks, corresponding

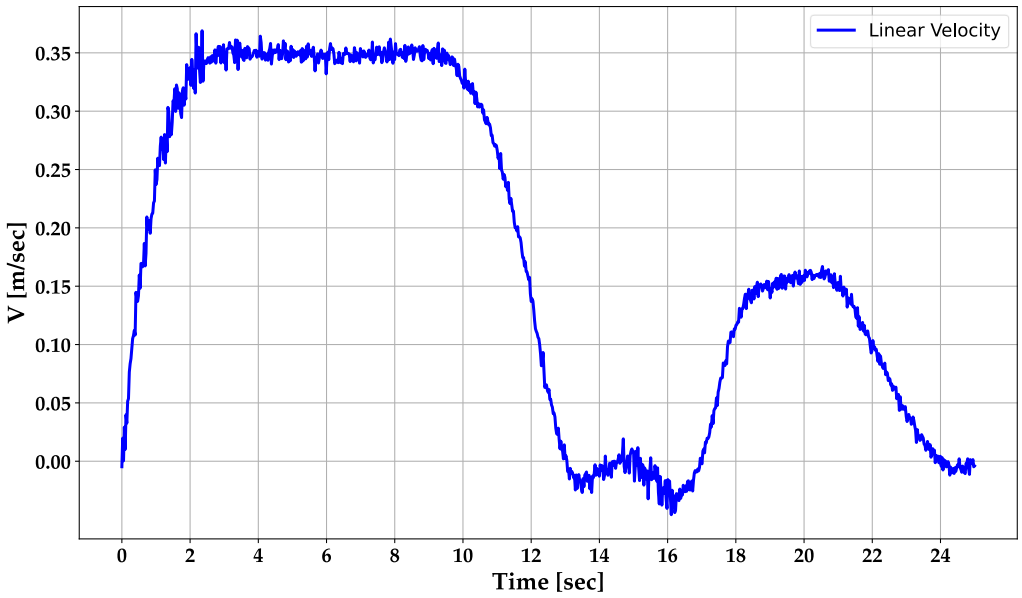

Figure 4.24: The linear velocity command log generated by the model predictive controller during the right turn.

to the vertical and horizontal sections of the desired trajectory, respectively. Furthermore, the middle segment of the plot corresponds to the execution of the right turn maneuver, characterized by a 2-second pause in the robot's linear velocity. To gain further insight into the controller's behavior during this critical phase, it is imperative to examine the log of the angular velocity command as well.

One notable characteristic of the two command log plots is the existing random noise across their span. This is attributed to the nature of the state estimation algorithm, where existing noises in the sensing modules such as depth estimation error or wheel drift can cause oscillations in the estimated state of the mobile platform. However, this is not significant enough to hinder the performance of the model predictive controller as it is able to withstand the existing noise.

The table below summarizes the performance metrics of the model predictive controller in the testing scenarios.

Table 4.5 summarizes the real-life experimentation of the model predictive controller. The quantitative error analysis suggests that since both scenarios

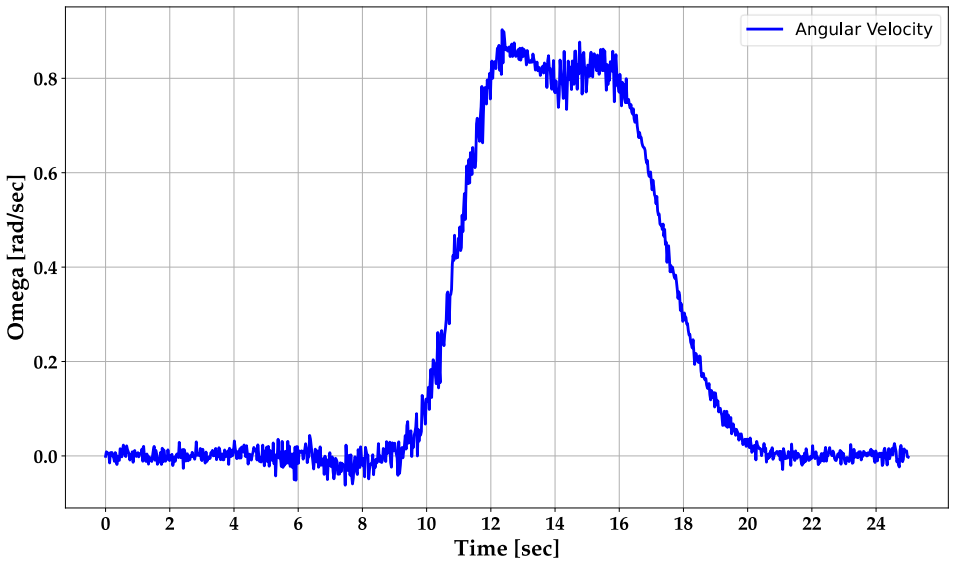

Figure 4.25: The angular velocity command log generated by the model predictive controller during the right turn.

are subject to the same lateral and orientation errors, their settling times are equal and close to 2-3 seconds. However, Since the right turn poses a greater challenge in path following for the robot, we observe that the average and maximum errors in orientation and position are generally higher.

4.4 Discussion on the controllers

In the preceding sections, extensive experimental tests were conducted to comprehensively evaluate the performance of the controllers designed in the previous chapters. The findings highlight the limitations of the proportional-Integral-Derivative (PID) controller, which exhibited several deficiencies. Specifically, the PID controller displayed a lack of awareness of the state and input constraints inherent in the system. Moreover, occasional overshoots and undershoots in the generation of control commands were observed. These fluctuations not only compromised the controller's ability to precisely follow the desired trajectory but also raised concerns regarding its practical implementation in real-life scenarios.

Table of quantitative error analysis of MPC in real setup		
Metric	**Straight Path**	**Right Turn**
Initial Lateral Error	10 cm	10 cm
Initial Orientation Error	0 rad	0 rad
Mean lateral Error	3.9 cm	8.1 cm
Mean Orientation Error	0.17 rad	0.41 rad
Max Lateral Error	9.7 cm	15.2 cm
Max Orientation Error	0.33 rad	0.57 rad
Settling time	2.1 sec	2.9 sec

Table 4.5: Performance analysis table of MPC in the real-life experimentation setup

In contrast, the model predictive controller (MPC) demonstrated a higher level of flexibility in its design, incorporating performance metrics and accommodating necessary hard constraints on the control inputs through the integration of a comprehensive and tunable cost function. This feature renders the MPC a safer choice in safety-critical systems. The MPC's ability to calculate the optimal control input for each time step addresses a crucial shortcoming in the derivation of the PID controller.

Furthermore, the control signals generated by the MPC exhibited superior stability, absence of over/undershoots, and seamless integration with the desired path. These desirable characteristics make the MPC the controller of choice in numerous real-life applications where a comfortable user experience is paramount, such as manned or unmanned autonomous agents.

An additional advantage of the MPC lies in its adaptability to various operating conditions and environmental factors. The ability to consider dynamic constraints and update control inputs based on real-time feedback enables the MPC to respond to changing scenarios, ensuring satisfactory performance in tracking challenging paths. Such dynamic features of the environment include the presence of obstacles, tools in the workspace, or even human beings. Collision with such dynamic entities of the environment can be easily avoided by adding soft or hard constraints to the optimization program.

Moreover, the MPC's ability to handle complex trajectories, such as sharp turns and intricate path patterns, with precision and stability further high-

lights its suitability for demanding tasks. The comprehensive analysis of the MPC's performance across various test scenarios demonstrated its remarkable capability to maintain accurate trajectory tracking, even in challenging situations.

It is also worth mentioning that the computational efficiency of the MPC can be a significant factor to consider in certain applications. While the MPC may require more computational resources compared to the PID controller, advancements in hardware technology and algorithm optimizations can alleviate potential concerns related to the computational burden, making the MPC an increasingly viable choice for real-time control tasks.

To further enhance the performance of the MPC in real-life scenarios, ongoing research and development efforts focus on refining the cost function design, incorporating more advanced optimization techniques, and exploring the integration of machine learning and adaptive control methods. These endeavors aim to improve the MPC's overall performance, ensuring its continued relevance in diverse applications.

In conclusion, the experimental evaluation of the designed controllers underscored the distinct advantages of the model predictive controller over the proportional-Integral-Derivative controller. The MPC's ability to consider performance metrics, account for input constraints, and generate optimal control inputs in a computationally efficient manner positions it as the superior choice for trajectory control in safety-critical systems and applications that prioritize stability, accuracy, and user comfort. Continued research in this field promises to further enhance the capabilities of the MPC, opening up new avenues for its utilization in real-world scenarios.

4.5 Summary

The results chapter of this thesis has provided a comprehensive evaluation of the designed state estimation and controllers algorithm for the task of autonomous mobile material deposition. Through extensive experimentation and analysis in common deposition scenarios such as straight and L-shaped

path tracking, valuable insights have been gained into the performance of the proportional-integral-derivative (PID) controller and the model predictive controller (MPC).

Specifically, the model predictive controller achieves an average tracking error of 8 cm in the real scenario while following a right-turn path, with an average orientation error of 0.4 rad in the same scenario. Furthermore, the model predictive controller outperforms the PID controller in tracking a rectangular path, exhibiting an error that is 20 times less in the simulation environment for both lateral and orientation errors. In terms of critical control errors, such as over/undershoots, it is observed that the model predictive controller makes no such errors in the simulation environment, while the PID controller exhibits 2 overshoots and 1 undershoot.

The PID controller demonstrates a fast computation time, less than 1 ms for each iteration, outperforming the MPC counterpart, which takes 20 ms or more to find optimal solutions to the nonlinear programming problem.

The state estimation algorithm proves its capability to cope with inherent uncertainties and environmental noise, significantly contributing to the overall performance of the controllers. Its accurate estimation of the robot's pose plays a crucial role in achieving precise trajectory tracking.

Moreover, limitations of the PID controller are identified, including its agnostic nature towards system constraints and occasional deviations from the desired trajectory due to overshoots and undershoots. Conversely, the MPC controller demonstrates remarkable flexibility in its design, considering performance metrics and accommodating hard constraints on control inputs. This adaptability makes the MPC the preferred choice for safety-critical systems, showcasing optimal control input calculations for each time step and ensuring superior stability and smooth trajectory tracking.

Chapter 5

Conclusion and future work

In this work, a visual-based physically informed state estimation framework is proposed, in which depth estimation with object instance segmentation is incorporated to generate a dense 3D representation of the object using a stereo camera sensor. The quality of this representation is improved by performing 3D point cloud filtering. By utilizing prior knowledge of the dimensions of the target object, an optimal bounding box is estimated based on the normal distance of the points from the surfaces of the bounding box. Furthermore, physical kinematic constraints of the target object are taken into account in the optimization process to minimize pose estimation errors caused by inaccuracies in the depth estimation.

In addition, a motion model of the system is developed based on the piece-wise constant acceleration assumption. Uncertainty in the motion model and the visual estimation is characterized through empirical expressions, resulting in an algorithm that has a higher estimation accuracy. The state estimation algorithm is tested and quantitatively evaluated on fixed and mobile cold spray systems in a variety of scenarios.

As part of the work on visual navigation, a model predictive controller was developed to account for the physical kinematic constraints of the autonomous mobile material deposition agent during trajectory tracking. This controller utilized an objective function incorporating position and orientation error, control effort, terminal cost term, and the proximity term as the main tool for generating a smooth and feasible control signal, effectively driving the robot

on a desired path visually both in simulation and in real setups.

In future work, the focus will be on addressing challenges arising from occlusion and inaccuracies in depth estimation by incorporating onboard sensors such as onboard cameras and LiDARs, thus improving the accuracy of localization. The model predictive controller's efficiency can be enhanced by exploring novel optimization techniques or adaptive control strategies in the context of trajectory tracking. Implementing advanced methods for handling uncertainties and disturbances can boost the controller's ability to cope with real-world environmental complexities.

Lastly, exploring real-time optimization and control strategies could lead to improved visual navigation performance in dynamic and changing environments. The ability to make real-time adaptations to environmental variations and dynamic obstacles would be valuable in achieving efficient autonomous material deposition operations and could include the usage of intelligent collision avoidance modules in the framework, leading to a higher degree of robustness in the future.

References

(1) Zhang, J.; Yang, X.; Wang, W.; Guan, J.; Ding, L.; Lee, V. C. *Automation in Construction* **2023**, *146*, 104699.

(2) Bernardo, R.; Sousa, J. M.; Gonçalves, P. J. *Journal of Manufacturing Systems* **2022**, *65*, 339–350.

(3) Menendez, E.; Victores, J. G.; Montero, R.; Martínez, S.; Balaguer, C. *Automation in Construction* **2018**, *87*, 117–126.

(4) Camblor, B.; Salotti, J.-M.; Fage, C.; Daney, D. *Theoretical Issues in Ergonomics Science* **2022**, *23*, 60–79.

(5) Papadopoulou, A.; Kumar, N. S.; Vanhoestenberghe, A.; Francis, N. K. *British Journal of Surgery* **2022**, *109*, 921–932.

(6) Fisher, M.; Cardoso, R. C.; Collins, E. C.; Dadswell, C.; Dennis, L. A.; Dixon, C.; Farrell, M.; Ferrando, A.; Huang, X.; Jump, M., et al. *Robotics* **2021**, *10*, 67.

(7) AUTONOMOUS, SAFE AND ENVIRONMENTALLY FRIENDLY solution for ship surface treatment, `https://ambpr.com/en/ambpr-robotic-solution-for-sandblasting-and-painting/`, Accessed: 2010-09-30.

(8) Additive manufacturing and autonomous robotic printing, `https://www.ddsplm.com/blog/additive-manufacturing-and-autonomous-robotic-printing/`, Accessed: 2019-09-30.

(9) Bolot, R.; Deng, S.; Cai, Z.; Liao, H.; Montavon, G. *Journal of thermal spray technology* **2014**, *23*, 296–303.

(10) Herman, H.; Sampath, S.; McCune, R. *MRS bulletin* **2000**, *25*, 17–25.

(11) Ren, J.; Sun, Y.; Hui, J.; Ahmad, R.; Ma, Y. *Robotics and Computer-Integrated Manufacturing* **2023**, *83*, 102569.

(12) Nylén, P.; Edberg, M. In *ITSC 1997*, 1997, pp 583–592.

(13) Candel, A.; Gadow, R. *Surface and Coatings Technology* **2006**, *201*, 2065–2071.

(14) Hegels, D.; Wiederkehr, T.; Müller, H. *Robotics and Computer-Integrated Manufacturing* **2015**, *35*, 1–15.

(15) Mohan, L. J.; Ignatious, J. In *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR)*, 2018, pp 1–5.

(16) Ma, J.; Bajracharya, M.; Susca, S.; Matthies, L.; Malchano, M. *The International Journal of Robotics Research* **2016**, *35*, 631–653.

(17) Ravi Kiran, B.; Roldao, L.; Irastorza, B.; Verastegui, R.; Suss, S.; Yogamani, S.; Talpaert, V.; Lepoutre, A.; Trehard, G. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp 0–0.

(18) Jones, E.; Soatto, S. *Intl. J. of Robotics Res* **2011**, *6*.

(19) He, G.; Yuan, X.; Zhuang, Y.; Hu, H. *IEEE Transactions on Instrumentation and Measurement* **2020**, *70*, 1–9.

(20) Su, Y.; Wang, T.; Shao, S.; Yao, C.; Wang, Z. *Robotics and Autonomous Systems* **2021**, *140*, 103759.

(21) Shan, T.; Englot, B. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp 4758–4765.

(22) Cognetti, M.; Oriolo, G.; Peliti, P.; Rosa, L.; Stegagno, P. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp 350–356.

(23) Janković, N. V.; Ćirić, S. V.; Jovičić, N. S. In *2015 23rd Telecommunications Forum Telfor (TELFOR)*, 2015, pp 619–622.

(24) Shim, J. H.; Cho, Y. I. *Sensors* **2016**, *16*, 195.

(25) Babinec, A.; Jurišica, L.; Hubinskỳ, P.; Duchoň, F. *Procedia Engineering* **2014**, *96*, 1–9.

(26) Ramer, C.; Sessner, J.; Scholz, M.; Zhang, X.; Franke, J. In *2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2015, pp 65–70.

(27) Guo, F.; He, Y.; Guan, L. In *2017 IEEE global conference on signal and information processing (GlobalSIP)*, 2017, pp 408–412.

(28) Song, K.-T.; Chang, Y. C. In *2018 International Automatic Control Conference (CACS)*, 2018, pp 1–6.

(29) Hoyer, L.; Steup, C.; Mostaghim, S. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp 1388–1395.

(30) Feng, S.; Shen, S.; Huang, L.; Champion, A. C.; Yu, S.; Wu, C.; Zhang, Y. *Journal of Network and Computer Applications* **2019**, *146*, 102425.

(31) Flögel, D.; Bhatt, N. P.; Hashemi, E. *Robotics* **2022**, *11*, 82.

(32) Salimzadeh, A.; Bhatt, N. P.; Hashemi, E. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, 2022, pp 1124–1131.

(33) Hu, H.; Zhao, T.; Wang, Q.; Gao, F.; He, L. In *CICTP 2020*, 2020, pp 918–929.

(34) Chen, Y.; Tai, L.; Sun, K.; Li, M. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp 12093–12102.

(35) Chen, W.; Duan, J.; Basevi, H.; Chang, H. J.; Leonardis, A. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

(36) D'Innocente, A.; Carlucci, F. M.; Colosi, M.; Caputo, B. *CoRR* **2017**, *abs/1705.02139*.

(37) Oquab, M.; Bottou, L.; Laptev, I.; Sivic, J. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp 685–694.

(38) ACC Fontes, F. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* **2003**, *13*, 191–209.

(39) Abbas, M. A.; Milman, R.; Eklund, J. M. *Canadian journal of electrical and computer engineering* **2017**, *40*, 12–22.

(40) Aguiar, A. P.; Dačić, D. B.; Hespanha, J. P.; Kokotović, P. *IFAC Proceedings Volumes* **2004**, *37*, 167–172.

(41) Kanjanawanishkul, K.; Zell, A. In *2009 IEEE International Conference on Robotics and Automation*, 2009, pp 3341–3346.

(42) Ostafew, C. J.; Schoellig, A. P.; Barfoot, T. D.; Collier, J. *Journal of Field Robotics* **2016**, *33*, 133–152.

(43) Lim, H.; Kang, Y.; Kim, C.; Kim, J. *International journal of precision engineering and manufacturing* **2014**, *15*, 831–840.

(44) Ostafew, C. J.; Schoellig, A. P.; Barfoot, T. D. *The International Journal of Robotics Research* **2016**, *35*, 1547–1563.

(45) Hu, H.; Gu, D. In *IASTED International Conference on Intelligent Systems and Control, Santa Barbara, CA, USA*, 1999, pp 28–30.

(46) Donaire, A.; Romero, J. G.; Perez, T.; Ortega, R. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp 4385–4390.

(47) Elnagar, A.; Hussein, A. *Robotics and Autonomous systems* **2000**, *33*, 195–206.

(48) Kaliński, K. J.; Mazur, M. *Mechatronics* **2016**, *37*, 79–88.

(49) Poonawala, H. A.; Spong, M. W. In *2015 10th international workshop on robot motion and Control (RoMoCo)*, 2015, pp 97–102.

(50) Allgöwer, F.; Zheng, A., *Nonlinear model predictive control*; Birkhäuser: 2012; Vol. 26.

(51) Van Essen, H.; Nijmeijer, H. In *2001 European Control Conference (ECC)*, 2001, pp 1157–1162.

(52) Mehrez, M. W.; Mann, G. K.; Gosine, R. G. In *2013 16th International Conference on Advanced Robotics (ICAR)*, 2013, pp 1–6.

(53) Piovesan, J. L.; Tanner, H. G. In *2009 IEEE International Conference on Robotics and Automation*, 2009, pp 94–99.

(54) Hedjar, R.; Alsulaiman, M.; Almutib, K. In *2011 first international conference on robot, vision and signal processing*, 2011, pp 296–299.

(55) Backman, J.; Oksanen, T.; Visala, A. *IFAC Proceedings Volumes* **2010**, *43*, 133–138.

(56) Backman, J.; Oksanen, T.; Visala, A. *Computers and Electronics in Agriculture* **2012**, *82*, 32–43.

(57) Fruchard, M.; Allibert, G.; Courtial, E. In *2012 American Control Conference (ACC)*, 2012, pp 4149–4154.

(58) Backman, J.; Oksanen, T.; Visala, A. *IFAC Proceedings Volumes* **2013**, *46*, 35–40.

(59) Kanjanawanishkul, K.; Zell, A. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp 2771–2776.

(60) Nascimento, T. P.; Conceiçao, A. G.; Moreira, A. P. *Robotica* **2016**, *34*, 549–567.

(61) Wang, Z.; Kinugawa, J.; Wang, H.; Kazuhiro, K. In *2015 IEEE international conference on robotics and biomimetics (ROBIO)*, 2015, pp 415–422.

(62) Mehrez, M. W.; Mann, G. K.; Gosine, R. G. In *2015 Moratuwa Engineering Research Conference (MERCon)*, 2015, pp 130–135.

(63) Abbas, M. A.; Milman, R.; Eklund, J. M. *Canadian journal of electrical and computer engineering* **2017**, *40*, 12–22.

(64) Mayne, D. Q.; Rawlings, J. B.; Rao, C. V.; Scokaert, P. O. *Automatica* **2000**, *36*, 789–814.

(65) Fontes, F. A. *Systems Control Letters* **2001**, *42*, 127–143.

(66) Gu, D.; Hu, H. *IEEE Transactions on Robotics* **2005**, *21*, 1022–1028.

(67) Gu, D.; Hu, H. *IEEE Transactions on Control Systems Technology* **2006**, *14*, 743–749.

(68) YANG, T.-T.; LIU, Z.-Y.; CHEN, H.; PEI, R. *Acta Automatica Sinica* **2008**, *34*, 588–593.

(69) Nazir, A.; Wani, M. A. In *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2023, pp 1088–1095.

(70) Du, W.; Xiang, Z.; Chen, S.; Qiao, C.; Chen, Y.; Bai, T. Real-time Instance Segmentation with Discriminative Orientation Maps, 2021.

(71) Setyawan, R.; Sunoko, R.; Choiron, M.; Mudjirahardjo, P. *Indonesian Journal of Electrical Engineering and Computer Science* **2018**, *12*, 585–591.

(72) Suga, A.; Fukuda, K.; Takiguchi, T.; Ariki, Y. In *2008 19th International Conference on Pattern Recognition*, 2008, pp 1–4.

(73) Bangunharcana, A.; Cho, J. W.; Lee, S.; Kweon, I. S.; Kim, K.-S.; Kim, S. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp 3542–3548.

(74) Zhang, K.; Chermprayong, P.; Xiao, F.; Tzoumanikas, D.; Dams, B.; Kay, S.; Kocer, B. B.; Burns, A.; Orr, L.; Choi, C., et al. *Nature* **2022**, *609*, 709–717.

(75) Liu, Y.; Wang, L.; Brandt, M. *The International Journal of Advanced Manufacturing Technology* **2019**, *105*, 1055–1067.

(76) Borrelli, F.; Bemporad, A.; Morari, M., *Predictive Control for Linear and Hybrid Systems*; Cambridge University Press: 2017.