Control of Sensorless Articulated Manipulators using Vision-based Estimation Techniques

by

Charanjot Singh

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

Master of Science

Department of Mechanical Engineering
University of Alberta

Abstract

Conventional control of an articulated manipulator such as robot arms involves the use of sensor measurements of joint values to calculate the position and orientation of the end-effector and to perform motion control. Conversely, in cases where direct sensing is not available, a vision-based method can prove to be advantageous. This thesis deals with one such system, an excavator used regularly at construction sites, with the long-term goal of enabling fully autonomous operation. For our hardware experiments, we used a scale excavator model inside a lab setting, using a single external monocular camera to estimate the joint states of the vehicle in real-time and using the resulting information to run motion planning.

Our solution employs a computer vision algorithm integrated with numerical optimization to perform 2D and 3D pose estimation, respectively, from single RGB image frames in real-time. We use a limited collection of real images for training, in contrast to other approaches presented in the literature which require a CAD model to generate a large set of synthetic data. We leverage geometric constraints of the manipulator and the camera's optical parameters within our iterative optimization step. We also build a custom microcontroller-based system to drive the scale excavator from a desktop computer. The experimental results are benchmarked against a high-fidelity optical motion capture system installed in our lab to quantify the accuracy of our estimates. Since our proposed method relies only on a small training dataset and geometric information about the manipulator, it can readily be ported onto larger manipulator systems, such as large industrial excavators or cranes found on real construction sites.

Preface

This thesis is an original work by Charanjot Singh. No part of this thesis has been previously published.

Acknowledgements

I would like to thank Dr. Martin Barczyk and Dr. Alireza Bayat for their continuous support, supervision, and guidance throughout the course of this research work. I am also grateful to Aravindh Ganesan and Lucas Gandrey for helping with the data collection and initial work with the development of the system in the summer of 2022; Dr. Younes Al-Younes, Amir Ebrahimnezhad, Ivy Friesen and Shane Allan for productive discussions throughout my graduate work. Finally, I would like to thank my family for their constant support and love.

Table of Contents

List of Tables

List of Figures

Abbreviations

CAD Computer Aided Design.

CNN Convolutional Neural Networks.

DFT Discrete Fourier Transform.

DNN Deep Neural Network.

DOF Degree-of-freedom.

FFT Fast Fourier Transform.

FPN Feature Pyramid Network.

GPS Global Positioning System.

IMU Inertial Measurement Unit.

MAE Mean Absolute Error.

OKS Object Keypoint Similarity.

PCB Printed Circuit Board.

PCK Percentage of Correct Keypoints.

PID Proportional Integral Derivative.

POE Product of Exponentials.

PWM Pulse Width Modulation.

ROS Robot Operating System.

RPN Region Proposal Network.

SDK Software Development Kit.

SVM Support Vector Machine.

URDF Unified Robotics Description Format.

UWB Ultra-Wide Band.

YOLO You Only Look Once.

Chapter 1

Introduction

1.1 Background

Robot manipulators have been extensively used in industrial settings for many years. These manipulator arms are used to automate simple tasks such as pick and place operations to perform more complex applications such as robotic welding, which require high precision. The key to their path accuracy is the availability of precise joint angle measurements using onboard encoders/rotary transducers to solve the inverse kinematics problem in these automation tasks.

For calculations, these arm joints are assumed to be rigid links and ignore any flexibility in the drivetrain or the joint which is generally not the case [1]. Further, this problem is compounded due to the presence of backlash in motors [2] and magnetic deviations in measurements [3] resulting in further diversion from the true joint state. These problems restrict the use of some less expensive and flexible robots for use in non-industrial or household settings due to the presence of even larger errors.

A different problem arises for inexpensive encoder-less manipulators which do not have any feedback system available or manipulators which are controlled directly by a person (such as construction equipment). A common practice in this case, would be to use a precision encoder system or an inertial measurement unit (IMU) sensor by making alterations to the physical system causing a lot of downtime and reconfiguring. To automate these machines efficiently the requirement of an external

joint state estimation system becomes even more imperative.

This is where computer vision can provide a solution. With the continuous advancement in computation power in recent years, vision-based neural networks are becoming common for complex applications in object detection [4], autonomous navigation [5] and face recognition [6]. There has been a growing interest to apply similar techniques in robotic systems as well. Recent works in [7–9] have established the use of different vision algorithms for the estimation of a robot manipulator using an externally mounted RGB camera to detect and localize the position. This thesis takes inspiration from these recent works and uses external camera-based sensing to estimate the joint configuration of a robot manipulator for position control of the arm.

1.1.1 Motivation of Research

The recent shift to automation in residential settings has paved way for the development of affordable and compact robotic systems. However, for these systems to be easily operable, they must possess the ability to comprehend their surroundings and current position for collaboration with other systems or for human-robot collaboration. Even performing a relatively simple motion control requires knowledge of the current position of the robotic system and a feedback control system to enable the system to reach the desired final position accurately.

On the other hand, manually operated heavy equipment, such as industrial excavators and cranes, differ significantly from robotic systems as they are externally controlled or required direct operation. In these systems, the operator relies on visual observations and manually moves the joystick to actuate the onboard motors to move to the final position. This reliance on human operators to control the equipment restricts their use in environments not safe for human operation or hazardous environments such as mine sites and nuclear reactors.

An external camera in this case can serve as the visual sensor to capture real-time

images which can be processed by machine learning algorithms to extract the pose of the robot manipulator. Moreover, vision-based sensing systems also facilitate the development of more intuitive human-robot interfaces, allowing users to interact with the robotic systems using natural gestures as in [10].

Previous related work on this topic at the University of Alberta's Mechatronic Systems Lab was performed in [11, 12]. These studies explored the detection problem in depth for joint angle computation of an industrial Baxter robot (hardware overview in Section 2.1.1.1). This thesis is a step further in the direction and addresses some of the shortcomings of the previous work. It tries to answer a few questions which arise in the current studies:

- Can the detection model work with a small training set while limiting any adverse effects (such as overfitting)?
- How do we train a model where a CAD model of the manipulator is non-existent?
- What is the required number of keypoints for precise pose estimation?
- How to develop a real-time control for the system, as many of the current methods are limited to offline processing only?
- How to obtain a transferable method that can be applied to different robotic systems?
- Can the pipeline be integrated into path-planning applications?

1.1.2 System Overview

Our aim here is to design a vision-based feedback control system for an encoder-free robot manipulator to be used in real-time applications. We propose a structured modular methodology as presented in Figure 1.1 consisting of three modules. Here the main technical challenge is to build a keypoint detection model and then estimate

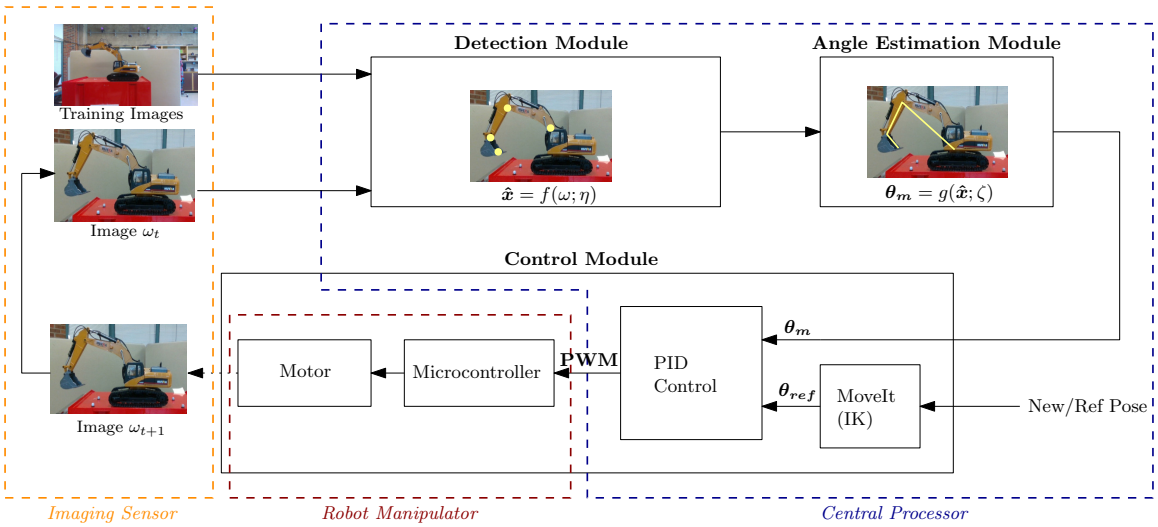


Figure 1.1: An overview of our entire system in a module based structure. Solid black boxes represent the successive stage of data computation and information flow, colored dashed boxes show the physical hardware used.

the joint angles from the extracted keypoints, all using information from a monocular camera image. The first component, the *keypoint detection module* consists of a neural network that computes the keypoint pixel locations, $\hat{\boldsymbol{x}}$, from a single input RGB image frame ω, given η, the network weights of the model found while training using the training images, mathematically $\hat{\boldsymbol{x}} = f(\omega;\eta)$. The next module is for *pose estimation* using keypoint information to solve an iterative optimization problem using known camera parameters, ζ, to compute the measured joint angles, $\boldsymbol{\theta_m}$. The final section is the *controller* that receives a reference trajectory information, converts it to reference angles, $\boldsymbol{\theta_{ref}}$, using an inverse kinematic solver, and compares it with the measured angles to obtain a $\boldsymbol{PWM}$ signal for motor actuation of the manipulator. The camera then provides another image ω_{t+1} and the system runs in a loop for autonomous control of the system. We refer to this framework as *VECTOR* (for Vision-based Estimation and ConTrol Of Robot manipulator).

For the detection and angle estimation stage, the Baxter robot is used as an initial setup for pipeline verification and comparison purpose, while the entire thesis is heavily focused on the use of these methods for the control of a scale excavator, used

in all the sections; a comparative analysis is presented wherever possible.

1.2 Literature Review

Following is a review of relevant works in detection and estimation with an emphasis on application in the construction industry. Detailed background on the neural network structure used in our work is provided in Chapter 3 and is not included here to keep the discussion brief.

1.2.1 Vision-based sensing and detection

Over the last decade, due to the availability of constantly increasing computational power, the exponential growth of Computer Vision in research has been observed. One of the earliest convolution neural networks (CNN) in computer vision use was in text recognition [13] which now has expanded to generate multimodel text caption for images using an m-RCNN [14] or text classification using CNN [15]. These networks are now used even in medicine for X-ray image classification [16], sports activity recognition [17], and also real-time defect detection in fabrics [18, 19].

In an industrial setting, vision methods have been used in task recognition and assessment of the job site utilizing a multi-camera system [20, 21]. [22–24] discuss vision-based sensing in transportation for object recognition and detection on streets and highways, with a multi-vision sensor for mapping proposed in [25].

The integration of computer vision in robotics and automation relies on advanced human-like sensing for tasks like object detection and tracking [26–29], depth estimation from monocular camera [30, 31], face recognition from images [6, 32, 33] and more recently autonomous detection and grasp [34] or controlling by hand motion imitation [10].

1.2.2 Keypoint detection

Object detection has been studied extensively in rigid bodies of known geometries [35–39] and can be extended to keypoint detection. Many initial pose detection methods used fiducial markers [40] or depth estimation [41]. Further, sensor fusion methods [42, 43] were explored to achieve better performance. The use of vision-based pose estimation gained traction for the application of object localization with cameras onboard a robot system [4, 44–46]. The problem we try to address here is a bit different as we try to estimate a robot manipulator in motion which makes it a non-rigid body as a whole and direct estimation methods do not work effectively in these cases.

Therefore, we make use of a keypoint detection pipeline to extract the pose. Keypoints are user-defined point locations that are of specific interest in a detected scene. One of the initial developments in this domain was for human keypoint detection for pose estimation [47–49]. For our work, keypoints are points on the manipulator which can be used to compute the current position of each link, generally taken as joint centers.

The previous state-of-the-art works on a robotic system for 2D keypoint detection are limited mainly to DREAM (Deep Robot-to-camera Extrinsics for Articulate Manipulators) [7] and CRAVES (Controlling Robotic Arm with Vision-based Economic System) [8]. Both of these methods use a stacked hourglass network for the detection of keypoints. DREAM uses the joint positions as the keypoints and trains the network on different industrial robots while CRAVES uses a set of 17 keypoints for measuring the joint pose of an OWI-535 robot (an inexpensive robot with no joint encoders). Recently, a different method using render and compare instead of keypoints has also been proposed in RoboPose [9] for joint estimation. Though having a few differences, all of these studies rely on a CAD model for the robot and a large training dataset of synthetic images. To the best of our knowledge, the excavator system discussed in

our study has never been used in any previous work for keypoint detection or pose estimation.

1.2.3 Pose estimation techniques in construction

Knowledge of an articulate robot movement can significantly enhance human-robot interactions and simplify safety evaluation at construction sites. Classical image-processing systems have proven beneficial for improving safety, productivity, and general monitoring but are still not prominent in the construction industry [50]. Previous studies have employed computer vision techniques for hard-hat detection [51], hand protection detection [52], worker-equipment interaction [53, 54] and equipment pose forecasting [55], to provide and enforce a safe operational environment. Additionally, they have also been used to assess the productivity of particular equipment [56] or the entire job site using multi-camera systems [57]. [58] provides a summary of various studies using image data in construction and equipment tracking.

To estimate the pose and location of construction equipment, two possible techniques can be applied; sensor-based or vision-based. For sensor-based methods, sensors such as Inertial Motion Unit (IMU), Global Positioning System (GPS), Ultra-Wide Band (UWB), etc. are deployed onboard the physical system. All of these have significant drawbacks in an urban setting or have large drift after repeated use [59].

On the other hand, vision-based methods analyze the pose from an externally mounted image-sensing system. These systems are further categorized into two branches, marker-based and marker-less estimation methods. Marker-based techniques estimate the pose using fiducial markers mounted directly on the equipment [60, 61], similar to the sensor-based method. [62] achieves an average error of less than 8.5° for orientation estimation using CALTag markers mounted at multiple locations on an excavator.

Marker-less vision-only methods directly extract image features for the detection and estimation of the pose of these manipulators. Many direct pose estimation ap-

proaches use geometrical information and conventional computer vision methods, such as skeleton detection [63] and key node tracking using a stereo camera [64] for excavator pose estimation. These methods heavily rely on specific shapes and viewpoints, therefore cannot be generalized in complex environmental settings [63].

More advanced methods leverage machine learning models to solve the detection problem. [65] proposes an R-FCN (Region-based Fully Convolutional Network) for bounding box detection and localization of construction equipment. A simplistic vision-based grading control in a simulated environment using an artificial neural network with 3 layers to estimate an excavator's cylinder displacements is presented in [66]. Keypoint detection neural network structures for excavators include a regression-based deep neural network for keypoint detection [67], a comparison of stacked hourglass network, cascade pyramid network, and an ensemble version of the two [68], a unique recurrent neural network [55], a YOLOv5 network for excavator detection and a SimpleBaseline [69] network with Resnet backbone for keypoint [70], etc. are just a few of the possible many variations.

[71] utilizes a stacked hourglass network to compute the 3D pose of a robot arm with an attached bucket in a lab setting to resemble an excavator and employs a fast dataset collection technique that uses encoder data from the robotic arm to generate the labeled dataset. [72] uses a DREAM [7]-like method to generate synthetic images using domain randomization for training an HRNet [73] architecture and testing accuracy on real images for keypoint detection of an excavator. Both these studies [71, 72] acknowledge the sparse image dataset in a construction setting and try to address the concern using different methods, many of these methods rely on the availability of a highly realistic CAD model to obtain synthetic images.

Furthermore, many deep CNN structures require expensive computations while training and have slow operational speeds. Sparse constrained, compact CNN networks with fewer parameters are more recently studied [74] and applied in cases where a limited image dataset is available for construction equipment [75, 76].

Our work is a step forward towards autonomous excavator detection and control using a small training dataset and relying only on the physical dimensions of the excavator, such that a CAD model of the system is not required.

1.3 Thesis Outline

This chapter provides a background on the robot manipulator pose estimation problem and the motivation to perform the research outlined, along with the questions this thesis tries to answer. Further, a high-level overview of our system is provided with a comprehensive literature study of related works in the area.

Chapter 2 lists the hardware and software used in the study, including the robot manipulators, imaging sensors, and the custom-built ROS packages to successfully run our network in real-time applications.

Chapter 3 covers the foundations of vision-based neural networks and their applications in solving complex problems in detection and localization. Following this, we present the keypoint detection CNN model architecture and its training results on the Baxter and scale excavator. A benchmark comparison with previous works is provided for the Baxter on an online detection test run where both models run in parallel, while only visual qualitative results are presented for the excavator at this stage.

Chapter 4 provides mathematical preliminaries for the conversion of 2D keypoints to 3D poses using joint angle estimations. Using camera calibration, intrinsic and extrinsic parameters are obtained and applied in solving the kinematics using a Product of Exponentials-based formulation in the optimization step. Multiple optimizer equations are proposed and the best-performing one is selected based on merit. Finally, the joint angles are compared to ground truth angles obtained from encoders (for the Baxter) and an optical motion capture system installed in the lab (for the excavator).

Chapter 5 develops a simple digital PID algorithm for joint control of the excavator and integrates the entire pipeline with the MoveIt package for path-planning capabil-

ities. It also includes a signal filter design for our measured joint angles. The entire integration with ROS and MoveIt enforces the generalization to other manipulators, a claim presented in the thesis. Experimental verification of the entire framework is tested using random target positions in the workspace and the findings are reported in Section 5.5.1.

Chapter 6 summarizes the work performed in the thesis and the results obtained. Possible future directions along with limitations of the current study are discussed.

1.3.1 Statement of Contributions

This thesis claims the following research contributions (listed in the order of appearance):

- Demonstrate the feasibility of using a minimal training dataset to achieve competitive keypoint detection on two different robot manipulators, a Baxter robot arm and a scale excavator, whereas most existing methods require a very large collection of synthetic images using a CAD model for training.

- A Product of Exponentials-based optimization algorithm to compute joint angles from 2D keypoint information of a known robot geometry and camera parameters.

- Real-time computation and implementation of joint estimation framework on both manipulators with a comprehensive assessment of our work with recent benchmark studies.

- Development of a custom URDF (Unified Robotics Description Format) for the scale excavator used within the optimization problem.

- Integration of developed vision-based joint state estimates into a digital PID feedback controller and a path planning algorithm for precise point-to-point position control of the end-effector.

Chapter 2

Hardware and Software Architecture

2.1 Hardware Background

This section categorizes all the hardware components used in this thesis along with the specifications of the system and their precise setup.

2.1.1 Robot Manipulator

2.1.1.1 Baxter Robot

The Baxter is a human-size industrial robot developed by Rethink Robotics Inc. referred to as the 'Baxter robot' or 'Baxter' in the thesis. It has two arm manipulators, each of which is a 7-DOF (degree of freedom) system. The robot has two end-effector variations: a vacuum and an electric gripper; an external air unit powers the vacuum one, while the robot itself powers the latter. We only use the electrical gripper for this research to perform initial pick and place tests. In the later stage, the gripper used becomes irrelevant as it is not controlled/actuated.

The Baxter comes with an onboard display that serves as the face. It also has multiple sensors: sonar on the head, precise encoders on each revolute joint, and imaging sensors on the face (camera) and cuffs (camera and IR sensor). Out of the above list, we only use the encoder joint angle values, which have an average position accuracy of $\pm 0.1^{\circ}$, with a maximum variation of $\pm 0.25^{\circ}$ [77]. The robot comes with

a laptop PC with a Linux operating system having all the required Baxter SDK dependencies for ROS. This external laptop is connected over the Ethernet to the main computer on board the Baxter.

Figure 2.1: Baxter robot with arms in free position without joint actuation

Further, as each of Baxter's arms is a 7-DOF system where each arm is built up of a set of 7 isolated revolute joints, making it an *RRRRRRR* system with a kinematic redundancy (as the robot DOF is greater than the 3D workspace). This makes it possible to have multiple configurations with the same end-effector position. Also, as described in [78], the Baxter has a complex geometry with multiple joint offsets

making the existence of an analytic solution for the inverse kinematic problem almost non-possible. As a result of this and other limitations, some joints are considered rigid/fixed for the current research (details in Section 3.4.2).

2.1.1.2 Scale Excavator

The second robot arm used in this research is a 1:14 scale model of an industrial excavator. This Huina 1580[1] excavator model has full operational capabilities using an RC system working on a 2.4GHz transmitter, such that the excavator can be controlled from distances up to 50m away, it is referred as the 'excavator' or 'scale excavator' throughout the thesis. This model is a 7-DOF system (4-DOF arm and 3-DOF planar motion of base).

Figure 2.2: Huina 1580 1:14 scale model of industrial excavator

The model uses standard DC motors for all joints connected to the central PCB. These DC motors then move the joint directly (in case of the base) or indirectly using a screw mechanism to provide linear motion (in case of the arm, wrist and bucket). As the system uses standard DC motors, we do not have any feedback from

[1]https://huinaconstructiontoys.com/products/rc-huina-1580-excavator

the excavator with regard to its current position. Moreover, as discussed later in Section 3.4.2, only the arm is moved while keeping the excavator rollers stationary. This makes the reduced system a 4-DOF arm manipulator on a fixed base similar to the Baxter robot.

2.1.2 Imaging Devices

For the research, we use two kinds of imaging devices, a single camera facing straight onto the robot and another set of cameras to provide the ground truth values of the joint estimates.

2.1.2.1 RGB-D Camera

The Intel® RealSense™ D435i RGB-D camera is a commonly available off-the-shelf camera used as the primary camera for this research. It provides two different kinds of images; one a standard RGB image and the second a grayscale depth image using the stereo cameras.

Figure 2.3: RealSense D435i RGB-D Camera with mounting stand and 3D printed support

It should be noted that the stereo camera component is not used in any part of this thesis. We only use the RGB image obtained from the camera. The sole purpose of this specific RGB-D camera is its simple SDK package support for ROS, making it easy to integrate the live camera feed for use in our system. The RGB image obtained

has a frame resolution of 1280 X 720 (2 Megapixels) with a 30fps frame rate.

2.1.2.2 Vicon Tracking System

Motion capture systems are tools to dynamically estimate a specific body's position and orientation using reflective markers and calibrated cameras. This is widely used for real-time estimation of human movement in movies for CGI effects, sports, and healthcare applications [79].

Figure 2.4: Vicon Vera Camera with mounting in the lab setting

Motion capture systems come in various types; here, we use a Vicon Vero v2.2 camera-based system with ten infrared (IR) cameras mounted in the lab, used to triangulate the 3D position of a retroreflective marker attached to a body with respect to a pre-set custom origin. This system is used to compute the accurate placement of the robot arm base wrt the camera and also to provide precise joint state values for validation of our computer vision system.

The Vicon system in the lab is connected to a stand-alone computer running the Tracker 3 software to compute the position information. Each marker must be in the field of view of at least three cameras for triangulation of the position. These cameras can give near real-time estimates with an average latency of around 3.6 ms [80].

Figure 2.5: Retroreflective markers for motion capture system

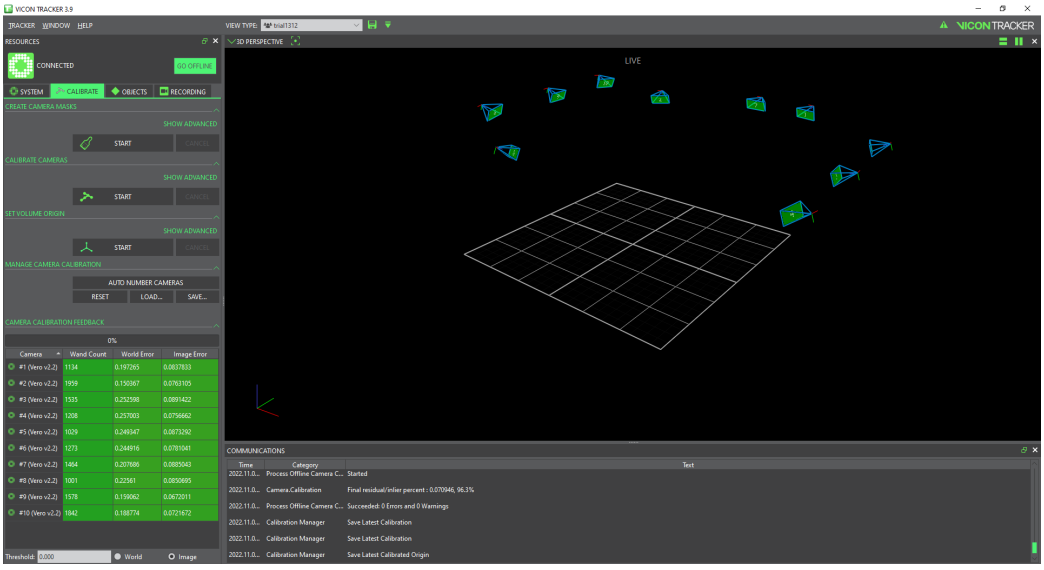


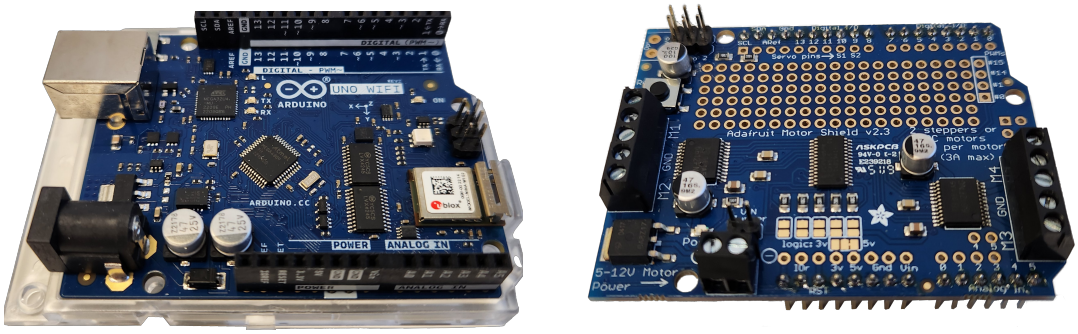
Figure 2.6: Tracker3 software running on a stand-alone desktop connected to the Vicon Vera cameras in the lab. The left bottom (in green) shows the successful calibration of all the cameras

The cameras need to be calibrated before use. This process involves using a calibration wand which is moved around the volumetric space under observation. Each camera then detects at least 1500 frames of the wand to compute the transformation of each camera with respect to the others, and the software will provide the image

error and the world error, which should be around 0.5mm. If the error is significant, the calibration needs to be repeated. Further, the software prompts the selection of the world origin, which is done by placing the wand at a specific location marking the center of its cross the origin in this case. After this, the calibration is complete. If the camera is bumped or moved, the camera will turn red, and the calibration process must be repeated.

2.1.3 Microprocessor

To integrate the scale excavator with ROS (discussed in Section 5.3), the controller onboard the excavator is changed to an Arduino Uno Wifi Rev.2 with an Adafruit Motor shield V2. This setup provided the operation and control of the excavator's arm through the central PC instead of the remote controller.



(a) Arduino Uno Wifi Rev.2 Board (b) Adafruit Motor Shield V2 board

Figure 2.7: Microprocessor Components

The Arduino board is connected to the central system using Arduino IDE, as discussed later in Section 2.2.5. This Arduino board connects to the motor shield, which drives the four motors on the excavator's arm.

2.1.4 Central Processor

With extensive computation power to integrate all communication from the microprocessor, robot manipulation and image sensors to the vision algorithm, a desktop PC is used as the main central system. This PC is equipped with 64 GB RAM, an 8

core Intel® Core™ i9-9900K Processor CPU (Central Processing Unit) running at a base frequency of 3.60GHz and an NVIDIA GeForce RTX 2070 Super GPU (Graphic Processing Unit) enabling parallel computing of the neural network algorithm. The system runs a Linux Ubuntu 18.04.6 LTS operating system.

2.2 Software Background

The following section provides an overview of the software used throughout the research to analyze and integrate data collection and delivery between the hardware components and the machine learning algorithm.

2.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) provides a cross-platform library in C/C++ and Python for real-time digital image analysis, processing and feature recognition. The library was initially developed by Intel and lays a foundation for further machine learning technologies available now for computer vision problems. This was used extensively for image visualization, overlapping and camera calibration process throughout the research.

2.2.2 CUDA

NVIDIA CUDA is a platform providing a parallel computing environment and additional APIs (application processing units) for software to select a certain GPU. This platform is used to train and run neural networks (Section 3.4.2) along with their real-time deployment (Section 3.5).

2.2.3 ROS

Robot Operating System or ROS is an open-source platform that provides exceptional abilities to operate and program a robot. The name might mislead into the belief that it is an alternative operating system (OS) whereas it is an add-on system working

alongside the traditional OS. It has the ability to run across multiple machines along with a combination of different hardware attachments and multiple robots.

ROS provides a simple infrastructure where everything is linked to a central `roscore` having the required communication information for bridging the communication across platforms. The system consists of a set of *nodes* publishing data through unique transmission carriers called *topics* in its program code. The platform is set in a way that any node can subscribe to any published/to-be-published topic containing real-time data. This makes it possible to break down a massive program into small chunks of code that can be used repeatedly and independently much more efficiently than traditional methods. Also, cross-platform abilities allow the data published by one node accessible by any other node irrespective of the language (say C++ with Python) used. All ROS programs are compiled under a specific *package* containing the build and execution files. Multiple ROS packages used throughout the research are listed below.

2.2.3.1 `vicon_bridge`

The `vicon_bridge` package is a ROS wrapper used to broadcast data from the Tracker 3 software connected to the Vicon system. It publishes the location and orientation of rigid bodies as defined in the tracker software along with each marker position.

The package can be run on any machine connected to the same network making the marker information available across various machines. The package runs on the central system receiving data from the stand-alone PC running the Tracker 3 software. Further, only the marker locations in 3D (w.r.t. the origin) are used/ recorded for the entire thesis and no other information is collected from the system.

2.2.3.2 `baxter`

The `baxter` SDK provides all the basic platforms for the development and use of the Baxter robot. The most common file used is `baxter.sh` used to initialize the Baxter

robot and establish a connection between the robot master system with the laptop PC.

2.2.3.3 `baxter_tools`

The `baxter_tools` repository provides the basic operational and maintenance tools for the Baxter, such as calibration protocol for the arms, enabling the robot to establish further communication with other tools to run the Baxter.

2.2.3.4 `baxter_common`

The `baxter_common` repository provides all the URDF and mesh files that are required to simulate the Baxter in a virtual environment and to replicate the movement and path planning from the simulated environment to the physical Baxter robot. The URDF describes the position, orientation and type of each joint and the mesh file provides the 3D representation of each of the robot links.

2.2.3.5 `baxter_examples`

The `baxter_examples` repository provides an extensive list of programs to run the Baxter robot in different ways; like using a
joystick (`joint_position_joystick.py`),
keyboard (`joint_position_keyboard.py`) or
waypoints (`joint_position_waypoints.py`);
record and replay a manual trajectory (`joint_trajectory_file_playback.py`); or other different aspects of its motion.

The trajectory feedback is used for initial testing and keyboard control is used extensively during the data collection for the ML model.

2.2.3.6 MoveIt

MoveIt is a robot manipulation framework for ROS. It provides a repository to incorporate motion planning, manipulation, inverse kinematics, control, 3D perception,

and collision checking under one big structure.

The basic task considered in the design of MoveIt is the control of a robot manipulator from a given configuration to a goal state while taking note of all the physical and dynamic constraints for the motion [81]. MoveIt has been successfully used in various industrial robots and has a pre-built repository for the Baxter robot, making path planning for the Baxter robot very convenient. MoveIt uses a URDF file for the robot to define the joint type and location and a mesh file that has a 3D model of all the links. The interactive interface Rviz is used to define planning scenes and to show the path planned with object avoidance and self-collision detection if required. The thesis describes the use of MoveIt for our work in detail in Section 5.3.

Following is the list of primary packages developed by the author and maintained by Dr. Martin Barczyk under the Mechatronic Systems Lab GitLab repository,

2.2.3.7 `keypoint_rcnn`

The `keypoint_rcnn` package includes a comprehensive list of Python scrips for training the PyTorch model, as well as the code for deploying the system in real-time using the camera feed.

2.2.3.8 `rscam_calib`

The camera calibration package is the repository for all the calibration steps presented in our work. Using this package we find the intrinsic and extrinsic parameters of the camera with respect to the base of the robot manipulator used in the optimizer algorithm.

2.2.3.9 `vicon_angle`

This repository provides the fundamental tools to compute the ground truth angles for the excavator and the joint angle based controller design, with the following essential scripts:

- `angle_vicon_cal.py`: This Python script is a set of functions that uses the Vicon markers position in 3D space to find the angle between lines and planes for computation of highly precise joint angle for the excavator.

- `exc_control.py`: This Python script is the digital controller of our system that takes the reference and estimated joint angles as input to compute the output signal to the motors on board the excavator.

2.2.3.10 `mmm_ros`

The repository is the keyboard motion control for the excavator where a set of keys are used to actuate the robot arm. `teleop_control.py` is the main script for the actuation with different speeds and joints to move the excavator links randomly, this code is used in training and assessment of the keypoint detection model.

2.2.3.11 `angle_estimate`

The is the package constituting the C++ scripts and libraries for the iterative joint angle optimization algorithm using the LM optimizer (Section 4.3.1). This package consists solely of C++ scripts for real-time execution with minimal delay.

2.2.3.12 `exc_arm_config`

This package is generated using the MoveIt setup wizard for control of the excavator inside the MoveIt environment.

All these packages discussed are linked together using ROS *subscribers* and *publishers* for real-time data transfer.

2.2.4 MATLAB

MATLAB is a numeric computing software and programming language developed by MathWorks. It serves as a powerful tool for complex mathematical problems, such as matrix computations, linear algebra, optimization, and signal processing, while also

providing a range of functions for data analysis, visualization, and simulation. Matlab also has a large library of toolboxes that make its use even further in the domains of robotics and machine learning. One other advantage is its ability to interact with other programming languages like C++ or Python.

Matlab is used here to perform Fourier transforms and to build the signal filter and the Robotics toolbox is used to verify the inverse kinematic equations for the optimization problem.

2.2.5 Arduino IDE

Arduino IDE (Integrated Development Environment) is a software tool used to write and upload code to Arduino boards and many different microcontrollers. It is an open-source platform based on very simple hardware and is designed to control simple electronics. Further, the IDE can auto-detect the connected board and its serial port. We use a custom-made `exc_control_test.ino` script to bridge the data from the central processor to the onboard Arduino Uno Wifi Rev.2 connected to the excavator's motors.

2.2.6 SOLIDWORKS

SOLIDWORKS is a 3D CAD software by Dassault Systèmes used commonly to model, simulate and extract technical drawings from solid models. It is feature-based design software that helps create models based on the shape and size of each part which can then be combined with other individual parts to form a complex assembly. Another advantage to SOLIDWORKS is the scale-back feature which allows one to scale back the features in a model and make changes to a previous version without affecting and subsequent feature utilizations.

Here we use SOLIDWORKS only to modify an existing CAD model and then use the SW2URDF plugin to build and export the URDF file for the scale excavator with the required joint positions and types.

Chapter 3

Vision-Based Detection Model

3.1 Overview

This chapter covers the foundation of pose estimation and the development of a machine learning based neural network for detection of the robot manipulator in the camera frame. First, we explore the advancements in computer vision algorithms that use deep learning platforms and models. Subsequently, a brief description is made regarding the various neural network structures used in the computer vision problem and how they all link to form the final model used. Finally, the model is tested in real-time and extensive comparison is provided with other relevant works.

3.2 Machine Learning

Machine Learning (ML) is a broad subfield under the umbrella term of Artificial Intelligence (AI). It explores the various ways for computer systems to process, analyze and extract relevant information from data. Today, ML systems are used for extracting features from an image, automation of voice-to-text systems, and providing content suggestions online [82]. Conventional algorithms to obtain similar results require extensive engineering background and complex optimizations, resulting in substantial expertise and time commitment to a single problem. The ML algorithms are classified into three categories based on the input information to the algorithm: supervised, semisupervised, and unsupervised learning. This research focuses on the

use of supervised learning; learning based on correctly labeled information.

With the advancement of computation power and data availability ML has seen significant advancements over recent years. This paved way for the development of Deep Learning platforms that use deep neural networks (DNN) with multiple layers of neurons to automatically detect the "representations" for the classification or detection problem [82].

3.2.1 Machine Learning Frameworks

3.2.2 TensorFlow

TensorFlow provides an open-source library developed by Google Brain for the building and training of machine learning models. It provides a high-level system with implementation details that allow for a simplified model design. Further, it provides a wide range of tools for deploying, visualizing, and evaluating models.

3.2.3 Pytorch

PyTorch is also an open-source library for machine learning developed by Facebook AI. PyTorch provides a more Python-based approach using the torch library for model development and it also comes with a wide range of pre-trained models. For this thesis, PyTorch is used for all deep learning model designs for its easier use and prebuilt backbone structures for the keypoint detection problem.

3.3 Deep Neural Network

3.3.1 Convolutional Neural Networks

The development of CNN started with the introduction of LeNet [13] in 1998 for document recognition specifically for handwritten characters. This is when the power of CNN was realized for image processing and detection. But due to the limited computation power available at that time to perform successive convolutions quickly, the use only became prominent with the development of AlexNet [83] in 2012.

AlexNet image classification network consisted of 5 convolution layers with 60 million parameters and achieved much better error rates compared to the previous state-of-the-art networks available during the time. With such an enormous network that may result in overfitting, two solutions were proposed to overcome the problem, data augmentation and dropout layers [84]. Where "dropout" refers to the setting of a neuron's weight to zero which has a probability of 0.5, this way each neuron was trained more robustly and reduced system complexities [83]. This laid the foundation for the development of more deep and complicated convoluted neural networks.

Here we discuss two of the recent networks which are used in some capacity in the thesis.

3.3.2 VGG-19

With CNN developing into one of the preferred networks for computer vision, further, attempts were underway to improve performance on the ImageNet database [85]. The VGG group [86] (VGG-11, VGG-13, VGG-16, VGG-19) of networks were designed with more layers and a filter with a smaller receptive field of 3X3 (compared to 11X11 and 5X5 used in AlexNet) to extract even more features. The VGG group also are one of the biggest networks with 144 million parameters for VGG-19, with 19 weight layers [86].

3.3.3 ResNet-50

After the success of VGG net, further studies focused on exploring the possibility of even deeper networks. But the solution to better performance is not as simple as stacking multiple layers in a single thread.

First, we start facing a problem of vanishing/exploding gradients as the depth increases [87], which results in poor performance as the weights go unstable. This specific problem was addressed with the development of Inception v1 which provided two auxiliary classifiers connected to the intermediate layers for computing an auxil-

iary loss, which gets weighted to the total loss during the training phase [88]. These auxiliary classifiers were then dropped at the implementation stage. Second, when these deep networks start to converge, a "degradation problem" is encountered [89]. This is witnessed as a decreased accuracy with increasing network depth (this is not caused due to overfitting as degradation leads to a substantial increase in the training error also). This problem was addressed with the development of a "deep residual learning framework", which is a feedforward system where the input is connected to the output of one of the following layers using a direct connection. This model has fewer filters and is less complicated than VGG nets while having more than twice as many layers [89]. ResNet-50 with 50 layers has only 26 million parameters.

All the above networks discussed the use of standard CNN for image classification problems. Another interesting computer vision problem is object detection, localization, and semantic segmentation. The major difference between object detection and classification problem is the presence of multiple objects in one image make a standard CNN ineffective in the case of the former as the output length is no longer constant, another problem is the presence of an object in a different aspect ratio, size or view. A possible solution is to find specific areas in an image and use a CNN on this specific area. The next list of models is a successive improvement on this basic baseline.

3.3.4 R-CNN

Contrary to the approach of localization of image sections using a regression solver, this section follows the pattern of *"recognition using regions"* from [90] to build the R-CNN [91] network: *"Regions with CNN Features"*.

This model follows a modular design for object detection and localization by first extracting around 2000 region proposals using the selective search [92] method followed by feature extraction CNN as described in [83]. For passing all the region

proposals of random shape and aspect ratios to the CNN these regions are wrapped in a bounding box of the required size [91]. Finally, a fixed-length feature vector for each selected region is passed through class-specific SVMs for classification and a regressor for the 4 corners of a bounding box.

This is a very long process and takes huge amounts of time to train and classify 2000 regions. Also, real-time implementation is impossible as it takes around 47sec per image with a VGG16 layer CNN [93].

3.3.5 Fast R-CNN

Fast R-CNN [93] is developed by Girshick (author of the R-CNN paper [91]) to improve some of the drawbacks in the R-CNN structure. The major difference is that the image along with region proposals are directly fed into a CNN network to produce a feature map. After which, every specific object proposal passes through an RoI (region of interest) pooling layer to reshape the RoI that finally goes through a set of fully connected layers successively which branch out to two output layers; one with softmax classifier and the other a bounding box regressor.

This model is much faster than R-CNN in training as well as at run time because we don't have to feed 2000 object proposals through a CNN each time. Also, the change from SVM fitting via multi-stage training to a softmax layer's one-shot fine-tuning is sufficient and has a slightly higher mAP score. Fast R-CNN is 9X faster in training a VGG16 network, while 219X faster at runtime compared to R-CNN network [93].

3.3.6 Faster R-CNN

Both R-CNN and Fast R-CNN use selective search for region proposals, this search process is slow and is based on an algorithm that is not learned during training making it complex even in the case of simpler images with fewer features. Faster R-CNN [94] is a tweaked version of Fast R-CNN with a fully convolutional network that proposes regions and a base of Fast R-CNN structure. A Region Proposal Network (RPN) is

used to generate the region proposals. This is done by sliding a small CNN network over the feature map output by the previous convolution layer [94]. These regions along with the feature maps are sent to the RoI layer the same way as in Fast R-CNN.

3.3.7 Mask R-CNN

Mask R-CNN [95] is based on the same underlying strategy of Faster R-CNN discussed in the previous section. It has the same first level of RPN followed by a second level which has an additional *parallel layer* to predict the binary mask of each RoI.

The multi-task loss is defined by an extra term L_{mask} in addition to the classification and bounding box loss. Another distinctive feature is using the RoIAlign layer instead of the RoIPool layer. The former use bi-linear interpolation rather than the quantization feature to estimate the exact features. This leads to a large improvement in object detection as shown in [95].

The Mask R-CNN network also explores the backbone with a Feature Pyramid Network (FPN) as presented in [96]. This method takes in an input image and outputs feature maps of proportional size at multiple levels. The system is a *bottom-up* CNN pathway with a *top-down network* with lateral connections using strong semantic feature maps from high pyramid layers, refer to [96] for detailed structural information and usage. This additional FPN structure produces the best results on the COCO dataset [97].

3.4 Keypoint Detection Model

Section 3.3 provides a background of convolutional neural networks in computer vision problems for object detection. The final remarks on Mask R-CNN demonstrate the application in semantic segmentation and mask of each object in an image. This is still not what we need as the mask or bounding box of an object can help in localization but the pose estimation requires the detection of keypoints (points of interest in an image). Here we present the keypoint detection model used for the thesis and the

underlying training process.

3.4.1 Network Architecture

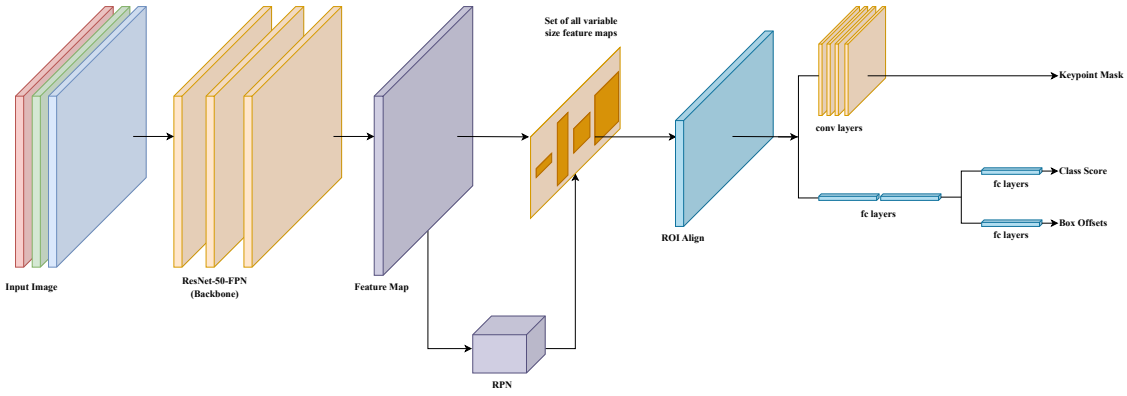


Figure 3.1: Keypoint R-CNN structure layout with a ResNet-50-FPN backbone including all components without specific layer dimensions

We use an extension of the Mask R-CNN model for our work with slight additions. For using a similar structure in keypoint detection, the location of each keypoint is assumed as a *'one-hot'* mask, which then extends to the n keypoints in any image instance. While training, only a single pixel is labeled as a foreground one-hot binary mask subsequent to each keypoint. These n keypoints are treated independently as in the case of instance segmentation masks. Other networks were also tested (such as MobileNet [98], etc.) but these did not provide accurate results in our case which is why we opted for the use of Mask R-CNN structural baseline for keypoint detection.

Different backbone structures for this Keypoint R-CNN were explored, and the ResNet-50-FPN backbone opted for its effective and accurate detection results. A deeper ResNet could have been deployed (like ResNet-101) but a tradeoff between the accuracy and computation time makes the ResNet-50-FPN variant optimal in this case. Also, this network is used as a backbone for feature extraction and a different head structure convolutional neural network is used for keypoint and bounding box which is a stack of 8 *conv* layers (3X3 filters) followed by a *deconv* layer with bilinear upscaling, resulting in a final output resolution of size [c,n,56,56] where c is the number

of classes (in this case 2, for background and object/keypoint) and n is the number of detected keypoints with an output resolution of 56X56. The optimizer parameters for stochastic gradient descent are set with a learning rate of 0.01, a momentum of 0.9, and a weight decay of 0.0005. The learning rate decays by 0.3 every 5 epochs for each parameter group.

This model serves as our first module/function f for estimating 2D keypoints ($\hat{\boldsymbol{x}}$) given an image input (ω) to solve $\hat{\boldsymbol{x}} = f(\omega; \eta)$, where η consists of the model weights learned during training of the model. Here, $\hat{\boldsymbol{x}} \in \mathbb{R}^{n\times 2}$ is a 2D pixel coordinate matrix, where $n = 5$ for the Baxter and $n = 4$ for the excavator.

3.4.2 Training

Contrary to the commonly used large datasets for training, as in [95], the study here is focused on situations where a labeled dataset is not available. This problem is mostly solved by the use of a small dataset with data augmentation or by the generation of a simulated dataset where a highly realistic 3D CAD model is available [7–9, 11, 12], which requires a transfer learning step for domain adaptation to a real-world setting. We explore the possibility of using a different approach where a highly realistic CAD model is unavailable, we use a very small set of hand-labeled (auto-labeled in the case of Baxter) real environment image datasets for supervised learning for training. The common problem with using a smaller dataset is the overfitting of the model on the training images, but with the advanced level of CNN architecture used the model is expected to perform well in operational situations.

The model training loss is defined as a sum of three loss as follows,

$$L = L_{cls} + L_{box} + L_{kp} \tag{3.1}$$

where L_{cls} is the classification loss, L_{box} is the bounding-box loss, and L_{kp} is the keypoint loss. Though including the mask enhances the keypoint accuracy [95], the mask output is not used to train the model as the collection of ground truth data for

the mask requires more computation and effort to label. The loss in each case is the binary cross-entropy loss (Eq. (3.2)) or mathematically known as the log loss. The penalty on estimation increases logarithmically with the increase in error.

$$Loss_{log} = -(y_i log(f(x_i)) + (1 - y_i) log(1 - f(x_i))) \tag{3.2}$$

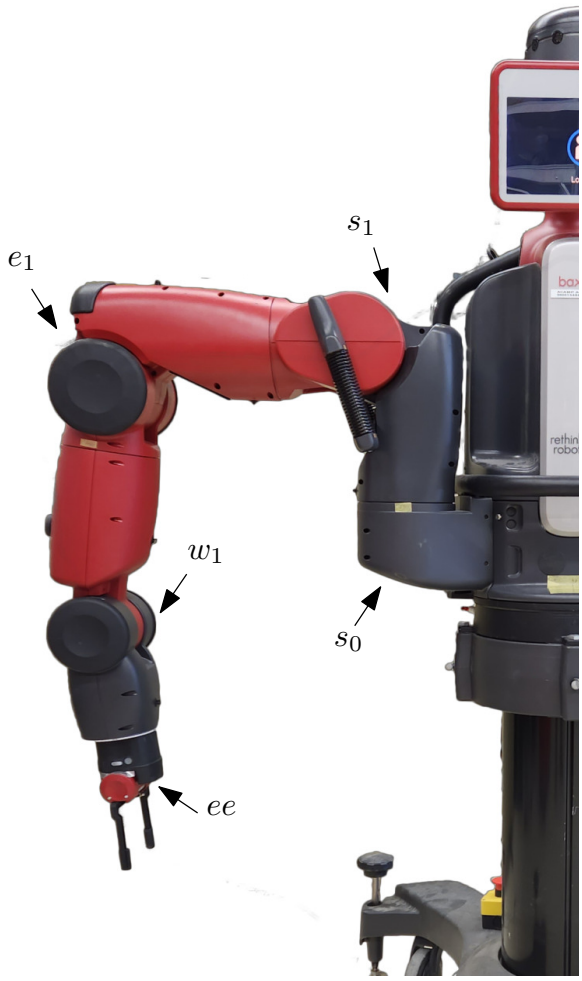


Figure 3.2: Baxter joints along with names used in the keypoint detection algorithm (s-shoulder, e-elbow, w-wrist, ee-end effector)

The keypoints are auto-computed for the Baxter based on the joint feedback from encodes (with known intrinsic and extrinsic parameters for the camera, further information in Section 4.1), the location of each keypoint is the exact location of the joint center based on the URDF file for the Baxter. We label a total of 5 keypoints (one each for joint s_0, s_1, e_1, w_1, and an end effector point) on the right arm and only

move these joints to replicate a 4-DOF system. In the case of the excavator, the base remains stationary and all other joints are rotated randomly. The keypoints here are labeled manually corresponding to highly visible points or joint positions on the front face and not the 3D centroid, as in the case of Baxter. Here we only hand-label 4 keypoints, corresponding to the minimum required number of keypoints for detection and control in a 4-DOF system.

3.4.3 Evaluation Techniques

For evaluating the performance of the keypoint detection model two strategies are used; Percentage of Correct Keypoints (PCK) and Mean Absolute Error (MAE). PCK@0.2 (pixel error$<$ 0.2 $\max(h, w)$; height, width of the bounding box) score is used to evaluate the performance of the model in real-time tests for the Baxter robot, as ground truth keypoints can be computed directly from the available data. In the case of the excavator, no such way to compute real-time ground truth keypoints is available, necessitating only the provision of qualitative results at this stage. However, in the subsequent chapters, more robust performance metrics will be utilized to provide a more quantifiable assessment.

3.5 Experimental Setup and Results

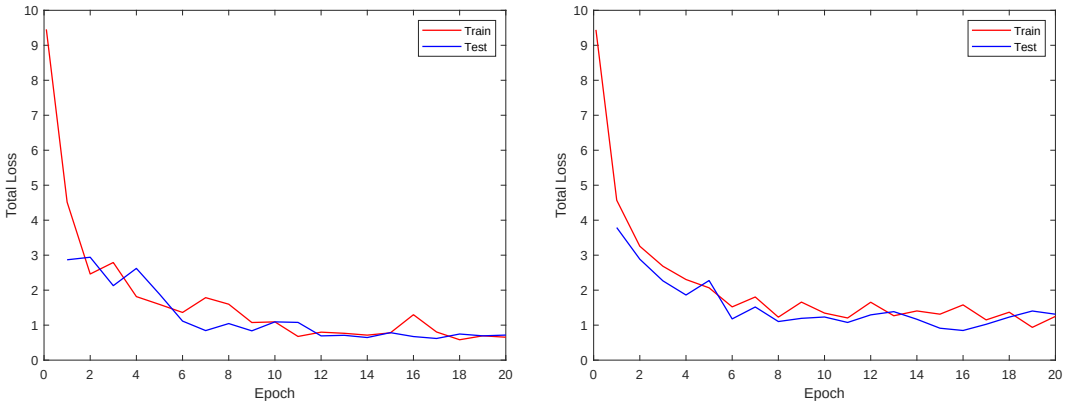
3.5.1 Dataset

The dataset for the entire training is collected in the lab environment with a backdrop of single color separators exactly behind the manipulator for easy distinguishment of the joint from the lab background. The training is done with this as the background for extracting as many features as possible of the manipulator arm using the limited training dataset. For both the Baxter and the scale excavator we move the 4 joints randomly to cover a wide trajectory operation range and collect image data from the RealSense D435i camera facing front directly towards the arm with an image size of 640 X 480.

Also throughout the process, the Baxter joint states are recorded along with the image to obtain the ground truth keypoint positions. However, for the excavator, no such encoder data is available so the keypoints are hand-labeled. We capture 650 images of the Baxter and 350 images of the excavator in random joint orientations. Then we randomly divide the training and test set in a 9:1 ratio, respectively.

3.5.2 Results and Discussion

We first evaluate the model training based on the calculated loss. Training and test loss are shown in Figure 3.3 representing a competitive final test loss compared to training. This shows that our model works well with limited real-case training images while avoiding overfitting. The training is done for a total of 20 epochs and the batch size used is 20 images.



(a) Baxter Robot (b) Excavator

Figure 3.3: Training and test loss for 20 epochs, total loss is the sum of all losses as described in Equation (3.1)

For the training, we also report the average precision (AP) score as per COCO evaluation metrics on the test images at the end of 20 epochs. We edit the *OKS* sigma to correspond to the correct number of keypoints for our problem and report the AP for bounding box and keypoint for both the models in Table 3.1

After evaluating the model accuracy using the loss functions in training, the final model is deployed for real-time estimation. Tables 3.2 and 3.3 summarizes the

Table 3.1: AP on test images for box and keypoint detection. The backbone is ResNet-50-FPN (higher the better)

	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}
Baxter	0.738	0.99	0.95	0.948	0.99	0.99
Excavator	0.928	0.99	0.99	0.987	0.99	0.99

accuracy for keypoint detection in the case of the Baxter robot from our method compared to a popular keypoint detection method DREAM (using the VGG-Q CNN structure) [7]. This data is collected to estimate the joint position in real-time using the two models and comparing them with ground truth joint angles recorded using the onboard encoders. Our model performs better in detection accuracy in three of the total five detected keypoints (namely shoulder, wrist, and hand) with a huge difference in the results for the hand keypoint where our model has 50% less error compared to DREAM. Our model represents a more robust and consistent detection with a PCK@0.2 score higher than 90% in all joint detection cases.

Table 3.2: PCK@0.2 for each joint position in real-time estimation of Baxter robot (higher the better)

	Base	**Shoulder**	**Elbow**	**Wrist**	**Hand**
DREAM [7]	**0.99**	0.98	**0.98**	0.95	0.76
VECTOR (Ours)	0.96	**0.99**	0.94	**0.97**	**0.91**

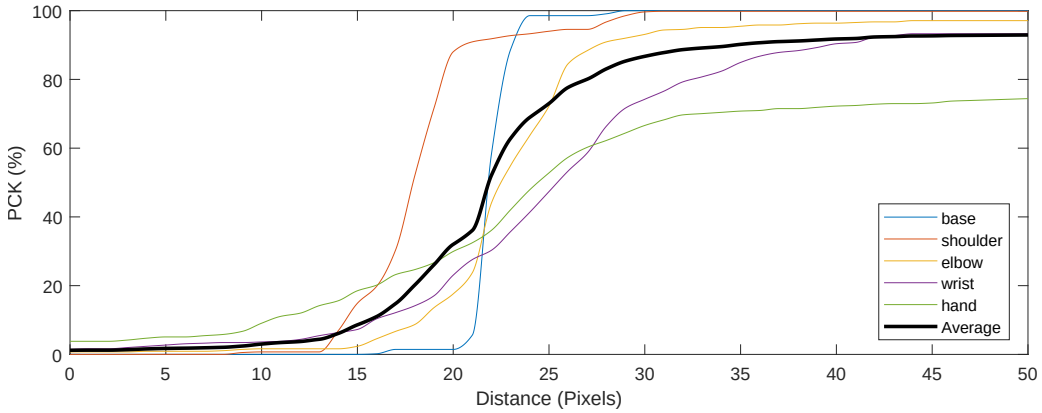
Table 3.3: MAE (in pixels) for each joint position in real-time estimation of Baxter robot (lower the better)

	Base	**Shoulder**	**Elbow**	**Wrist**	**Hand**
DREAM [7]	**22.49**	23.38	**28.10**	38.93	86.89
VECTOR (Ours)	42.55	**19.11**	33.82	**30.44**	**42.52**

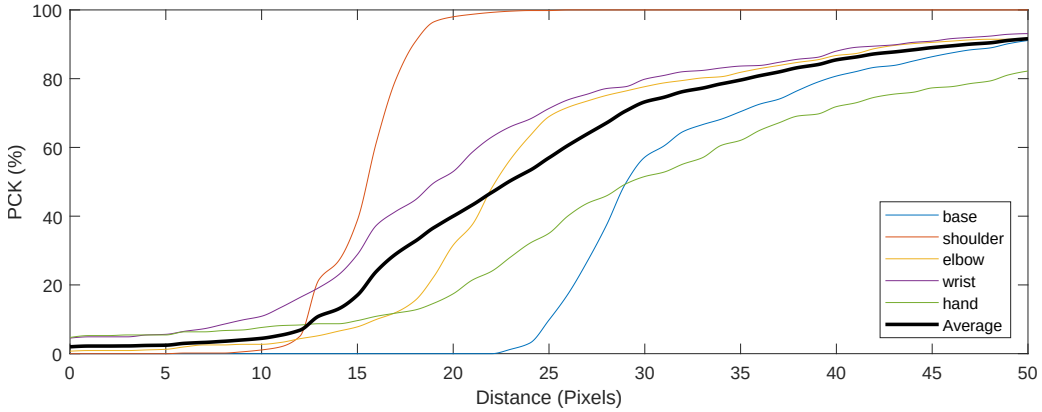
The same test results are also used to plot the absolute PCK(%age) in Figure 3.4 and compare the relative accuracy. The black solid line represents the average PCK value which is comparable for both models. This shows our model trained on a much smaller training set performing almost as well as or even better in some cases than DREAM.

One possible reason for DREAM to have lower accuracy, in this case, is the images used in training are synthetic simulated images and do not represent the exact lab environment, whereas our model is trained on images captured in the same lab as the real-time test environment. This makes the direct generalized comparison not applicable as DREAM might possibly work better in a different setting than our lab as it is trained on a wider range of images. Having mentioned that, our study's focus is not to compare benchmark detection models with ours but to showcase a possible way of keypoint detection where no pre-trained model exists or the available training set is small.

All these quantitative real-time assessment results cannot be obtained for the excavator as the ground truth keypoint position is unknown in this case. This restriction makes only qualitative assessments to be presented in this chapter. We use the same setup and move the excavator to random positions in the work area and collect images with keypoint overlay in real-time (Figure 3.5). The four detected keypoints are connected with lines with the base keypoint found using the extrinsic parameters as described in Section 4.1.2 corresponding to the absolute location of the first joint or the base of the excavator. Low-light detection results are shown in Figure 3.6 and some cases where the detection is not accurate are shown in Figure 3.7.



(a) DREAM [7]



(b) VECTOR (Ours)

Figure 3.4: PCK Scores with respect to pixel distance in real-time estimation of Baxter robot

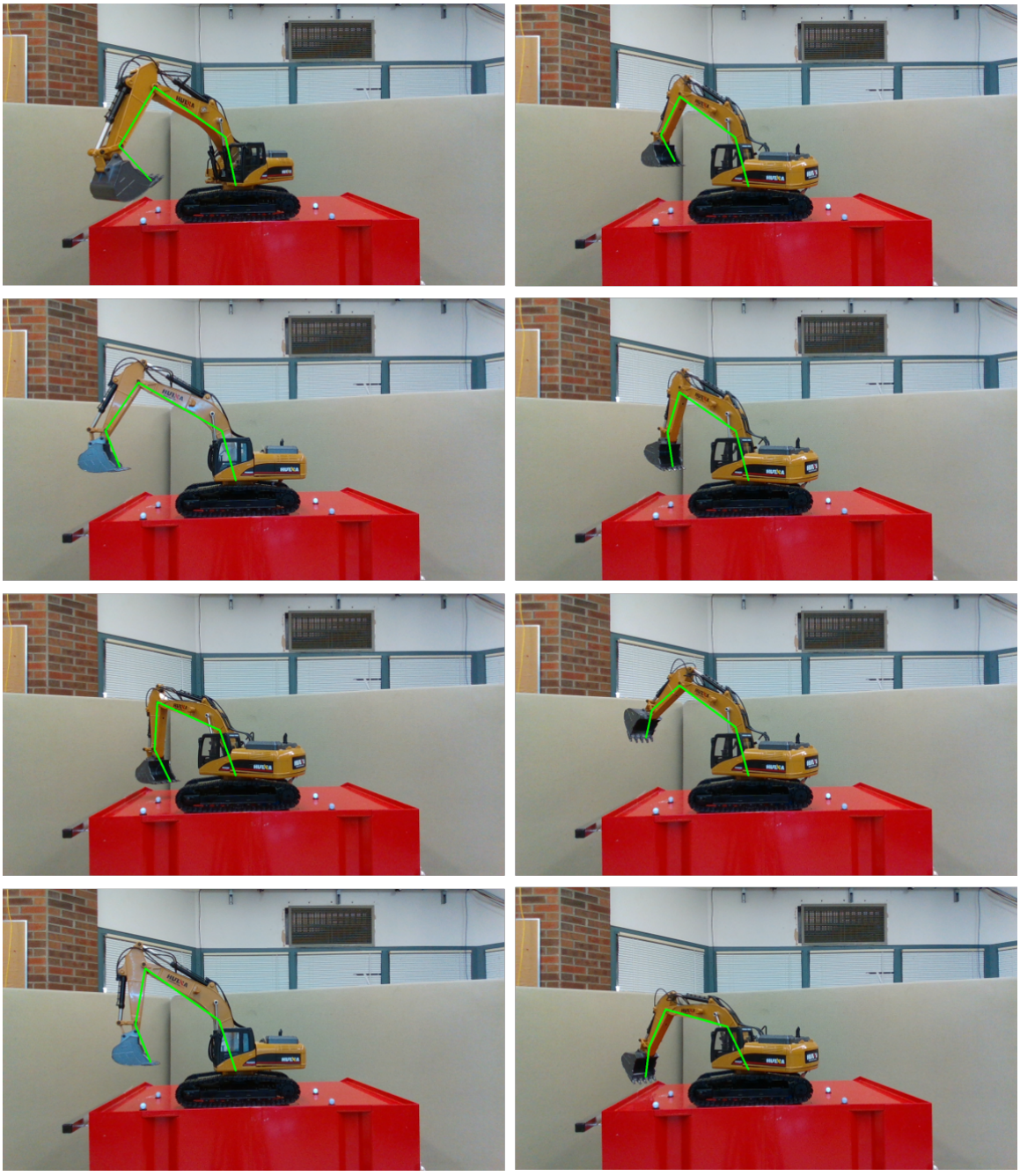


Figure 3.5: Qualitative results in real-time assessment of the excavator in random positions in the work area, keypoints are connected with green solid lines where the base keypoint is fixed and calculated using the extrinsic parameters

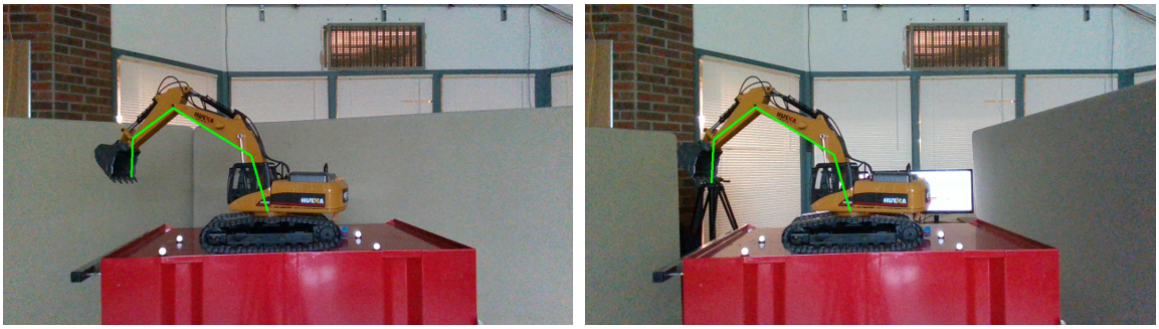


Figure 3.6: Results in real-time assessment of the excavator; low-light (left), low-light and dynamic background (right)

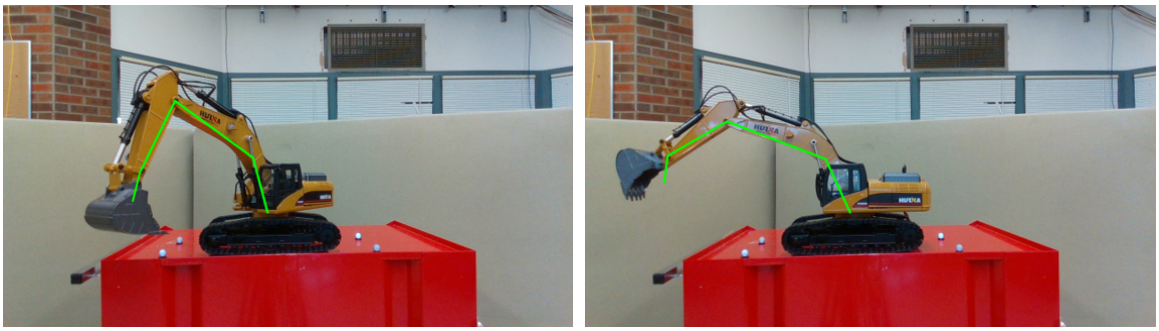


Figure 3.7: Low accuracy detection cases in real-time assessment of the excavator, major error in detection of end-effector while other keypoints are still accurate

Chapter 4

Joint State Estimation from Detected Keypoints

This chapter details the process implemented to find the joint state estimates from the 2D keypoints detected using the neural network discussed in the previous section. We first introduce the image formation and mathematical models used to find the relation between 2D image pixel coordinates and their corresponding 3D world coordinates. Following this, the forward kinematic equations for the specific robot model are listed along with the various iterative optimization techniques used. Finally, joint angle computation is carried out in real-time and results are compared for both robot manipulators.

4.1 Image Formation

We deploy a digital camera that includes transparent lenses to form a visual image on a sensing surface. This is done by directing the light through the lens. In terms of optics, the light is directed in three different ways: reflection, refraction, and deflection [99]. For our research, we only consider the effect of refraction and further use a thin lens model to represent the monocular camera in the study.

The thin lens is defined by an axis passing through the center of the lens, known as the *optical axis*. The functionality of this thin lens is governed by two characteristics. The first is that a ray parallel to the optical axis entering the aperture of the lens

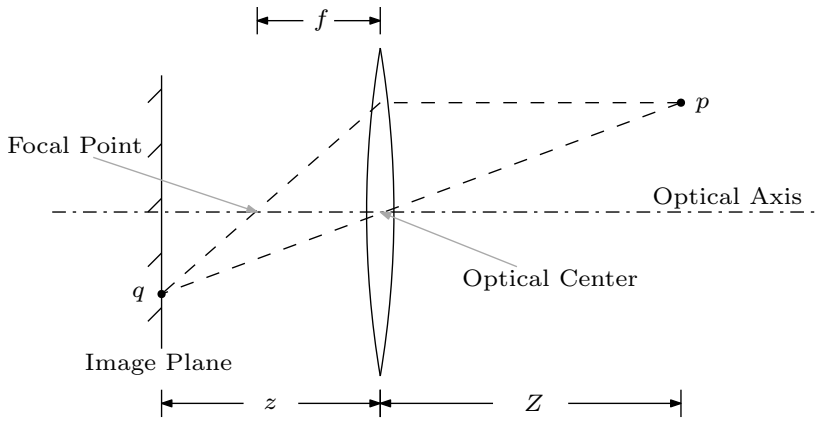


Figure 4.1: Thin lens refraction model for a point p

intersects the optical axis at a distance f (focal length) from the optical center. This point is called the focus of the lens and is a fixed property for each lens. The second is that any ray passing through the optical center remains undeflected.

Figure 4.1 shows a point p at a distance Z from the lens projected on the image plane of the camera as a point q, located at a distance z from the lens. This point q is the image of the point p.

Using similar triangles in Figure 4.1 yields the *fundamental equation of thin lens* [99],

$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{f}$$

4.1.1 Intrinsic Parameters

To simplify this model further, it is assumed that the aperture of the thin lens is almost zero. In such conditions rays from a point p will pass only through the optical center and in turn will remain undeflected. This model is known as the *pinhole camera* model. In this model, the distance between the lens and the image frame is fixed as f (focal length of the pinhole camera), because all rays from the lens at the optical center will pass through the focal length as per design characteristics.

In Figure 4.2, we attach a lens-fixed reference frame C at the optical centre of the thin lens and a frame I on the image plane with origin at the intersection of

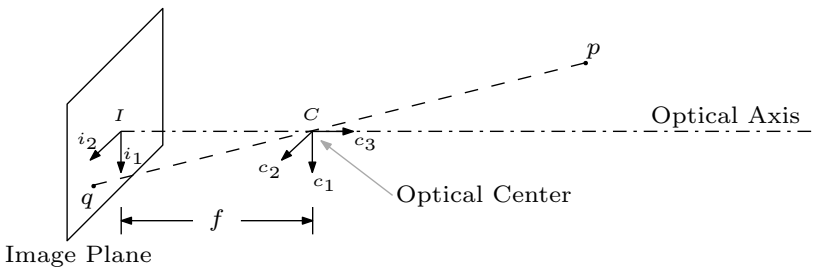


Figure 4.2: Pinhole camera refraction model for a point p in 3D space

image plane with the optical axis. The axis are chosen as per the standard camera convention. Assuming, the location of point p to be (X,Y,Z) in frame C and its corresponding image point q as (x,y) in frame I. Using similar triangles we have,

$$x = -f\frac{X}{Z},\ y = -f\frac{Y}{Z} \tag{4.1}$$

The negative sign in the equation represents that the image we obtain on the camera sensor array is a mirror-flipped, upside-down version of the point p in front of the camera. This effect is compensated inside the firmware of the camera such that we obtain a corrected version of the point. To represent this mathematically we assume the image frame to be in front of the lens (Figure 4.3), this flips the sign of f in Equation (4.1), and the new image point (x,y) in frame I will be,

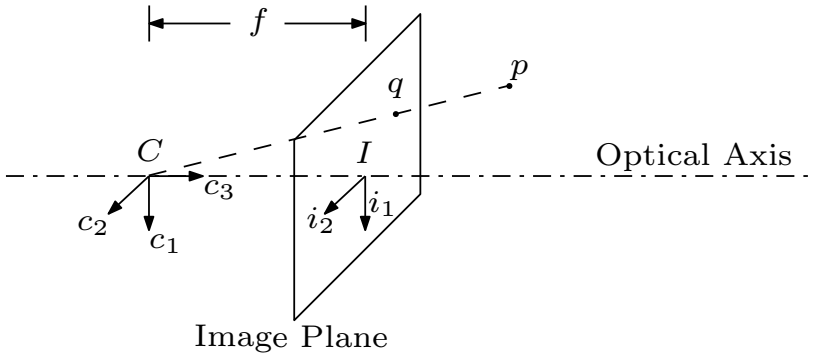


Figure 4.3: Frontal pinhole camera refraction model for a point p in 3D space

$$x = f\frac{X}{Z},\ y = f\frac{Y}{Z} \tag{4.2}$$

Where $f>0$ is the focal length of the lens. Additionally, Figure 4.3 is only to represent the mathematical relation of the image point as seen on the output image and has no physical meaning. This map from 3D points($\boldsymbol{X} = [X, Y, Z]^T$) to corresponding 2D image points($\boldsymbol{x} = [x, y]^T$) is represented as a map h,

$$h : \mathbb{R}^3 \to \mathbb{R}^2;\ \boldsymbol{X} \mapsto \boldsymbol{x}$$

This map in homogeneous coordinates can be represented as,

$$\begin{aligned}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX/Z \\ fY/Z \\ 1 \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} X \\ Y \\ Z/f \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\ = \frac{1}{Z}\underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{K_f}\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\Pi_0}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}\end{aligned} \tag{4.3}$$

To maintain dimensional homogeneity in Equation (4.3) units of (x,y), f, (X,Y,Z) should be consistent, such as m (or cm, or in). But, as we are using a digital camera the output image will have units of pixels (px) with the origin at the top-left of the image frame. Due to this inconsistency in units and origin location, another transformation is required to represent (x,y) coordinates in their corresponding pixel coordinates (u,v). This transformation is done by scaling the image from m to px and then by translating the origin to the top-left corner. This is mathematically done using,

$$u = s_x x + c_x$$

$$v = s_y y + c_y$$

where s_x, s_y is the scaling factor having units px/m and $(c_x, c_y)$are the coordinates of the optical centre. We can write this in matrix form as,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.4}$$

Combining Equation (4.3) and Equation (4.4) gives:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z} \underbrace{\begin{bmatrix} s_x f & 0 & c_x \\ 0 & s_y f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\Pi_0} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.5}$$

where K is the *intrinsic camera matrix*, governed by the physics of the camera and is fixed. Further, we define the projection matrix for the camera $P{:=}K\,\Pi_0$.

4.1.1.1 Camera Calibration

Equation (4.5) is the ideal transformation between the 3D world coordinates and the 2D image points, it does not account for *distortions* in the image, which warps the image in different ways. Different methods are used to remove this distortion from the image of which we use the default model using the OpenCV libraries known as the *plum bob model* where the input is the distorted image in the image frame (x_d, y_d) and outputs the corrected image in the image frame as (x, y) by using the following,

$$\begin{aligned} r^2 &= x_d^2 + y_d^2 \\ x &= x_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ y &= y_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 x_d y_d + p_1(r^2 + 2y_d^2) \end{aligned} \tag{4.6}$$

where k_1, k_2, k_3 model the effect of *radial* distortion created by wide angle lens, and p_1, p_2 model the *tangential* distortion created by nonparallel lens and imaging sensor planes.

The above parameters as well as the intrinsic camera matrix K can be found using a camera calibration process with a checkerboard that has precisely known dimensions and number of squares.

The distortion is auto-rectified after performing the calibration step and therefore the distortion coefficients are not reported here. The intrinsic camera matrix K is essential for solving Equation (4.5) and is found out as follows,

$$K = \begin{bmatrix} 918.6686 & 0 & 639.3897 \\ 0 & 918.4182 & 361.0150 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

4.1.2 Extrinsic Parameters

Let us now consider the point p again but with coordinates $\boldsymbol{X_w} = [X_w, Y_w, Z_w]^T \in \mathbb{R}^3$ relative the the world frame. The transformation from the world to the camera frame for the point p can be represented as a rigid-body transformation $g_{cw} = ({}^cR_w, {}^ct_w) \in SO(3) \times \mathbb{R}^3 \coloneqq SE(3)$ (known as the Special Euclidean Group).

$$\boldsymbol{X} = {}^cR_w \boldsymbol{X_w} + {}^ct_w$$

Here we use g_{cw}, the transformation of the point in world frame to the camera frame, converse to most common practice to use the reference frame as the world. This makes the matrix multiplication and substitution easier with the previously discussed camera parameters. From this coordinate transform we can write,

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^cR_w & {}^ct_w \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.8}$$

Combining Equation (4.8) and Equation (4.5), the overall model is,

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x f & 0 & c_x \\ 0 & s_y f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^cR_w & {}^ct_w \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

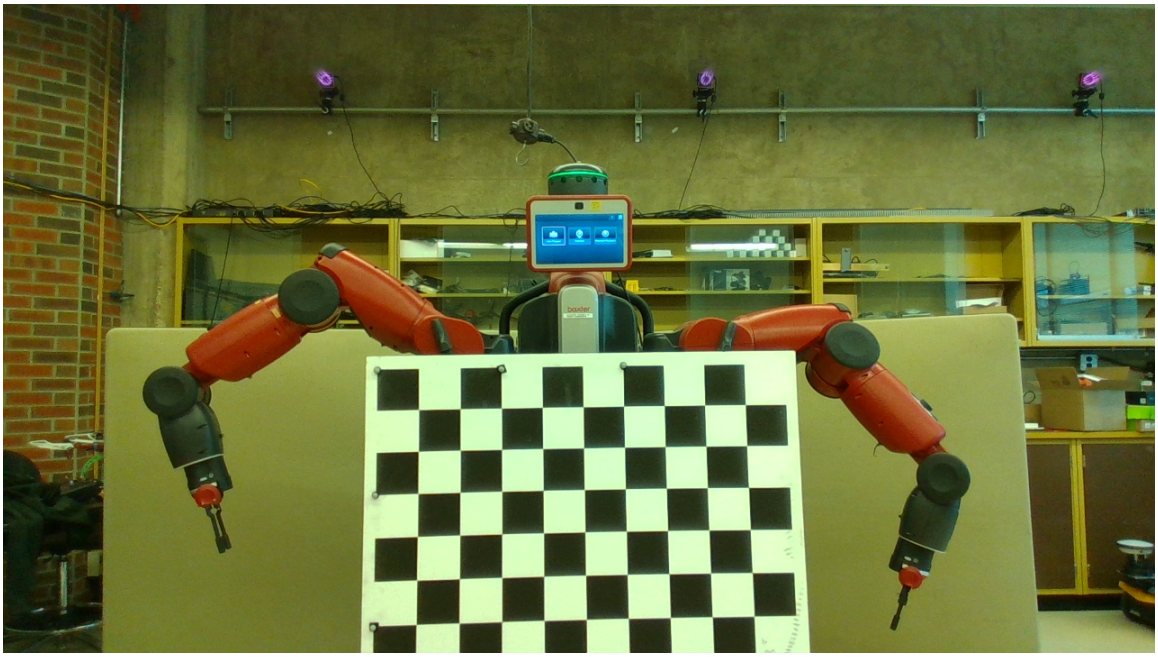


Figure 4.4: Checkerboard used for camera calibration and extrinsic parameter identification with each square of size $10cm \times 10cm$

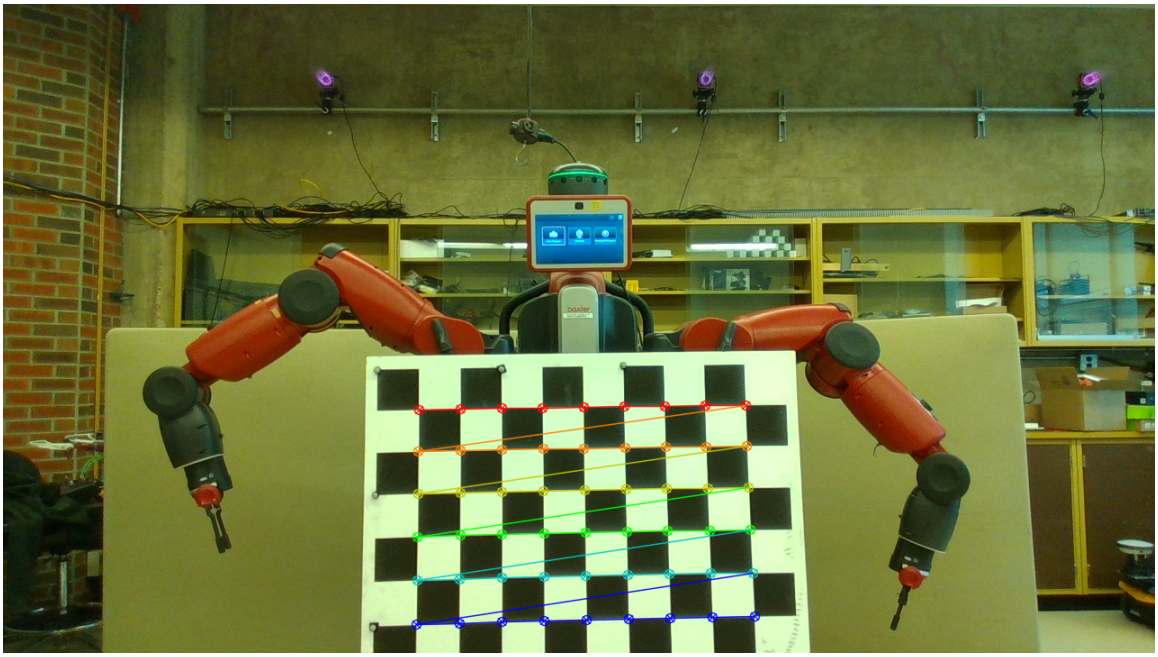


Figure 4.5: Checkerboard with detected corner points labeled using a Python script and OpenCV PnP computations

or,

$$Z\boldsymbol{x} = K\Pi_0\boldsymbol{X} = K\Pi_0 g_{cw}\boldsymbol{X_w} \tag{4.9}$$

The matrix g_{cw} is computed using a special multi-stage method where we first compute the transformation between the camera frame and the checkerboard frame h (g_{ch}) and then the checkerboard with the Vicon tracker origin frame o (g_{ho}) and finally the Vicon tracker origin and world frame (g_{ow}); in our case the base of each robot to give,

$$g_{cw} = g_{ch} \times g_{ho} \times g_{ow}$$

where for the Baxter robot,

$$g_{cw} = \begin{bmatrix} 0.03318 & -0.0693 & -0.9969 & 2.0951 \\ 0.9993 & 0.0084 & 0.0327 & -0.1358 \\ 0.0061 & -0.9975 & 0.0695 & 0.1866 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

and for the excavator,

$$g_{cw} = \begin{bmatrix} 0.0548 & -0.0306 & -0.9980 & 0.9539 \\ 0.9985 & 0.0014 & 0.0548 & -0.0052 \\ 0.0033 & -0.9995 & 0.0309 & 0.0770 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

4.2 Manipulator Kinematics

A manipulator consists of a set of rigid links connected by different joints. The kinematics describes the relation between the rigid links in motion. Throughout this thesis we only use one kind of joint mechanisms known as the *revolute joints*.

A revolute joint has only one degree of freedom, rotation along a fixed axis. As we will be using twist coordinates to represent the motion of the joints in 3D space, each revolute joint can be represented as a screw motion with zero pitch.

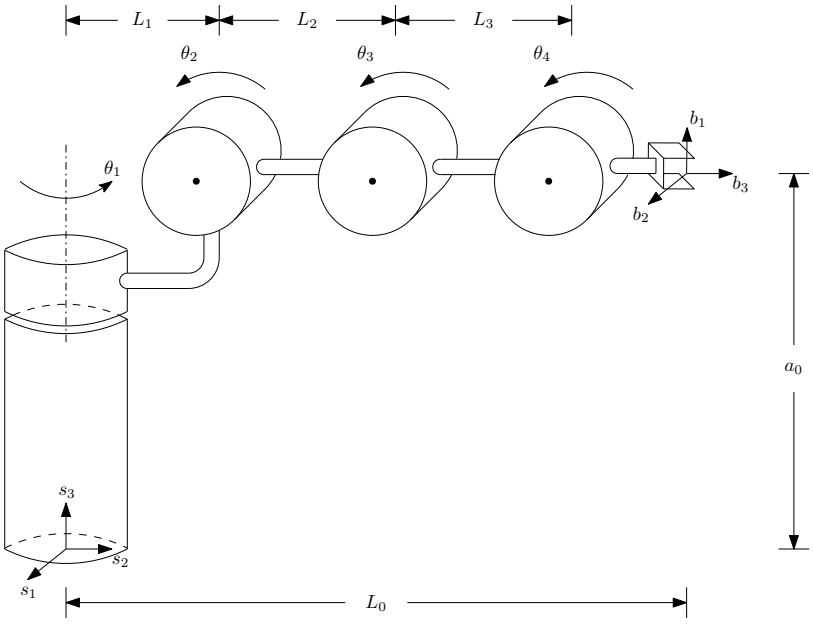


Figure 4.6: $RRRR$ Manipulator in reference configuration

4.2.1 Exponential Coordinates for rigid motion

We define axis of rotation as $\omega \in \mathbb{R}^3, ||w|| = 1$, and $p \in \mathbb{R}^3$ is a point on this axis [100].

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

where $\hat{\omega} \in so(3)$ is a skew-symmetric matrix of ω, and $v = -\omega \times p$ (specific only for revolute joint). Using this we define $\hat{\xi} \in se(3)$. Each transformation with angular change can be represented in terms of the reference frame as:

$$g_{ab}(\theta) = e^{\hat{\xi}\theta} g_{ab}(0) \tag{4.12}$$

Expanding the Equation (4.12) for multi joint manipulator, with n joints and angle of rotation of of each joint as θ_1, θ_2, θ_3 ... θ_n. Starting from the last and moving down to the first we have the following,

$$\begin{aligned} g_{ab}(\theta_n) &= e^{\hat{\xi}_n \theta_n} g_{ab}(0) \\ g_{ab}(\theta_{n-1}, \theta_n) &= e^{\hat{\xi}_{n-1}\theta_{n-1}} e^{\hat{\xi}_n \theta_n} g_{ab}(0) \\ g_{ab}(\theta_1 \cdots \theta_n) &= e^{\hat{\xi}_1 \theta_1} \cdots e^{\hat{\xi}_n \theta_n} g_{ab}(0) \end{aligned} \tag{4.13}$$

Equation (4.13) is only based on $\hat{\xi}$ and $g_{ab}(0)$ which is a representation of the physical constraints and the reference position (base w.r.t. end-effector) respectively. This method to compute the forward dynamics known as POE (Product of Exponentials) formulation has an edge over the most commonly used Denavit-Hartenberg formulation [101] of kinematics. The Denavit-Hartenberg formulation computes a relation between the successive links of a manipulator which multiply to form the full end-to-end forward kinematics. For solving the inverse kinematic problem Denavit-Hartenberg formulation becomes cumbersome to solve while the POE forms an elegant formulation of a group of the canonical problems [100].

4.3 Joint Angle Computation Methodology

4.3.1 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm [102, 103] or widely known as the LM algorithm is a technique used to find the optimal solution to non-linear least square problems. The technique is a combination of the Gauss-Newton method and the Gradient Descent method [104]. It behaves like the gradient descent, that is reducing the sum of squared errors by changing the parameter in the steepest-descent direction, when the initial value is far from the optimal solution. After the error is small and the estimated value is close to the optimal solution the algorithm weights more to decrease the sum of squared error by assuming the function to be quadratic locally and then computing the minimum, which is in line with the Gauss-Newton method. This method is faster and more robust to find an optimal solution than using the Gauss-Newton method. But the problem of getting stuck at a local minimum is still persistent as is the case with almost any iterative optimization algorithm.

We consider the problem of finding the joint state estimates from the keypoint data using the LM algorithm. We use the error term $\boldsymbol{d}$ and define the problem of finding

the optimum solution as follows,

$$\begin{aligned} &\underset{\theta_1,\theta_2,\theta_3,\theta_4}{\arg\min}\ \boldsymbol{d}^T\boldsymbol{d} \\ &where,\ \boldsymbol{d} = \begin{bmatrix} d_1 & d_2 & d_3 & d_4 \end{bmatrix}^T \end{aligned} \tag{4.14}$$

Following is a discussion of the different equations used to find the optimal solution.

4.3.2 3D Distance Method

This is a way where we use the mathematical foundations of manipulator kinematics to compute the estimated 3D world coordinate ($\hat{\boldsymbol{X}}_{\boldsymbol{i}}$) of i^{th} joint center corresponding to each CNN estimated 2D pixel location ($\hat{x_i}$) and then leverage the rigid dimension constraints between joints to estimate the joint angles. This method closely resembles a depth estimation from an available 2D image and a known transformation of a point on the camera frame. Rearranging Equation (4.9) we have,

$$\hat{\boldsymbol{X}}_{\boldsymbol{i}} = Z_i g_{wc} K_a \hat{x_i} \tag{4.15}$$

where $g_{wc} = g_{cw}^{-1}$ and $K_a = (K\Pi_0)^{-1}$.

Denoting the distance between two successive joints i and $i+1$ (or link length) as l_i,

$$\begin{aligned} \left|\left| \hat{\boldsymbol{X}}_{\boldsymbol{i}} - \hat{\boldsymbol{X}_{\boldsymbol{i+1}}} \right|\right| &= l_i \\ (\hat{\boldsymbol{X}}_{\boldsymbol{i}} - \hat{\boldsymbol{X}}_{\boldsymbol{i+1}})^T (\hat{\boldsymbol{X}}_{\boldsymbol{i}} - \hat{\boldsymbol{X}}_{\boldsymbol{i+1}}) &= l_i^2 \end{aligned} \tag{4.16}$$

Simplifying Equation (4.16) using Equation (4.15) gives,

$$a_i Z_i^2 + b_i Z_i Z_{i+1} + c_i Z_{i+1}^2 = l_i^2 \tag{4.17}$$

where,

$$\begin{aligned} a_i &= \hat{x_i}^T M \hat{x_i}, \\ b_i &= -(\hat{x_i}^T M \hat{x}_{i+1} + \hat{x}_{i+1}^T M \hat{x_i}), \\ c_i &= \hat{x}_{i+1}^T M \hat{x}_{i+1}, \text{ and} \\ M &= (K_a K_a^T)^{-1} \end{aligned}$$

are all known. Using Equation (4.17) for four known link lengths and known Z_1, the problem reduces to a set of four equations with four variables, and solving them simultaneously yields the Z_i for the remaining points. This can then be used to solve Equation (4.15) and obtain $\hat{\boldsymbol{X}}_{\boldsymbol{i}}$ for joint estimation.

This method was tested initially but ignored later due to inconsistent results with low accuracy.

4.3.3 Inverse Kinematic Solution

4.3.3.1 2D Euclidean Distance Method

Assuming the estimated 2D pixel location of i^{th} joint center (with frame j_i) from the CNN as $\hat{x}_i$ and the location calculated using the forward kinematics as x_i where $x_i \coloneqq f(g_{sj_i}; \theta_1 \cdots \theta_i)$. And formulating the elements of $\boldsymbol{d}$ as,

$$\begin{aligned} d_1 &= \left[\hat{x}_1 - x_1\right]; \quad d_2 = \left[\hat{x}_2 - x_2\right] \\ d_3 &= \left[\hat{x}_3 - x_3\right]; \quad d_4 = \left[\hat{x}_4 - x_4\right] \end{aligned} \tag{4.18}$$

Going back to Figure 4.6 and using the POE formulation as discussed in Equation (4.13), we define the i^{th} joint position in the 3D world frame $\boldsymbol{X}_{\boldsymbol{i}}$ as follows,

$$\boldsymbol{X_1} = e^{\hat{\xi}_1\theta_1} g_{sj_1}(0) \tag{4.19}$$

$$\boldsymbol{X_2} = e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} g_{sj_2}(0) \tag{4.20}$$

$$\boldsymbol{X_3} = e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} e^{\hat{\xi}_3\theta_3} g_{sj_3}(0) \tag{4.21}$$

$$\boldsymbol{X_4} = e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} e^{\hat{\xi}_3\theta_3} e^{\hat{\xi}_4\theta_4} g_{sj_4}(0) \tag{4.22}$$

Using the above in Equation (4.9), the 2D image pixel x_i based on manipulator kinematics is,

$$x_i = \frac{1}{Z_i} K \Pi_0 g_{cw} \boldsymbol{X}_{\boldsymbol{i}} \; \forall \; i = 1, 2, 3, 4 \tag{4.23}$$

The entire iterative process to find the solution is depicted in Figure 4.7.

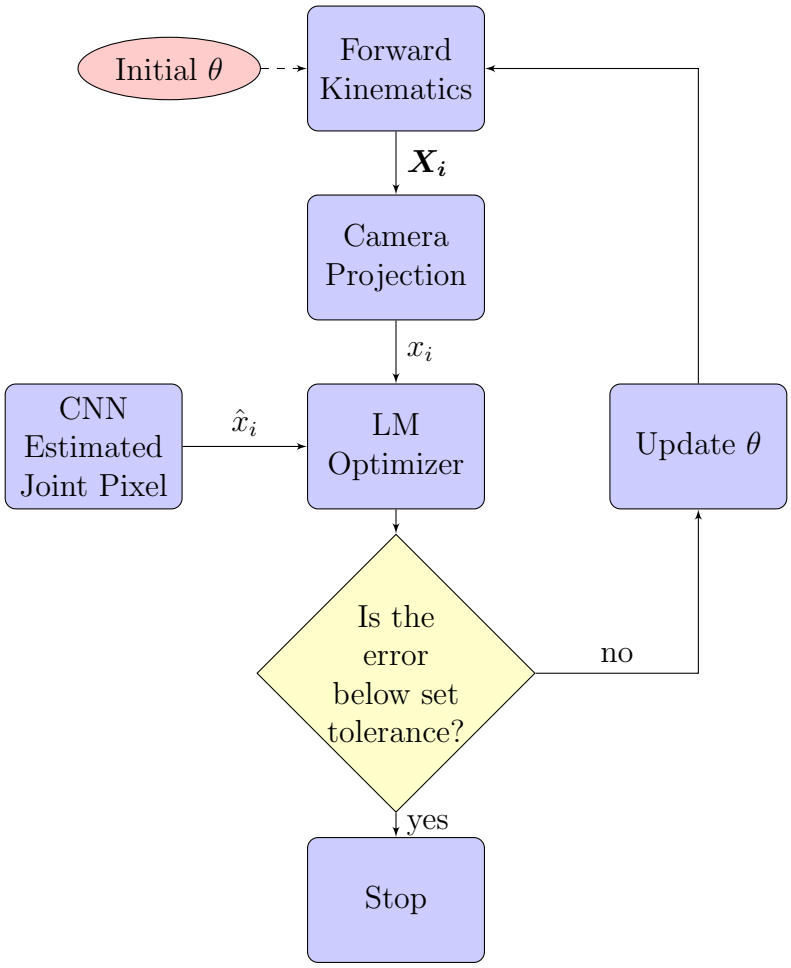


Figure 4.7: Iterative optimization process for angle estimation depicting step-wise computations

4.3.3.2 2D Pixel Matching using Object Keypoint Similarity

This is another possible way that can help improve the error tolerance of the numerical solver. We use a modified version of the object keypoint similarity (OKS) (as defined by MS-COCO [97]) error term instead of a mean squared error term.

$$OKS = \exp\left(-\frac{d_i^2}{2k_i^2}\right)$$
$$\text{where }, d_i = \left[\hat{x}_i - x_i\right] \ \forall\ i = 1, 2, 3, 4$$

similar to the d_i in Eq. (4.18). The underlying idea of using this term is that the error should take into account the scale of the image, thus making it more sensitive

to changes in size and keypoint location based on the central position. This technique is commonly applied in human keypoint detection, where keypoints on the face require more precise localization compared to keypoints on the arms or legs, which can tolerate a larger degree of error in the image frame.

We use the direct 2D euclidean distance inverse kinematic method for our joint estimation optimization problem. We use the URDF file to find the mathematical relation between the joints to compute the g_{sj_i} for the Equations (4.19) to (4.22). For the Baxter robot, the URDF file is readily available from the manufacturer, whereas for the excavator no such arm dimensions are available. This is where we generate our own URDF file based on physical measurements and a CAD file in Solidworks, more details on the specifics are in Section 5.3.1. Further, for the excavator the points are hand labeled which are not corresponding to the joint centers, this requires an extra transformation from the joint center to the detected keypoint location for optimization. In the case of the Baxter robot, keypoints are at the joint center so no such transformation is required.

4.4 Experimental Results

Using the information from Section 4.1, the intrinsic parameters of the camera are calculated using the camera calibration (Section 4.1.1.1), and the extrinsic parameters using the code for the camera to base transformation using Vicon markers and a checkerboard. The 3D pose estimation is a non-linear non-convex problem to solve for joint angles $\boldsymbol{\theta_m} = g(\hat{\boldsymbol{x}}, \zeta)$ where $\hat{\boldsymbol{x}} \in \mathbb{R}^{n\times 2}$, $\boldsymbol{\theta_m} \in \mathbb{R}^{n\times 1}$ and ζ is a set of equations consisting of K, g_{cw}, g_{sj_i} and $\hat{\xi}_i$; all the geometric information of the robot manipulator. For all of the results, direct 2D Euclidean distance (Section 4.3.3.1) optimization is used. The direct 3D distance method is a more conventional way to estimate depth using a 2D RGB image but doesn't work as efficiently for our case due to the additional variables in the optimization; (X, Y, Z)

Algorithm 1 Joint angle optimization for robot arm manipulator

Input: N joint keypoints ($\hat{\boldsymbol{x}}$)
Output: Estimated joint angles ($\boldsymbol{\theta_m}$)

procedure ANGLE OPTIMIZE($\hat{\boldsymbol{x}}$)
    `//Initialization`
    $\boldsymbol{\xi(\theta)} \leftarrow P.O.E.$ ▷ Forward kinematics equations
    $\boldsymbol{\theta^{(0)}} \leftarrow \theta_{guess}$ ▷ Initial guess
    $\boldsymbol{\Lambda} \leftarrow K, g_{cw}$ ▷ Camera transformation parameters
    $j \leftarrow 0$
    `//Optimization iterations`
    while $j \leq max\_itr - 1$ **do**
        $\boldsymbol{X^j} \leftarrow \boldsymbol{\xi(\theta^j)}$
        $\boldsymbol{x^j} \leftarrow \boldsymbol{\Lambda X^j}$
        $\boldsymbol{\theta^{j+1}} \leftarrow$ optimize($\boldsymbol{x^j}, \hat{\boldsymbol{x}}$) ▷ LM Optimizer
        $\boldsymbol{\varepsilon^{j+1}} \leftarrow$ error($\boldsymbol{x^j}, \hat{\boldsymbol{x}}$)
        $\boldsymbol{\theta_m} \leftarrow \boldsymbol{\theta^{j+1}}$
        if $\boldsymbol{\varepsilon^{(j+1)}} \leq \boldsymbol{\varepsilon_{tol}}$ **then**
            break
        end if
        $j \leftarrow j + 1$
    end while
end procedure

in 3D coordinate system for each keypoint compared to (x, y) in 2D image coordinates. A possibly advanced version of the optimization of the 2D keypoint method was also explored by using Object Keypoint Similarity (OKS) replacing the mean square error term (used commonly as a performance evaluation matrix in human keypoint detection) but the presence of other highly sensitive free parameters (such as k_i) that have to be manually adjusted with changing robot pose in the camera frame makes the use very restricted and therefore ignore in this study. We include the standard C++ files `unsupported/Eigen/NonLinearOptimization` and `unsupported/Eigen/NumericalDiff` with the LM parameters set as `maxfev` =350, `xtol` = 1.0e-10, `factor` = 0.1 to build the iterative optimizer loop. All the results presented here are real-time assessment results as was done in Chapter 3.

4.4.1 Baxter Robot

The joint estimation for the Baxter is straightforward using Equation (4.18) and Equation (4.23) as no extra computation is required other than the URDF-derived twist coordinates of the joints being actuated. The keypoint detected is corresponding to the 3D centroid of the joint and no further transformation is required. The results also present a comparison of our method with DREAM [7] using the same optimization step.

Table 4.1: MAE (in degree) for each joint angle in real-time estimation of Baxter robot (lower the better)

	$\theta_1(s_0)$	$\theta_2(s_1)$	$\theta_3(e_1)$	$\theta_4(w_1)$
DREAM [7]	4.548	**3.328**	**6.251**	4.491
VECTOR (Ours)	**3.997**	5.758	6.515	**3.233**

Table 4.1 provides the mean absolute error for all the joint angles over a time frame of 185 sec in real-time angle computation from detected keypoints compared with joint angles directly from the encoders (ground truth in this case). The accuracy of our model is better than DREAM for θ_1 and θ_4 whereas for the other two DREAM performs better.

An important factor to note for the error in Table 4.1 and angle plot in Figure 4.8 is that these results are only for angle values where DREAM provides a good keypoint detection.

There are many orientations of the Baxter arm where DREAM is unable to detect a keypoint with certain accuracy, resulting in a substantially inaccurate joint angle estimate (mostly of the order of 10^4) outside the optimizer bounds of $\pm\pi$. Figure 4.9 shows the angle (θ_3, θ_4) result for a different orientation where DREAM results are incorrect and joint detection unstable (time 0-17 sec and 24-34 sec). All these results were ignored in the calculation of the total MAE in Table 4.1.

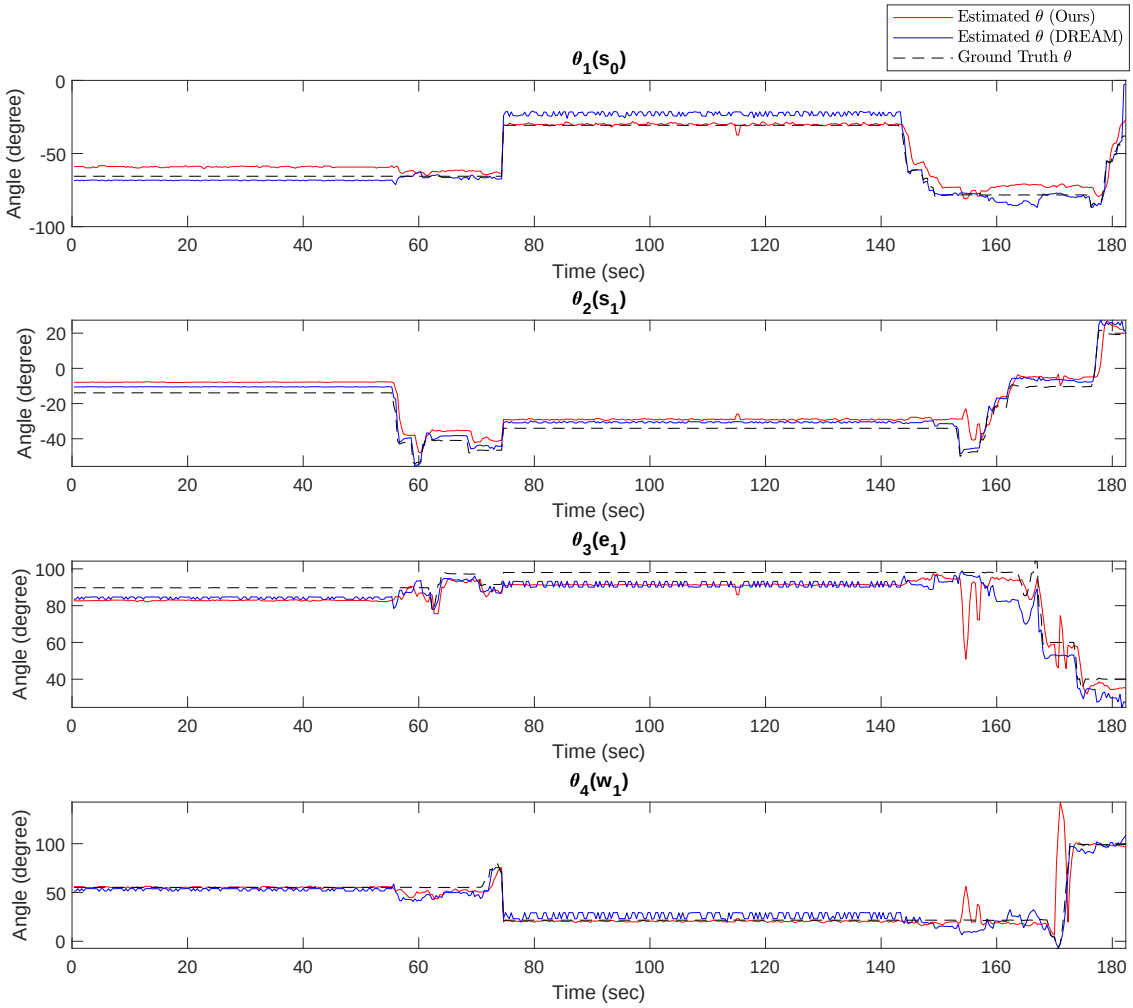


Figure 4.8: Joint angle vs time plot from our estimation, DREAM estimation, and Ground Truth values (Successive motor actuation) in real-time estimation of Baxter robot

Further, this comparison is just to show the accuracy of our method compared to other popular methods and cannot provide a true reflection of model performance as the training data for both methods vary largely. DREAM is trained on more than 10k synthetic images and uses transfer learning for implementation on real-world images, contrarily our model is trained on 650 images in a lab setting.

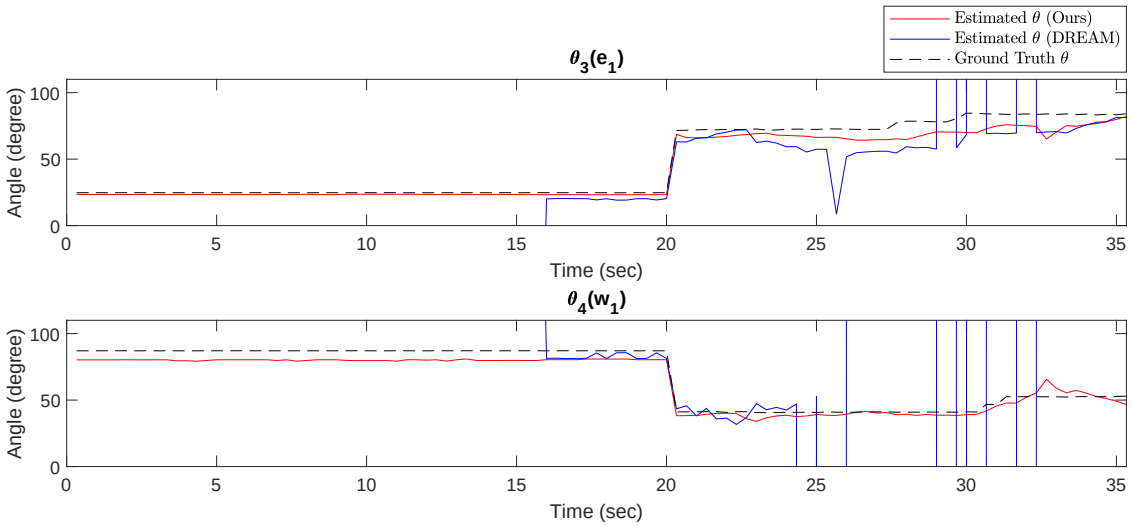


Figure 4.9: Joint angle vs time plot from our estimation, DREAM estimation, and Ground Truth values in real-time estimation of Baxter robot; where results from DREAM are highly inaccurate and not used in comparison metrics

4.4.2 Scale Excavator

In this case, the keypoint detection is not corresponding to the exact joint center. This requires the use of a second transformation from the joint center to the position of keypoint detection for g_{cw} in Equation (4.23). This is done by altering the URDF file with the measured dimensions on the physical system between the joint center and the detected keypoint to the best guess. The edited URDF (Appendix A) file is then used to compute the twist coordinated.

4.4.2.1 Ground Truth Angle Calculation

The excavator uses common DC motors with no servo feedback or encoders onboard the system. This makes the system only a one-way communication to receive a voltage for motor movement with no possible way to know its current position.

To generate a ground truth joint angle testbed for our model comparison, three Vicon markers are mounted on each link of the arm (Figure 4.10) and another four on the base (Figure 4.11) to compute the joint angles based on the angle between planes

Figure 4.10: Scale Excavator - Rear view with three reflective markers on each link of the arm manipulator for ground truth angle calculation

Figure 4.11: Scale Excavator - joint angles mentioned in green, reflective markers on the base for ground truth angle calculation

and 3D lines using the positions of the markers in a 3D world coordinate system. The motion capture system requires a minimum of three markers to define a rigid body, which in this case is each link. Also, these markers are positioned on the side not

visible from the camera frame, which makes them not a part of the training images and can be removed in the real-time implementation where error data is not logged.

Tables 4.2 to 4.5 provide the joint angle mean absolute error for the excavator in multiple cases from successive joint motion (one motor at a time, Figure 4.12) to the simultaneous joint motion of all four motors (Figure 4.15). In common practical motion cases, only two or three joints are actuated at a given time but these results focus on all possible conditions of motion.

Table 4.2: MAE (in degree) for each joint angle in real-time estimation of the excavator (Successive motor actuation)

	θ_1	θ_2	θ_3	θ_4
Unfiltered	2.270	0.836	2.405	5.192
Filtered	2.410	0.974	2.719	5.347

Table 4.3: MAE (in degree) for each joint angle in real-time estimation of the excavator (Multi motor actuation - 2 motors simultaneously)

	θ_1	θ_2	θ_3	θ_4
Unfiltered	3.020	1.437	2.411	4.316
Filtered	3.126	1.843	2.880	4.887

Table 4.4: MAE (in degree) for each joint angle in real-time estimation of the excavator (Multi motor actuation - 3 motors simultaneously)

	θ_1	θ_2	θ_3	θ_4
Unfiltered	4.832	1.072	3.276	5.188
Filtered	5.051	1.235	3.823	5.896

A common trend of increased error with an increase in the number of simultaneously moving motors is observed with a slight variation in the last case in Table 4.5. Further, we also use a Butterworth filter to smooth the output signal. More information on filter design and implementation is given in section Section 5.2, the results

Table 4.5: MAE (in degree) for each joint angle in real-time estimation of the excavator (Multi motor actuation - all motors simultaneously)

	θ_1	θ_2	θ_3	θ_4
Unfiltered	4.552	1.346	2.353	4.899
Filtered	4.861	1.656	2.897	5.724

are provided here as a comparison of the unfiltered signal with the one after using the Butterworth filter. The filter causes a further time shift as the signal responds with a delay in the joint detection as seen in Figures 4.12 to 4.15, but provides a smooth transition from one angle to the other without noise in the angle output.

To the best of the authors' knowledge and at the time of writing this thesis, no other joint estimation technique is implemented on the same system or a similar dataset. Therefore, no direct comparison of our results can be made with other publications. Table 4.6 only presents results from the systems closely related to an excavator or a similar 4-DOF system.

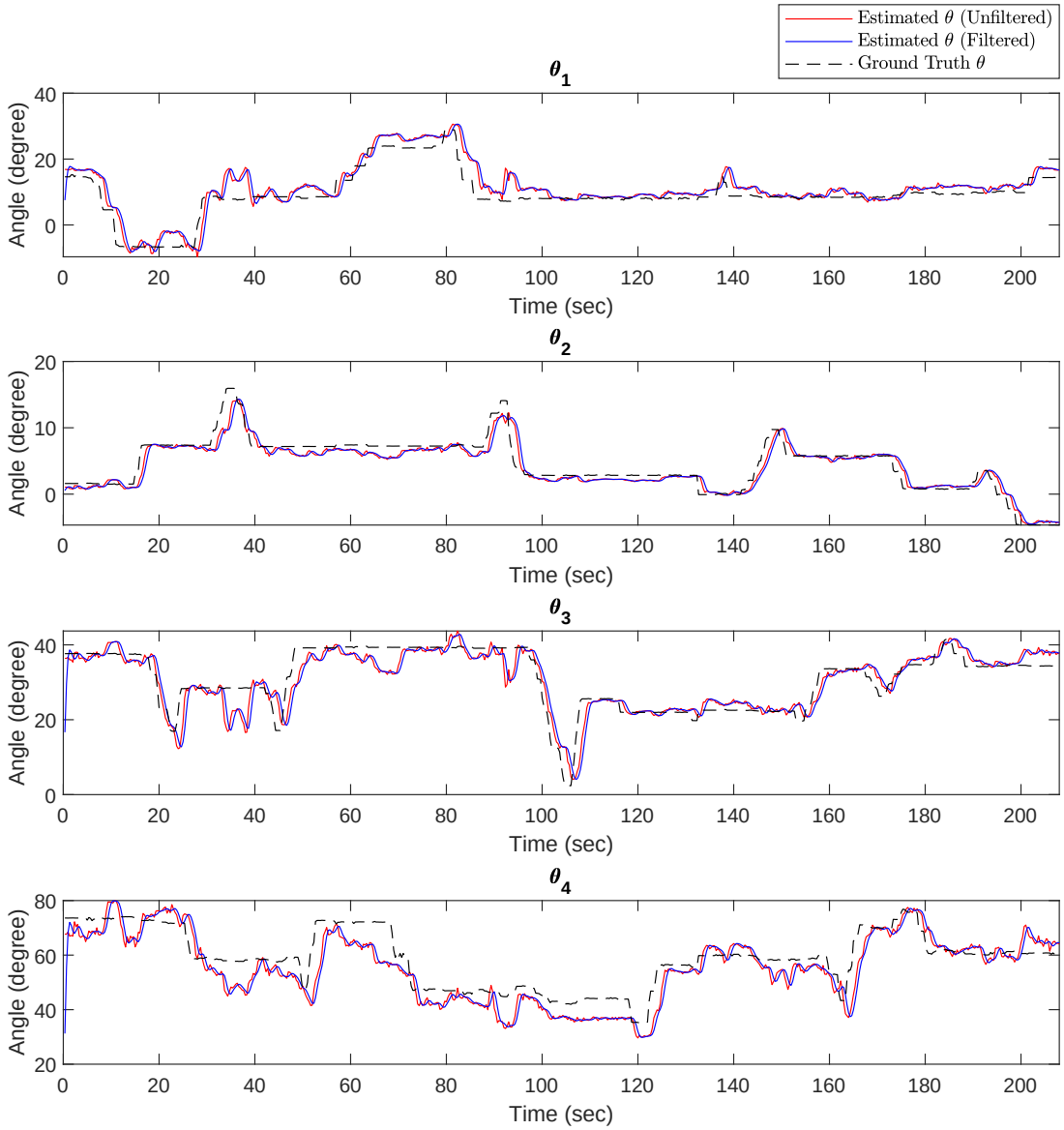


Figure 4.12: Joint angle vs time plot from Estimated, Refined/Filtered, and Ground Truth values in real-time estimation of the excavator (Successive motor actuation)

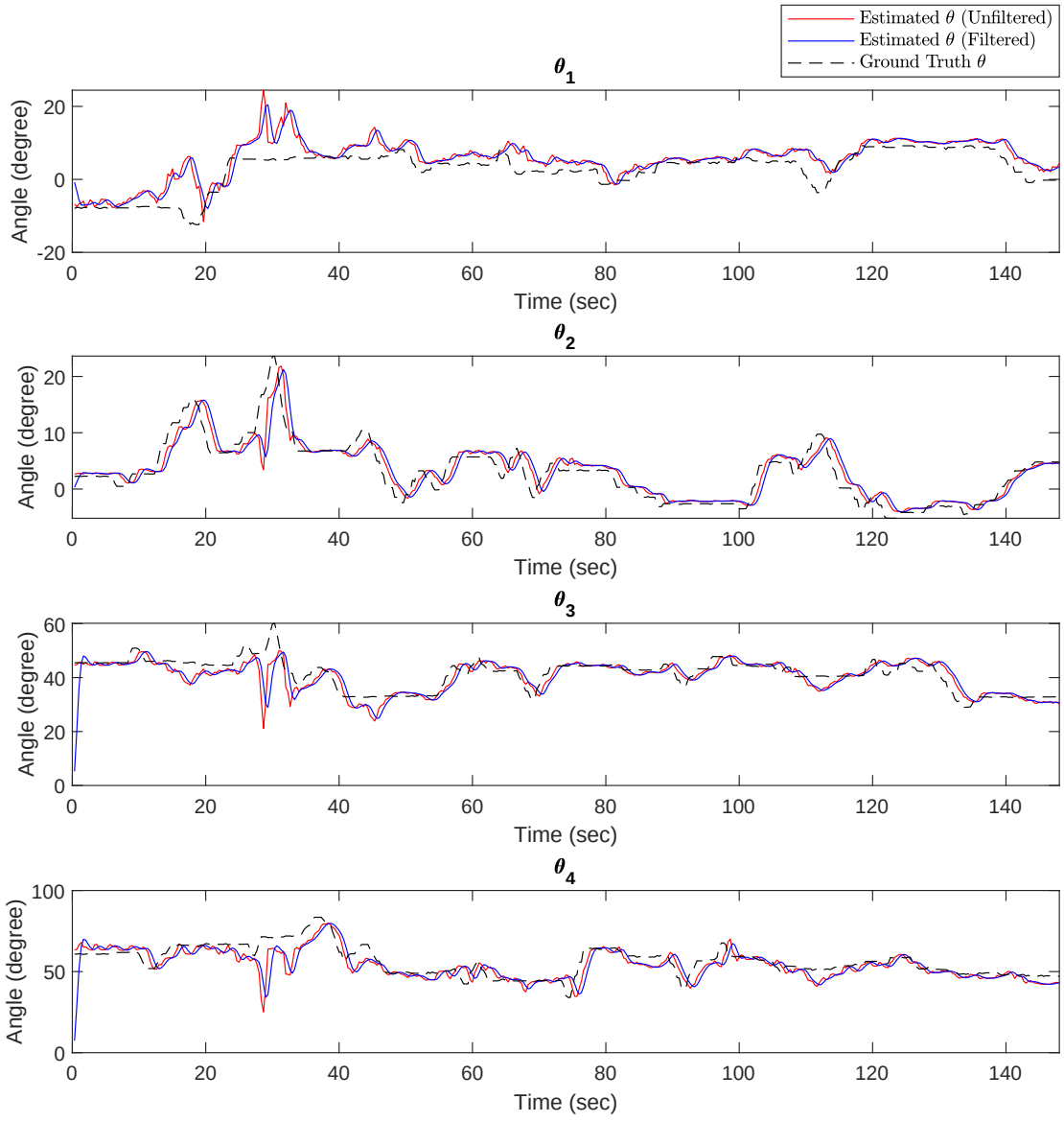


Figure 4.13: Joint angle vs time plot from Estimated, Refined/Filtered, and Ground Truth values in real-time estimation of the excavator (Multi motor actuation - 2 motors simultaneously)

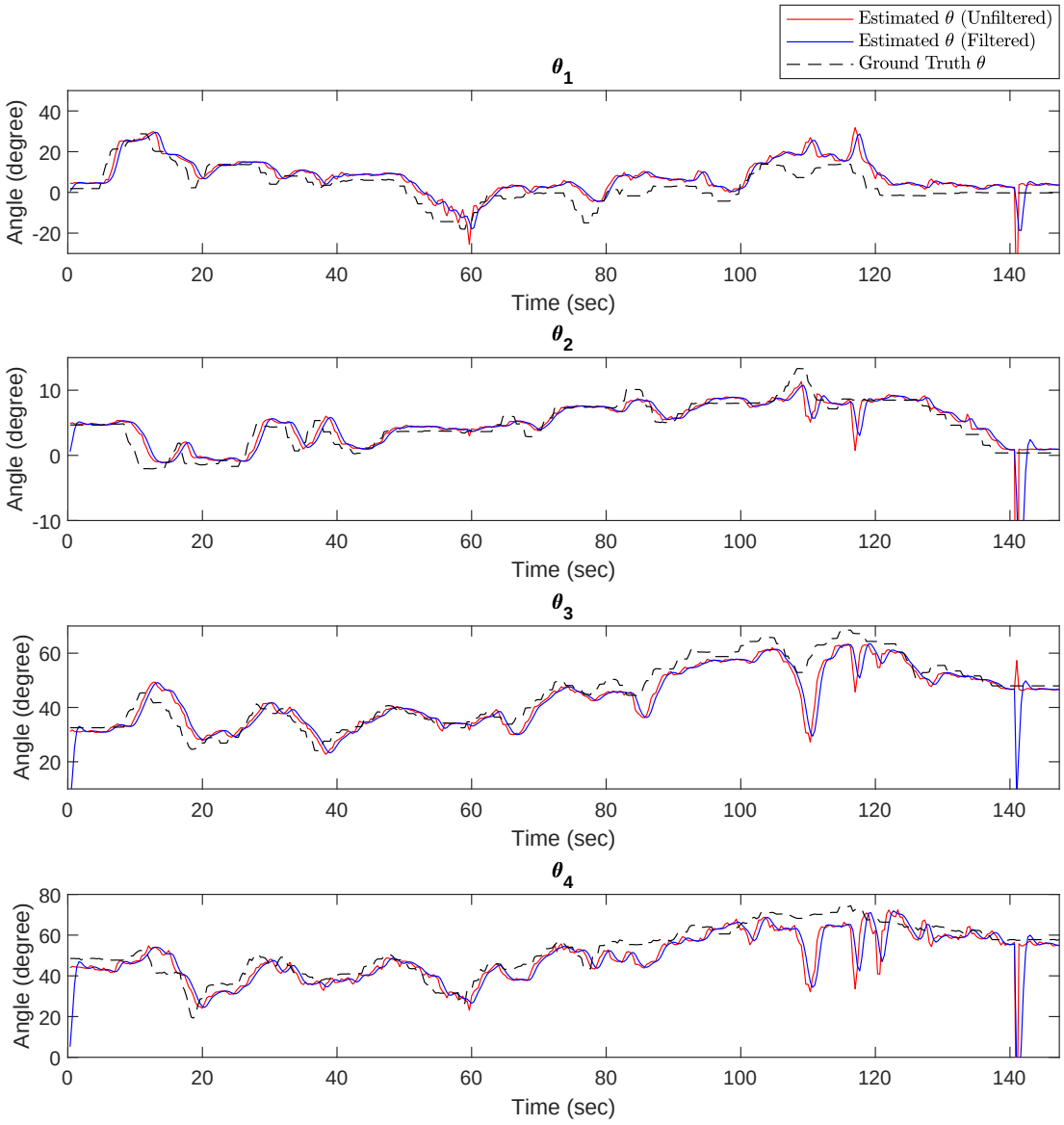


Figure 4.14: Joint angle vs time plot from Estimated, Refined/Filtered, and Ground Truth values in real-time estimation of the excavator (Multi motor actuation - 3 motors simultaneously)

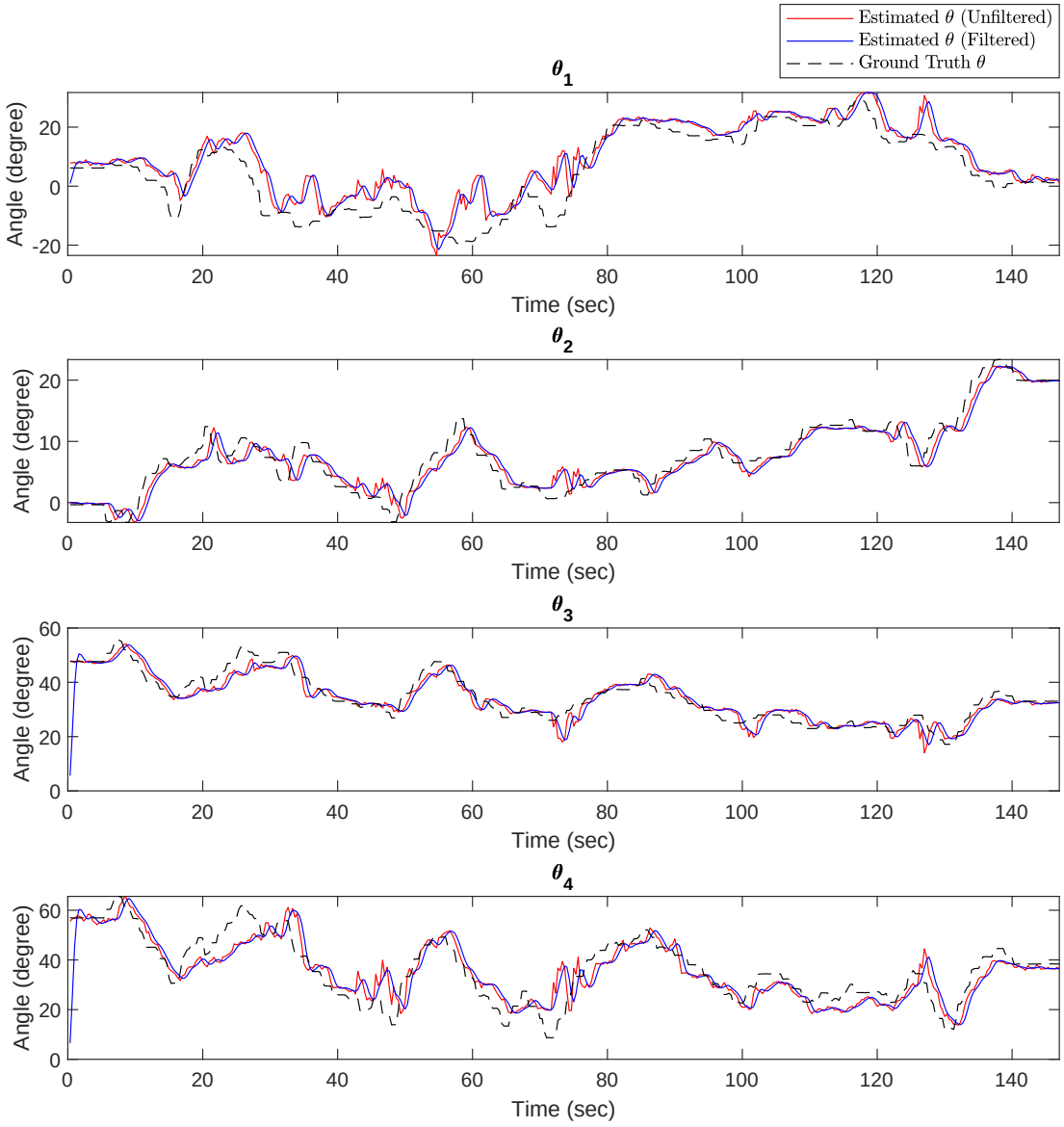


Figure 4.15: Joint angle vs time plot from Estimated, Refined/Filtered, and Ground Truth values in real-time estimation of the excavator (Multi motor actuation - all motors simultaneously)

Table 4.6: Review of different detection/angle estimation studies and a comparison with ours

Method	**Description**	**Results**
CRAVES [8]	4-DOF robot; 17 keypoints; joint angle and camera parameter estimation	Joint angle error: 7.13° (synthetic) and 4.18° (refined) in offline image processing
RoboPose [9]	multiple robots; render and compare method; joint angle and camera parameter estimation	Joint angle error 5.4° in offline image processing
J. Zhao et al. [62]	Industrial Excavator; marker-based detection; 3D pose estimation	Position Error: 22mm, Orientation Error: 8.5°
S. Zhang et al. [70]	Industrial Excavator; 10 keypoints; 2D pose estimation	Keypoint AP:0.965
C.-J. Liang et al. [71]	4-DOF KUKA robot with bucket attachment; 4 keypoints; 2D and 3D pose estimation	3D position error: 144.65 mm (mean for 3 detected positions; offline processing)
J. Xu et al. [66]	Virtual excavator; cylinder displacement measurement	End-effector maximum error: 6.02-19.09 cm (Horizontal-Vertical)
VECTOR (ours)	Scale Excavator; 4 keypoints; joint angle estimation	Keypoint AP: 0.987, Joint angle error: 3.28° (real-time processing)

Chapter 5

Control System Design and Implementation

With the joint angle estimation using the vision algorithm combined with the optimization, the 3D pose estimation problem is addressed. This chapter extends to the use of joint angle estimation to solve the control problem. In most manipulators, the simplest task of moving a manipulator from an initial point to a target end effector pose/final point is a common task. Here we first design a digital PID controller for control of the robot manipulator using commonly available open-source microprocessors and then present a way to combine our modular detection and control approach with a ROS-based path planning algorithm (MoveIt) for easy extension of our system to sophisticated motion planning and collision avoidance architectures.

5.1 Modular Control System Design

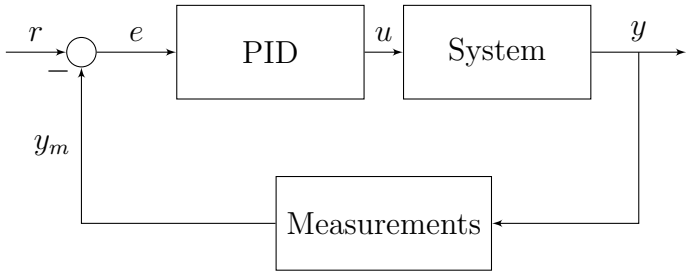


Figure 5.1: A general PID control structure

For point-to-point motion or a pick-and-place operation, the required control is a position control system. This control can be simplified to a joint angle control where we use a set of desired/reference joint angles ($\boldsymbol{\theta_r}$) and compare them with the current measured joint angles ($\boldsymbol{\theta_m}$), the error ($\boldsymbol{e_\theta}$) is fed to the controller which provides an input to the system, here a **PWM** (Pulse Width Modulation) signal. This setup is shown in Figure 5.2 with an indirect connection between the motors (robot manipulator) and the measurement sensor (vision algorithm) as contrary to a sensor on the robot manipulator, an external neural network-based joint computation pipeline is used. In our case, as we control four joint angles and four distinct motors therefore all the parameters $\boldsymbol{\theta_r}$, $\boldsymbol{\theta_m}$, $\boldsymbol{e_\theta}$, **PWM** are $\in \mathbb{R}^{4\times1}$.

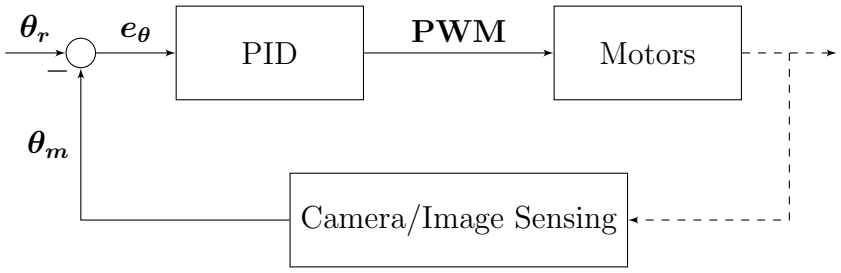


Figure 5.2: Control Structure in our case where the dashed line between motor output and camera/vision model represent no direct physical feedback through a sensor but rather an external camera-based angle estimation

The PID control is designed as a digital control in Python as a ROS node. The PID controller is chosen for its simple implementation in this conceptual proof. While this control system is adequate for simple problems the ROS node can be easily tweaked to use a different and more complex control. Here we solve for four joint angles and three control gains (k_p, k_i, k_d) for each which gives a total of 12 gain values that need to be tuned.

5.2 Filter Design

A signal processing technique is used here to provide a steady and smooth transition between joint estimations rather than a noisy output. Keypoint pixel detection jittering in real time causes the optimizer to compute varying joint angles for a stationary scene. To prevent this noisy signal for feedback to the controller a filter design is explored to minimize this effect on the controller output to the system, in this case, the PWM signal to the motor.

The first step in the design of a filter is to calculate the FFT (Fast Fourier Transform) for the signal, which here is all the measured joint estimations ($\boldsymbol{\theta_m}$). The FFT provides a graph of all the frequencies present in the signal along with the magnitude for each on the y-axis. Figure 5.3 is the Discrete Fourier Transform (DFT) using the FFT algorithm (`fft()` function in MATLAB) for the θ_1 in the case of the excavator. It is clear that the signal is made up of very low frequencies close to 0 and diminishes to a low value for higher frequencies.

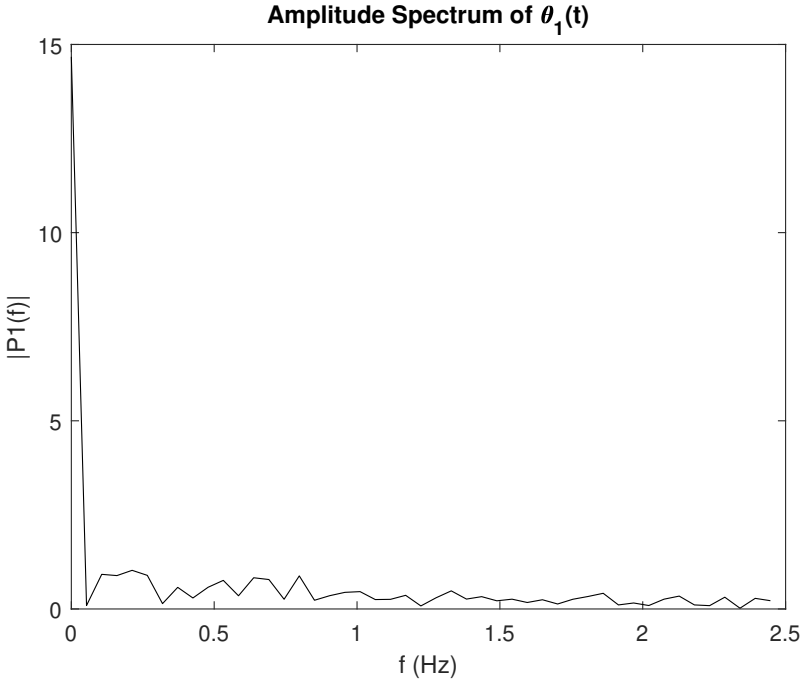


Figure 5.3: Amplitude spectrum plot for estimated θ_1 of the excavator using a Discrete Fourier Transform

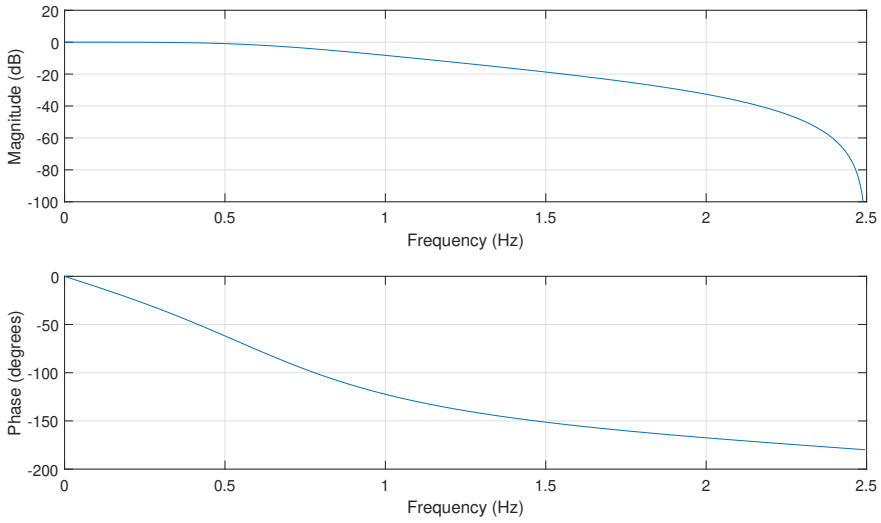


Figure 5.4: Bode plot for the designed Butterworth filter

We design a simple second-order low-pass Butterworth filter to eliminate the higher frequencies' contribution. A general analog low-pass Butterworth filter is given as a magnitude-squared function,

$$| H_{LP}(j\Omega) |^2 = \frac{1}{1 + \Omega^{2N}} \tag{5.1}$$

where N is the order of the filter. We design the filter with a cutoff frequency of 0.7 Hz using the `butter()` function in MATLAB which gives us the transfer function of the filter (Equation (5.2)) in discrete time as required for a digital filter. Figure 5.4 shows the bode plot of the designed filter with the required cutoff frequency.

$$H(z) = \frac{0.1174 + 0.2347z^{-1} + 0.1174z^{-2}}{1 - 0.8252z^{-1} + 0.2946z^{-2}} \tag{5.2}$$

Figure 5.5 provides the comparison of the filtered signal with the initial signal output angle. Note the slight time shift between the unfiltered signal with the one after being filtered, this is a common problem as the filter is a causal system and can work only based on past information, this results in a small latency in the response.

The same filter parameters are used for all four angle values as the same cutoff frequency is used in all cases.

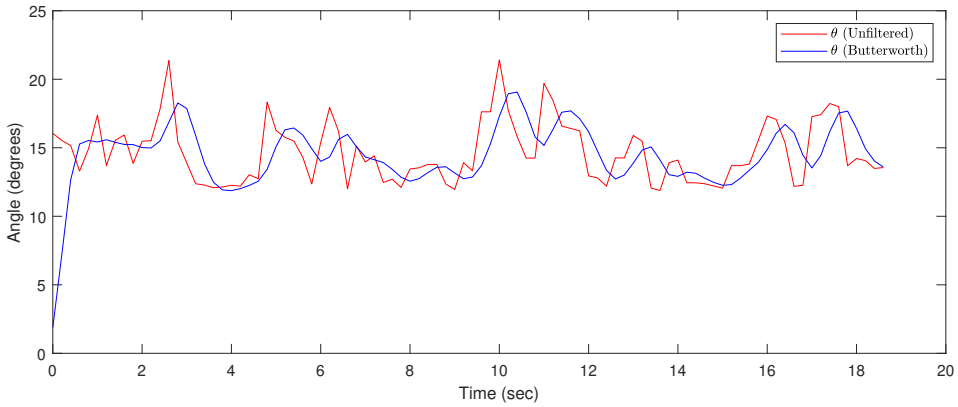


Figure 5.5: Joint angle (θ_1) vs time plot from unfiltered signal and signal after Butterworth filter implementation which has a smother output with lower noise

5.3 ROS Integration

The control system works on the error signal between the reference and measured angle, but in a point-to-point control problem, we only have the end effector's final position. This final pose needs to be converted to joint states for the manipulator using inverse kinematics.

One conventional way to move forward is to write a custom inverse kinematic (IK) solver to compute the angles using trigonometric relations. Another advanced way is to use ROS and MoveIt which provides a very interactive IK solver in RViz (a robot simulation environment for ROS). MoveIt here provides a path-planning algorithm to give successive joint angle signals to reach the final orientation and position and RViz shows the planned path in simulation.

To use this strategy we require a CAD file of the robot in our case the excavator along with dimensions and joint limits. All the kinematics information needed for MoveIt is stored in a special XML file known as a URDF file.

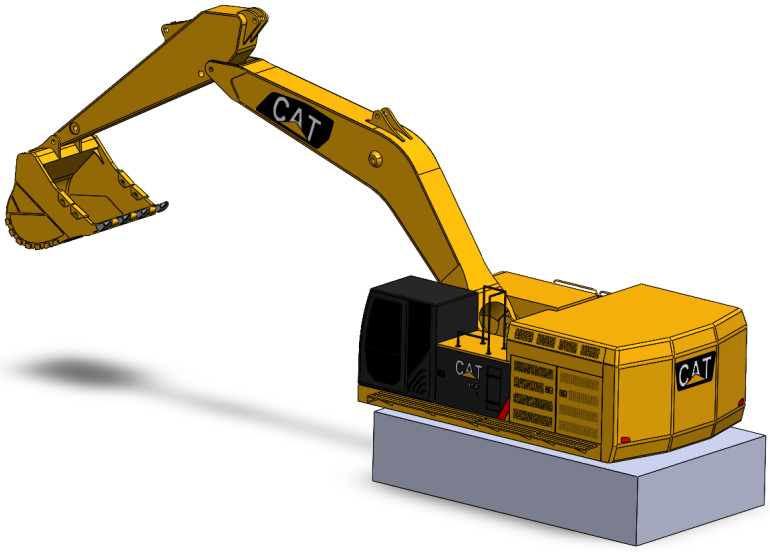


Figure 5.6: Excavator CAD model in home configuration (for the URDF) with edited dimensions and a fixed base to replicate our physical system and constraints

5.3.1 Excavator 3D CAD Model Design

To design the CAD file for the excavator an open-source Caterpillar 390F excavator model[1] was edited in SOLIDWORKS to represent the exact dimensions of the 4-DOF excavator arm in the study. It is important to note that a very simple system with planar links can also be used to represent the model and its geometrical relation between links but this model is used to represent the excavator motion realistically in simulation. In no part, this CAD model has a contribution in training the keypoint detection model conversely to other popular keypoint models.

This model in SOLIDWORKS is then used to build the URDF file using the SW2URDF plugin in SOLIDWORKS by manually naming all the joints and links.

After obtaining the CAD file and URDF, the next step is to build a custom robot setup in MoveIt Setup Wizard and define all the arm links, joints, and home configurations. This provides us with a special ROS package for the specific robot with all the required files.

We use the `demo.launch` file to run all the required ROS nodes with a minor change

[1] https://grabcad.com/library/caterpillar-390f-1

with the \joint_state ROS topic which is replaced with the joint states obtained from the vision algorithm from our measured angles. This change is directly reflected in RViz where the current orientation of the excavator can be seen and any planning group node can be activated to move from the current position and orientation.

Another advantage of integrating MoveIt into our system is the easy extension of simple point control to exploit the full range of path planning algorithms with object detection and collision avoidance for more complex motion paths.

5.4 Experimental Procedure

To implement the discussed ROS controls on the excavator, a change in the hardware setup to control the motors that move the links is required. For this, the central controller on board is replaced with an Arduino Uno Wifi board with Adafruit Motor Shield that can control four DC motors as required in this case. The Arduino board is connected to the central processor using a USB cable to facilitate communication between the central processor and motor actuation.

To establish ROS communication with Arduino, a rosserial_arduino protocol is used to send serialized messages and topics to the board. As no feedback is available from the motor or the board, a one-way communication of the PWM signal from the digital PID controller for all four joints is sent from the central PC via a USB port with a 115200 baud rate to the Arduino board for actuating the motors. The feedback loop is through the camera-based video stream collecting data to estimate joint angles.

The biggest challenge in the design of a controller for this application is tuning the control gains is not very straightforward. A basic model identification cannot be applied and for the transferability of the control to any other robot, no mathematical relations are used here. Other physical tests such as the drop test were also not performed to limit a very harsh motor movement that might result in damage to the motor. Additionally, as the joint is not directly controlled by motors but with

a rather intricate system of screw motion attached at one end to the motor and the other to a position on the link not corresponding to the exact location of the joint the exact relation between motor rotation and arm angle is nonlinear. Due to all these constraints only manual tuning of gains is performed.

5.5 Joint Control Assessment

The setup is the same as in the previous chapter where the camera is facing head-on to the robot arm. We only design our controller for the excavator and real-time assessment is also done for the same. We ignore any control on the Baxter as the major focus was to verify the detection on Baxter (where encoder-based angles can be found) and transfer the tools to an unknown angle environment; the excavator. More than 20 random positions are chosen in the operational range of the excavator that forms the initial and final positions of the excavator for the control validation study. We use `simple_pid` package to implement the control equations. After manually tuning the gain values the gains are set as $k_p = 180$, $k_i = 0.5$, $k_d = 2$ for $\theta_1, \theta_2, \theta_3$, and $k_p = 180$, $k_i = 0.8$, $k_d = 2$ for θ_4. Further, a set of physical constraints are also applied to the control output (PWM signal). The max-min PWM signal is chosen as $[-200, 200]$ rather than the maximum operation PWM limit of $[-255, 255]$ to limit a very aggressive motion at full capacity. The motors also have a deadband at low PWM values ($< \pm 50$) due to motor friction and friction in the screw. To overcome this problem of no-motion zone even when the error is still significant a set of conditional statements is proposed as discussed in Algorithm 2.

5.5.1 Results and Discussion

The entire pipeline is verified for point-to-point motion control using RViz and MoveIt. The major control goal is to reach a desired position and orientation in the least possible time. The camera is fixed as in the rest of the cases. The random positions are selected to cover a wide range of points in the operational and detectable range

Algorithm 2 Joint angle control for robot arm manipulator with constraints

Input: Reference and measured angles ($\boldsymbol{\theta_{ref}}, \boldsymbol{\theta_m}$)
Output: Control signal to manipulator (**PWM**)

procedure PID CONTROL($\boldsymbol{\theta_{ref}}$,$\boldsymbol{\theta_m}$)
    `//Initialization`
    PID.limits $\leftarrow \pm 200$ ▷ Forward/Backward Maximum Speed
    $k_p, k_i, k_d \leftarrow$ gains
    $\varepsilon_{tol} \leftarrow 0.05$ ▷ 0.05 rad
    $j \leftarrow 1$
    `//Control signal computation`
    while $j \leq 4$ **do** ▷ For 4 joints
        $\varepsilon^j \leftarrow \theta^j_{ref} - \theta^j_m$
        $\text{PWM}^j \leftarrow \text{PID}(\varepsilon^j, k_p, k_i, k_d)$
        if abs(PWM^j) < 50 **then**
            if abs(PWM^j) ≥ 30 **then**
                if $\text{PWM}^j > 0$ **then**
                    $\text{PWM}^j \leftarrow 50$
                else
                    $\text{PWM}^j \leftarrow -50$
                end if
            else
                $\text{PWM}^j \leftarrow 0$
            end if
        end if
        if $\theta^j_{ref} - \theta^j_m \leq \varepsilon_{tol}$ **then**
            $\text{PWM}^j \leftarrow 0$
        end if
        $j \leftarrow j + 1$
    end while
end procedure

of the camera.

For the entire test run, reference angles ($\boldsymbol{\theta_r}$) from MoveIt (`move_group/fake_controller_joint_states`), measured angles ($\boldsymbol{\theta_m}$) from our vision-based estimation, and ground truth angles ($\boldsymbol{\theta_{gt}}$)from the motion capture system are recorded to collect quantitative results. These results for two successive orientation control are shown in Figure 5.8. After obtaining the results graphically, the absolute error $|\boldsymbol{\theta_{gt}} - \boldsymbol{\theta_r}|$ (ground truth and reference) and $|\boldsymbol{\theta_m} - \boldsymbol{\theta_r}|$ (measured and reference) is recorded in

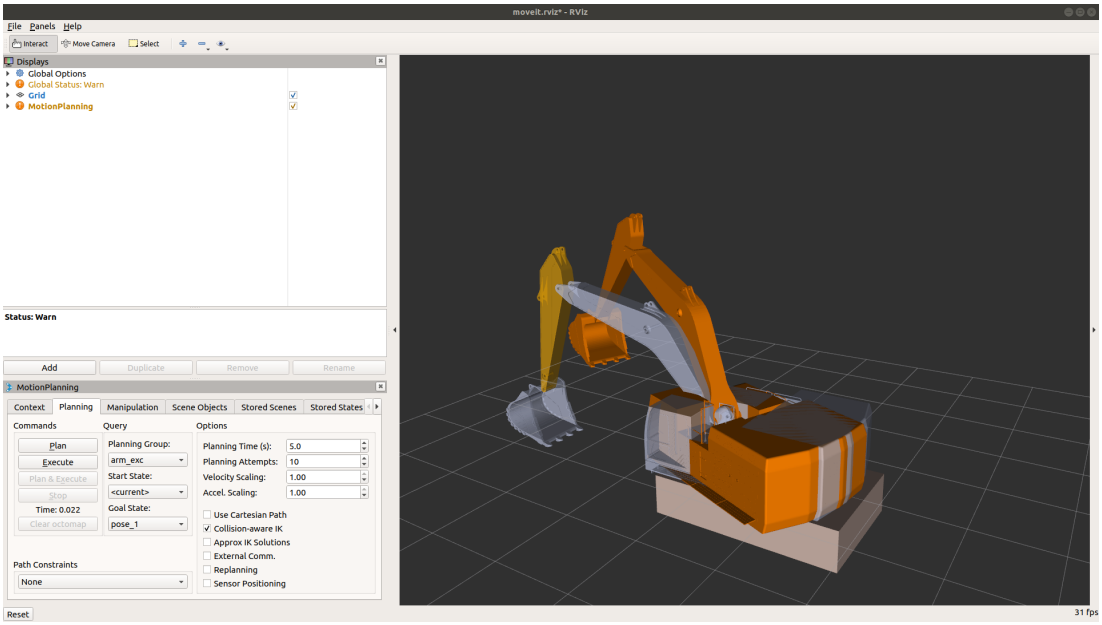


Figure 5.7: Excavator simulation in RViz environment for simplified motion planning, the translucent body represents current excavator position and the solid body represents the final target position for a particular planning scene

steady state conditions i.e. the error after the control cycle is complete. These results are reported in Table 5.1 where the average error for the final angle obtained and the reference is 3.7° with a maximum of 7.4° for θ_4. The error compared to the measured angle from vision algorithm is also reported showing the joint angles values used in the controller.

Table 5.1: Steady state error (in degree) for each joint angle in real-time control of the excavator, comparison of ground truth and estimated angle with the reference angle shows the accuracy of the final position achieved

Error	$\|\boldsymbol{\theta_{gt}} - \boldsymbol{\theta_r}\|$	$\|\boldsymbol{\theta_m} - \boldsymbol{\theta_r}\|$
$\boldsymbol{\theta_1}$	2.393	1.675
$\boldsymbol{\theta_2}$	0.716	0.674
$\boldsymbol{\theta_3}$	4.308	1.405
$\boldsymbol{\theta_4}$	7.367	3.630
Average	**3.696**	**1.846**

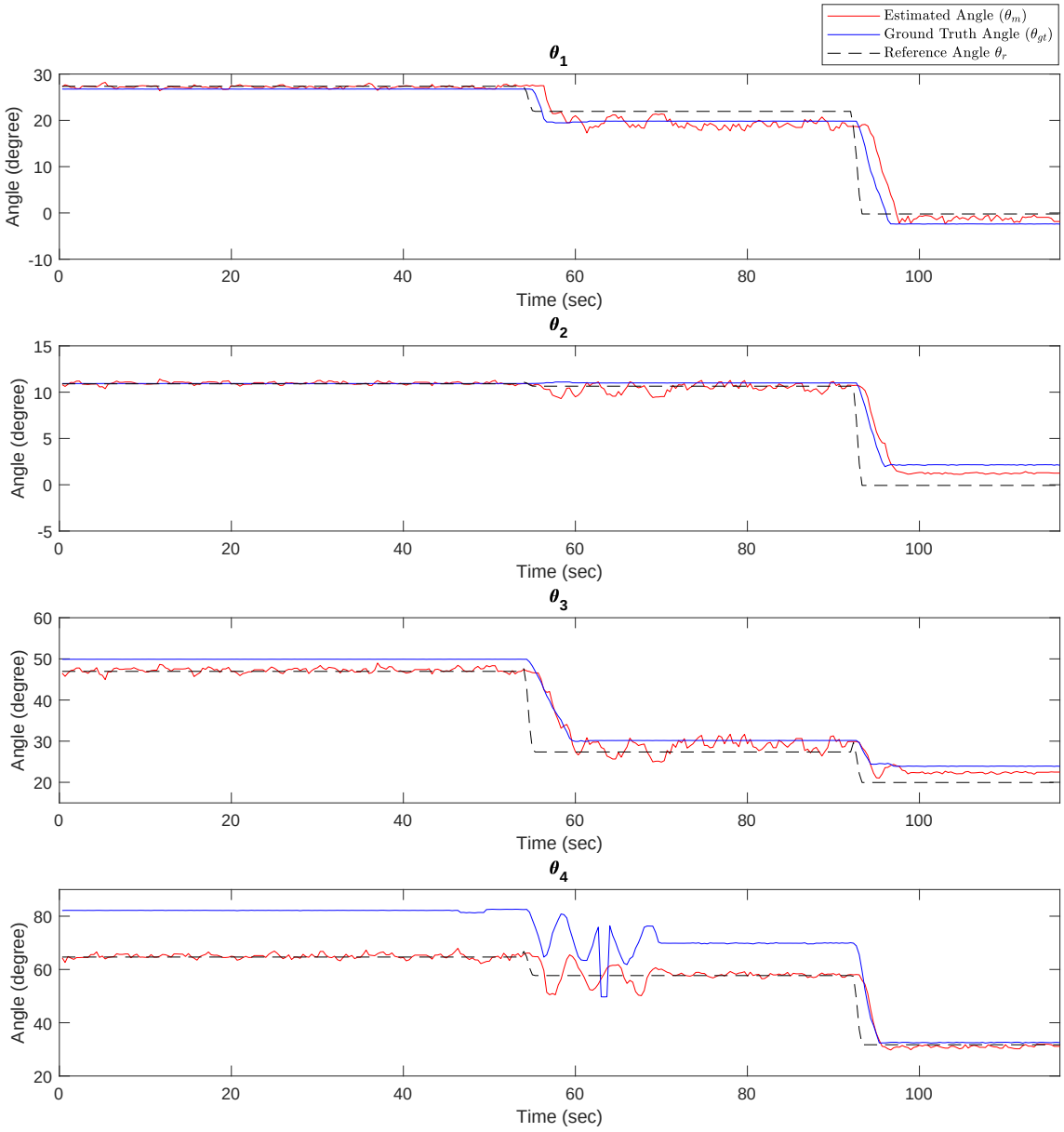


Figure 5.8: Joint angle vs time plot from Estimated, Ground Truth and Reference values in real-time control of the excavator

Finally, we also report a few other parameters for comparison with other published results on a similar system. Table 5.2 records the end effectors' mean absolute 3D position error based on the reference end effector position. The success rate, in this case, is the percentage of times when all the joints reach a final angle in close proximity to reach the required end effector position and remain stationary thereafter.

In some cases, the joint angles keep on fluctuating around the reference and are unable to reach a constant final stationary position; shown in Figure 5.9. This is caused by the latency (≈ 1.33 sec) in the joint computation by our framework, resulting in an active control action even when the desired angle value has been achieved, the amplitude of this periodic fluctuation is around $\pm 4 - 5^{\circ}$. The average time is reported as the time taken to achieve steady state conditions after a new orientation is broadcasted to the controller.

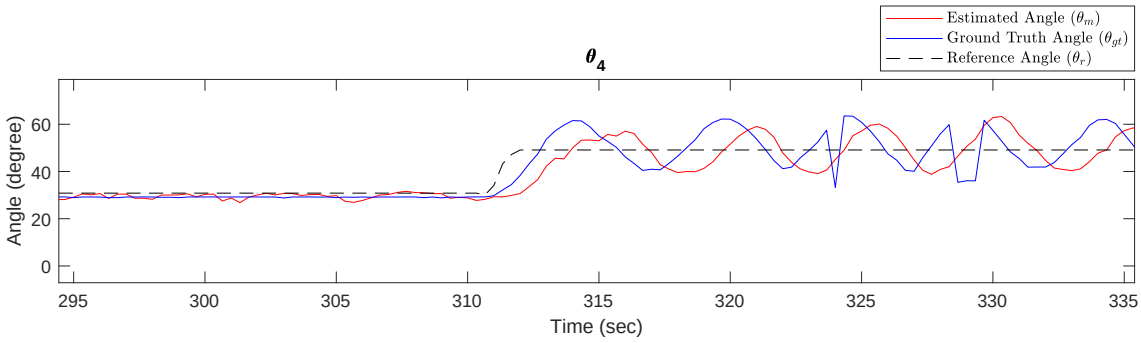


Figure 5.9: Joint angle (θ_4) vs time plot from Estimated, Ground Truth and Reference values in real-time control of the excavator, showing an oscillatory steady state with a phase difference between the estimated and ground truth angles

Table 5.2: Quantitative results for the position control problem in our case, distance error is the average steady-state error and average time is the time required to achieve steady-state

Agent	**Input Type**	**Distance Error (cm)**	**Success Rate**	**Average Time (s)**
VECTOR (Ours)	Camera	1.78	88.6%	6.5

As stated previously, to the best of the authors' knowledge no other keypoint detection and control algorithm is performed on the same robot arm or any other that closely resembles this system. A 4-DOF encoder-free robot arm control was performed in CRAVES [8], the control results of which are presented in Table 5.3 to give a slight sense of comparison of our results with a state-of-the-art system. CRAVES results are not directly comparable with our results due to many differences, one of the major ones being the use of completely different robot arms and training sets. Further, CRAVES results are based on moving the end effector above a set of certain reference points which is why they were able to provide a comparative study between human performance and the algorithm they used. Conversely, in our case, we chose random positions for control where a comparative study with human performance is not easy to make.

Table 5.3: Quantitative results from CRAVES [8] for completing a reaching task problem

Agent	**Input Type**	**Distance Error (cm)**	**Success Rate**	**Average Time (s)**
Human	Direct	0.65	100%	29.8
Human	Camera	2.67	**66.7**%	38.8
CRAVES	Camera	**2.66**	59.3%	**21.2**

Chapter 6

Conclusion and Future Work

6.1 Summary of Thesis

This thesis was successful in the development of a control strategy for an excavator model using purely vision-based sensing along with a digital PID controller. The entire work serves as a proof of concept for the possibility of using the developed toolchain to provide a solution for the autonomous control of sensorless manipulators, without requiring modifications or additions to the existing hardware.

The vision-based system provides the excavator's joint localization using keypoint detection for pose estimation in the camera frame using only a single RGB image frame. A special extension of the Mask R-CNN [95] network is used for the keypoint location which was originally proposed for human keypoints. This model is trained on a limited number of images collected in the lab environment. Real-time qualitative and quantitative test results were presented and compared with state-of-the-art methods.

The joint angle optimization algorithm was proposed to establish the 3D pose from the available camera and geometric information. Ground truth joint angle information was obtained using an indoor motion capture system. Again, a qualitative assessment was recorded.

A simple digital controller (PID) was established using a ROS MoveIt-based inverse kinematic solution and visualization of the path-planning system. This simple

position control using motor actuation provides a transferable setup with reusable modules which can be adapted to a different robot manipulator. Finally, our control results were compared with the related work performed in [8] (though the direct comparison is not possible due to the use of different robot manipulators), and our framework was shown to perform better in position control of the end-effector.

6.2 Limitations of Work

While the results presented in each stage of the thesis show improved performance relative to existing work, a few limitations were observed:

- Due to low variation in training images, as all datasets were collected in the lab environment and with restriction on the maximum angular motion to avoid occlusion of keypoints, results at this stage cannot be directly used for an industrial excavator in a construction environment.

- As we tried to use minimal keypoints, we have no redundant information for fine-tuning the joint computation or estimating the camera-to-robot base transformation, i.e., no extrinsic knowledge can be extracted.

- Precise position control could not be achieved due to the motor actuators on the scale excavator, which exhibited significant deadband and backlash, and furthermore the entire toolchain still exhibits latency in processing a new input image such that the controller works on delayed signals. This is reflected in the estimation plots as a time shift of the detected joint angles as compared to the ground truth.

6.3 Future Work

The work in this thesis can be used as a basis for many prospective projects. To reiterate, the work presented is a proof of concept and can be ported for use in a

more complex setting and eventually for real excavator control. Field testing of the detection system is one of the first steps toward the possibility of autonomous control in a job site.

More advanced ways to obtain the extrinsics in real-time which do not involve more keypoints can be explored; one possible method is proposed in [105].

Upgrades to the current software architecture to reduce the computation time of the pipeline and the development of a better control system for the excavator model using higher-quality motors for enhanced position control are the logical immediate next steps to improve overall performance.

Also, this entire study can be extended to include object avoidance, perform more sophisticated path planning (already supported within MoveIt), and integration with vision-based object detection and grasping. Together these would lead to fully autonomous operation of encoderless manipulators, such as excavators or cranes.

Bibliography

[1] P. Mesmer, M. Neubauer, A. Lechler, and A. Verl, "Robust design of independent joint control of industrial robots with secondary encoders," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102 232, 2022, ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2021.102232.

[2] U. Schneider *et al.*, "Improving robotic machining accuracy through experimental error investigation and modular compensation," *The International Journal of Advanced Manufacturing Technology*, vol. 85, no. 1-4, pp. 3–15, Jul. 2016, ISSN: 0268-3768. DOI: 10.1007/s00170-014-6021-2.

[3] K.-Y. Peng and J.-Y. Chang, "Effects of assembly errors on axial positioning accuracy for rotating machinery with magnetoresistance-based magnetic encoders," *Microsyst. Technol.*, vol. 27, no. 6, 2507–2514, 2021, ISSN: 0946-7076. DOI: 10.1007/s00542-020-05174-0.

[4] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," Sep. 2018. DOI: 10.48550/arxiv.1809.10790.

[5] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, "Deep learning sensor fusion for autonomous vehicle perception and localization: A review," *Sensors 2020, Vol. 20, Page 4220*, vol. 20, p. 4220, 15 Jul. 2020, ISSN: 1424-8220. DOI: 10.3390/S20154220.

[6] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," British Machine Vision Association, Dec. 2015, pp. 41.1–41.12, ISBN: 1-901725-53-7. DOI: 10.5244/C.29.41.

[7] T. E. Lee *et al.*, "Camera-to-Robot Pose Estimation from a Single Image," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 9426–9432, May 2020, ISSN: 10504729. DOI: 10.1109/ICRA40945.2020.9196596.

[8] Y. Zuo, W. Qiu, L. Xie, F. Zhong, Y. Wang, and A. L. Yuille, "Craves: Controlling robotic arm with a vision-based economic system," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 4209–4218, Jun. 2019, ISSN: 10636919. DOI: 10.1109/CVPR.2019.00434.

[9] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Single-view robot pose and joint angle estimation via render & compare," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1654–1663, 2021, ISSN: 10636919. DOI: 10.1109/CVPR46437.2021.00170.

[10] D. B. Chakraborty, M. Sharma, and B. Vijay, "Controlling by showing: I-mimic: A video-based method to control robotic arms," *SN Computer Science*, vol. 3, pp. 1–15, 2 Mar. 2022, ISSN: 2662-995X. DOI: 10.1007/S42979-022-01014-2/TABLES/3.

[11] B. Xie, "Applications of computer vision and machine learning to three engineering problems," Master's thesis, University of Alberta, 2021.

[12] M. Han, "Vision-based methods for joint state estimation of robotic manipulators," Master's thesis, University of Alberta, 2021.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2323, 11 1998, ISSN: 00189219. DOI: 10.1109/5.726791.

[14] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[15] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguist. Vol. 1, Long Pap.*, vol. 1, pp. 1107–1116, 2001 2016.

[16] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 3462–3471, Nov. 2017. DOI: 10.1109/CVPR.2017.369.

[17] T. Kautz, B. H. Groh, J. Hannink, U. Jensen, H. Strubberg, and B. M. Eskofier, "Activity recognition in beach volleyball using a deep convolutional neural network: Leveraging the potential of deep learning in sports," *Data Mining and Knowledge Discovery*, vol. 31, pp. 1678–1705, 6 Nov. 2017, ISSN: 1573756X. DOI: 10.1007/S10618-017-0495-0.

[18] S. Zhao, L. Yin, J. Zhang, J. Wang, and R. Zhong, "Real-time fabric defect detection based on multi-scale convolutional neural network," *IET Collaborative Intelligent Manufacturing*, vol. 2, pp. 189–196, 4 Dec. 2020, ISSN: 2516-8398. DOI: 10.1049/iet-cim.2020.0062.

[19] J. F. Jing, H. Ma, and H. H. Zhang, "Automatic fabric defect detection using a deep convolutional neural network," *Coloration Technology*, vol. 135, pp. 213–223, 3 Jun. 2019, ISSN: 1478-4408. DOI: 10.1111/COTE.12394.

[20] D. I. Kosmopoulos, A. S. Voulodimos, and A. D. Doulamis, "A system for multicamera task recognition and summarization for structured environments," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 161–171, 1 Feb. 2013, ISSN: 1551-3203. DOI: 10.1109/TII.2012.2212712.

[21] G. Wang, L. Tao, H. Di, X. Ye, and Y. Shi, "A scalable distributed architecture for intelligent vision system," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 91–99, Feb. 2012, ISSN: 15513203. DOI: 10.1109/TII.2011.2173945.

[22] R. P. Loce, E. A. Bernal, W. Wu, and R. Bala, "Computer vision in roadway transportation systems: a survey," *Journal of Electronic Imaging*, vol. 22, no. 4, p. 041 121, Dec. 2013, ISSN: 1017-9909. DOI: 10.1117/1.JEI.22.4.041121.

[23] R. Rad and M. Jamzad, "Real time classification and tracking of multiple vehicles in highways," *Pattern Recognition Letters*, vol. 26, no. 10, pp. 1597–1607, Jul. 2005, ISSN: 01678655. DOI: 10.1016/J.PATREC.2005.01.010.

[24] S. Messelodi, C. M. Modena, and M. Zanin, "A computer vision system for the detection and classification of vehicles at urban road intersections," *Pattern Analysis and Applications*, vol. 8, no. 1-2, pp. 17–31, Sep. 2005, ISSN: 14337541. DOI: 10.1007/S10044-004-0239-9.

[25] G. R. Leone *et al.*, "An intelligent cooperative visual sensor network for urban mobility," *Sensors (Basel, Switzerland)*, vol. 17, 11 Nov. 2017, ISSN: 14248220. DOI: 10.3390/S17112588.

[26] W. Ouyang *et al.*, "Deepid-net: Object detection with deformable part based convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1320–1334, 7 Jul. 2017, ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2587642.

[27] W. Ouyang *et al.*, "Deepid-net: Deformable deep convolutional neural networks for object detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07, pp. 2403–2412, Dec. 2014.

[28] N. Doulamis, "Adaptable deep learning structures for object labeling/tracking under dynamic visual environments," *Multimedia Tools and Applications*, vol. 77, pp. 9651–9689, 8 Apr. 2018, ISSN: 15737721. DOI: 10.1007/S11042-017-5349-7.

[29] Y. Li, F. Cui, X. Xue, and J. C.-W. Chan, "Coarse-to-fine salient object detection based on deep convolutional neural networks," *Signal Processing: Image Communication*, vol. 64, pp. 21–32, May 2018, ISSN: 0923-5965. DOI: 10.1016/J.IMAGE.2018.01.012.

[30] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1278–1289, Mar. 2020.

[31] S. J. Lee, H. Choi, and S. S. Hwang, "Real-time depth estimation using recurrent cnn with sparse depth cues for slam system," *International Journal of Control, Automation and Systems*, vol. 18, pp. 206–216, 1 Jan. 2020, ISSN: 20054092. DOI: 10.1007/S12555-019-0350-8.

[32] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, Sep. 2014, ISSN: 10636919. DOI: 10.1109/CVPR.2014.220.

[33] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997, ISSN: 10459227. DOI: 10.1109/72.554195.

[34] H. Sekkat, S. Tigani, R. Saadane, and A. Chehri, "Vision-based robotic arm control algorithm using deep reinforcement learning for autonomous objects grasping," *Applied Sciences 2021, Vol. 11, Page 7917*, vol. 11, p. 7917, 17 Aug. 2021, ISSN: 2076-3417. DOI: 10.3390/APP11177917.

[35] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *Robotics: Science and Systems*, 2018, ISSN: 2330765X. DOI: 10.15607/RSS.2018.XIV.019.

[36] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12362 LNCS, pp. 574–591, 2020, ISSN: 16113349. DOI: 10.1007/978-3-030-58520-4_34.

[37] F. Michel, A. Krull, E. Brachmann, M. Y. Yang, S. Gumhold, and C. Rother, "Pose estimation of kinematic chain instances via object coordinate regression," British Machine Vision Association, Dec. 2015, pp. 181.1–181.11, ISBN: 1-901725-53-7. DOI: 10.5244/C.29.181.

[38] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11218 LNCS, pp. 765–781, 2018, ISSN: 16113349. DOI: 10.1007/978-3-030-01264-9_45.

[39] K. Fu, J. Fu, Z. Wang, and X. Sun, "Scattering-Keypoint-Guided Network for Oriented Ship Detection in High-Resolution and Large-Scale SAR Images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 11 162–11 178, 2021, ISSN: 21511535. DOI: 10.1109/JSTARS.2021.3109469.

[40] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, Jun. 2014, ISSN: 00313203. DOI: 10.1016/J.PATCOG.2014.01.005.

[41] T. Schmidt, R. Newcombe, and D. Fox, "DART: Dense Articulated Real-Time Tracking," *Robotics: Science and Systems*, 2014, ISSN: 2330765X. DOI: 10.15607/RSS.2014.X.030.

[42] M. B. Alatise and G. P. Hancke, "Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended kalman filter," *Sensors (Switzerland)*, vol. 17, 10 Oct. 2017, ISSN: 14248220. DOI: 10.3390/S17102164.

[43] M. M. Soltani, Z. Zhu, and A. Hammad, "Framework for location data fusion and pose estimation of excavators using stereo vision," *Journal of Computing in Civil Engineering*, vol. 32, 6 Nov. 2018, ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000783.

[44] M. H. Ali, K. Aizat, K. Yerkhan, T. Zhandos, and O. Anuar, "Vision-based robot manipulator for industrial applications," *Procedia Comput Sci*, vol. 133, no. 2, pp. 205–212, 2018, ISSN: 18770509. DOI: 10.1016/j.procs.2018.07.025.

[45] B. A. Griffin, V. Florence, and J. J. Corso, "Video object segmentation-based visual servo control and object depth estimation on a mobile robot," *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pp. 1636–1646, Mar. 2020. DOI: 10.1109/WACV45572.2020.9093335.

[46] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," *Robotics: Science and Systems*, 2018, ISSN: 2330765X. DOI: 10.15607/RSS.2018.XIV.021.

[47] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9912 LNCS, pp. 483–499, 2016, ISSN: 16113349. DOI: 10.1007/978-3-319-46484-8_29.

[48] J. Wang *et al.*, "Deep 3D human pose estimation: A review," *Computer Vision and Image Understanding*, vol. 210, Sep. 2021, ISSN: 1090235X. DOI: 10.1016/j.cviu.2021.103225.

[49] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Transactions on Graphics*, vol. 33, 5 Aug. 2014, ISSN: 15577368. DOI: 10.1145/2629500.

[50] M. Arashpour, T. Ngo, and H. Li, "Scene understanding in construction and buildings using image processing methods: A comprehensive review and a case study," *Journal of Building Engineering*, vol. 33, Jan. 2021, ISSN: 23527102. DOI: 10.1016/j.jobe.2020.101672.

[51] J. Wu, N. Cai, W. Chen, H. Wang, and G. Wang, "Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset," *Automation in Construction*, vol. 106, Oct. 2019, ISSN: 09265805. DOI: 10.1016/j.autcon.2019.102894.

[52] S. Tang, D. Roberts, and M. Golparvar-Fard, "Human-object interaction recognition for automatic construction site safety inspection," *Automation in Construction*, vol. 120, Dec. 2020, ISSN: 09265805. DOI: 10.1016/j.autcon.2020.103356.

[53] H. Son, H. Seong, H. Choi, and C. Kim, "Real-Time Vision-Based Warning System for Prevention of Collisions between Workers and Heavy Equipment," *Journal of Computing in Civil Engineering*, vol. 33, no. 5, Sep. 2019, ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000845.

[54] B. Sherafat *et al.*, "Automated Methods for Activity Recognition of Construction Workers and Equipment: State-of-the-Art Review," *Journal of Construction Engineering and Management*, vol. 146, no. 6, Jun. 2020, ISSN: 0733-9364. DOI: 10.1061/(ASCE)CO.1943-7862.0001843.

[55] H. Luo, M. Wang, P. K. Y. Wong, J. Tang, and J. C. Cheng, "Vision-based pose forecasting of construction equipment for monitoring construction site safety," *Lecture Notes in Civil Engineering*, vol. 98, pp. 1127–1138, 2021, ISSN: 23662565. DOI: 10.1007/978-3-030-51295-8_78.

[56] M. Kassem, E. Mahamedi, K. Rogage, K. Duffy, and J. Huntingdon, "Measuring and benchmarking the productivity of excavators in infrastructure projects: A deep neural network approach," *Automation in Construction*, vol. 124, Apr. 2021, ISSN: 09265805. DOI: 10.1016/j.autcon.2020.103532.

[57] J. Kim and S. Chi, "Multi-camera vision-based productivity monitoring of earthmoving operations," *Automation in Construction*, vol. 112, Apr. 2020, ISSN: 09265805. DOI: 10.1016/j.autcon.2020.103121.

[58] R. Khallaf and M. Khallaf, "Classification and analysis of deep learning applications in construction: A systematic literature review," *Automation in Construction*, vol. 129, p. 103 760, Sep. 2021, ISSN: 09265805. DOI: 10.1016/j.autcon.2021.103760.

[59] J. Park, J. Chen, and Y. K. Cho, "Self-corrective knowledge-based hybrid tracking system using bim and multimodal sensors," *Advanced Engineering Informatics*, vol. 32, pp. 126–138, 2017.

[60] K. M. Lundeen, S. Dong, N. Fredricks, M. Akula, J. Seo, and V. R. Kamat, "Optical marker-based end effector pose estimation for articulated excavators," *Automation in Construction*, vol. 65, pp. 51–64, May 2016, ISSN: 09265805. DOI: 10.1016/j.autcon.2016.02.003.

[61] C. Feng, V. R. Kamat, and H. Cai, "Camera marker networks for articulated machine pose estimation," *Automation in Construction*, vol. 96, pp. 148–160, Dec. 2018, ISSN: 09265805. DOI: 10.1016/j.autcon.2018.09.004.

[62] J. Zhao, Y. Hu, and M. Tian, "Pose estimation of excavator manipulator based on monocular vision marker system," *Sensors*, vol. 21, 13 Jul. 2021, ISSN: 14248220. DOI: 10.3390/S21134478.

[63] M. M. Soltani, Z. Zhu, and A. Hammad, "Skeleton estimation of excavator by detecting its parts," *Automation in Construction*, vol. 82, pp. 1–15, Oct. 2017, ISSN: 09265805. DOI: 10.1016/j.autcon.2017.06.023.

[64] C. Yuan, S. Li, and H. Cai, "Vision-based excavator detection and tracking using hybrid kinematic shapes and key nodes," *Journal of Computing in Civil Engineering*, vol. 31, 1 Jan. 2017, ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000602.

[65] H. Kim, H. Kim, Y. W. Hong, and H. Byun, "Detecting construction equipment using a region-based fully convolutional network and transfer learning," *Journal of Computing in Civil Engineering*, vol. 32, 2 Mar. 2018, ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000731.

[66] J. Xu and H. S. Yoon, "Vision-based estimation of excavator manipulator pose for automated grading control," *Automation in Construction*, vol. 98, pp. 122–131, Feb. 2019, ISSN: 09265805. DOI: 10.1016/j.autcon.2018.11.022.

[67] M. Arashpour, V. Kamat, A. Heidarpour, M. R. Hosseini, and P. Gill, "Computer vision for anatomical analysis of equipment in civil infrastructure projects: Theorizing the development of regression-based deep neural networks," *Automation in Construction*, vol. 137, p. 104 193, May 2022, ISSN: 09265805. DOI: 10.1016/j.autcon.2022.104193.

[68] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end active object tracking and its real-world deployment via reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 1317–1332, 6 Jun. 2020, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2019.2899570.

[69] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11210 LNCS, pp. 472–487, 2018, ISSN: 16113349. DOI: 10.1007/978-3-030-01231-1_29.

[70] S. Zhang and L. Zhang, "Construction site safety monitoring and excavator activity analysis system," *Construction Robotics*, vol. 3, pp. 49–56, Jul. 2022, ISSN: 2509-811X. DOI: 10.1007/s41693-022-00077-0.

[71] C.-J. Liang, K. M. Lundeen, W. McGee, C. C. Menassa, S. Lee, and V. R. Kamat, "A vision-based marker-less pose estimation system for articulated construction robots," *Automation in Construction*, vol. 104, pp. 80–94, Aug. 2019, ISSN: 09265805. DOI: 10.1016/j.autcon.2019.04.004.

[72] A. Assadzadeh, M. Arashpour, I. Brilakis, T. Ngo, and E. Konstantinou, "Vision-based excavator pose estimation using synthetically generated datasets with domain randomization," *Automation in Construction*, vol. 134, Feb. 2022, ISSN: 09265805. DOI: 10.1016/j.autcon.2021.104089.

[73] J. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3349–3364, 2020.

[74] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9908 LNCS, pp. 662–677, 2016, ISSN: 16113349. DOI: 10.1007/978-3-319-46493-0_40.

[75] A. K. Langroodi, F. Vahdatikhaki, and A. Doree, "Activity recognition of construction equipment using fractional random forest," *Automation in Construction*, vol. 122, Feb. 2021, ISSN: 09265805. DOI: 10.1016/j.autcon.2020.103465.

[76] Y. Guo, H. Cui, and S. Li, "Excavator joint node-based pose estimation using lightweight fully convolutional network," *Automation in Construction*, vol. 141, p. 104 435, Sep. 2022, ISSN: 0926-5805. DOI: 10.1016/J.AUTCON.2022.104435.

[77] *Hardware specifications - sdk-wiki*, Accessed: 2023-01-16. [Online]. Available: https://sdk.rethinkrobotics.com/wiki/Hardware_Specifications.

[78] R. W. II, *Baxter humanoid robot kinematics*, Accessed: 2023-01-15, Apr. 2017. [Online]. Available: https://www.ohio.edu/mechanical-faculty/williams/html/pdf/BaxterKinematics.pdf.

[79] M. Menolotto, D.-S. Komaris, S. Tedesco, B. O'Flynn, and M. Walsh, "Motion capture technology in industrial applications: A systematic review," *Sensors*, vol. 20, no. 19, p. 5687, 2020.

[80] *Vero — compact super wide camera by vicon*, Accessed: 2023-01-23. [Online]. Available: https://www.vicon.com/hardware/cameras/vero/.

[81] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A moveit! case study," Apr. 2014.

[82] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 7553 May 2015, ISSN: 14764687. DOI: 10.1038/NATURE14539.

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[84] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[85] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[86] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014. DOI: 10.48550/arxiv.1409.1556.

[87] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[88] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[89] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, Dec. 2015, ISSN: 10636919. DOI: 10.48550/arxiv.1512.03385.

[90] C. Gu, J. J. Lim, P. Arbeláez, and J. Malik, "Recognition using regions," *2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009*, pp. 1030–1037, 2009. DOI: 10.1109/CVPR.2009.5206727.

[91] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," IEEE, Jun. 2014, pp. 580–587, ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.81.

[92] J. R. Uijlings, K. E. V. D. Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 2 Sep. 2013, ISSN: 09205691. DOI: 10.1007/S11263-013-0620-5/FIGURES/9.

[93] R. Girshick, "Fast r-cnn," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, Apr. 2015.

[94] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 6 Jun. 2015.

[95] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 386–397, 2 Mar. 2017, ISSN: 19393539. DOI: 10.48550/arxiv.1703.06870.

[96] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," Dec. 2016. DOI: 10.48550/arxiv.1612.03144.

[97] T. Y. Lin *et al.*, "Microsoft coco: Common objects in context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, pp. 740–755, PART 5 2014, ISSN: 16113349. DOI: 10.1007/978-3-319-10602-1_48.

[98] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," Apr. 2017. DOI: 10.48550/arxiv.1704.04861.

[99] Y. Ma, S. Soatto, J. Košecká, and S. S. Sastry, *An Invitation to 3-D Vision*. Springer New York, 2004, vol. 26, ISBN: 978-1-4419-1846-8. DOI: 10.1007/978-0-387-21779-6.

[100] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[101] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 22, pp. 215–221, 2 Jun. 1955, ISSN: 0021-8936. DOI: 10.1115/1.4011045.

[102] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[103] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[104] H. P. Gavin, "The levenberg-marquardt algorithm for nonlinear least squares curve-fitting problems," *Department of Civil and Environmental Engineering, Duke University*, vol. 19, 2019.

[105] A. Assadzadeh, M. Arashpour, A. Bab-Hadiashar, T. Ngo, and H. Li, "Automatic far-field camera calibration for construction scene analysis," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, pp. 1073–1090, 8 Aug. 2021, ISSN: 14678667. DOI: 10.1111/MICE.12660.

Appendix A: URDF for the scale excavator

Listing A.1: excavator_assembly.urdf

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This URDF was automatically created by SolidWorks to URDF Exporter!
     Commit Version: 1.6.0-4-g7f85cfe  Build Version: 1.6.7995.38578
     For more information, please see http://wiki.ros.org/sw_urdf_exporter -->
<robot
  name="Excavator_Assembly">
  <link
    name="base_link">
    <inertial>
      <origin
        xyz="0.54903 6.6613E-16 -0.26501"
        rpy="0 0 0" />
      <mass
        value="1994.8" />
      <inertia
        ixx="420.67"
        ixy="7.0217E-14"
        ixz="0.54095"
        iyy="1085.5"
        iyz="-4.1458E-14"
        izz="1412" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://my_arm_xacro/meshes/base_link.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0.79216 0.81961 0.93333 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://my_arm_xacro/meshes/base_link.STL" />
      </geometry>
    </collision>
  </link>
  <link
    name="body_link">
```

```
  <inertial>
    <origin
      xyz="0.70971 0.011983 0.4433"
      rpy="0 0 0" />
    <mass
      value="2158.1" />
    <inertia
      ixx="609.47"
      ixy="82.787"
      ixz="-86.856"
      iyy="1169.7"
      iyz="-5.0784"
      izz="1507.7" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://my_arm_xacro/meshes/body_link.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.79216 0.81961 0.93333 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://my_arm_xacro/meshes/body_link.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="joint_1"
  type="revolute">
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <parent
    link="base_link" />
  <child
    link="body_link" />
  <axis
    xyz="0 0 -1" />
  <limit
    lower="-0.7"
    upper="0.7"
    effort="2000"
    velocity="0.4" />
</joint>
<link
  name="arm_link">
  <inertial>
    <origin
      xyz="-1.2299 0.020523 0.90616"
      rpy="0 0 0" />
    <mass
      value="258.64" />
    <inertia
      ixx="31.757"
      ixy="0.062089"
      ixz="51.714"
```

```
      iyy="153.57"
      iyz="-0.042295"
      izz="125.08" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://my_arm_xacro/meshes/arm_link.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.79216 0.81961 0.93333 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://my_arm_xacro/meshes/arm_link.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="joint_2"
  type="revolute">
  <origin
    xyz="-0.1942 -0.020757 0.5673"
    rpy="0 0.076088 0" />
  <parent
    link="body_link" />
  <child
    link="arm_link" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-0.17"
    upper="0.4"
    effort="2000"
    velocity="0.4" />
</joint>
<link
  name="wrist_link">
  <inertial>
    <origin
      xyz="0.11376 -0.020757 -0.48183"
      rpy="0 0 0" />
    <mass
      value="136.5" />
    <inertia
      ixx="40.101"
      ixy="-6.0778E-08"
      ixz="-4.6951"
      iyy="41.634"
      iyz="-4.1983E-07"
      izz="2.5564" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
```

```
          filename="package://my_arm_xacro/meshes/wrist_link.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0.95686 0.68235 0.043137 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://my_arm_xacro/meshes/wrist_link.STL" />
      </geometry>
    </collision>
  </link>
  <joint
    name="joint_3"
    type="revolute">
    <origin
      xyz="-2.7107 0 1.3011"
      rpy="0 -0.98702 3.1416" />
    <parent
      link="arm_link" />
    <child
      link="wrist_link" />
    <axis
      xyz="0 1 0" />
    <limit
      lower="0.24"
      upper="1.24"
      effort="2000"
      velocity="0.4" />
  </joint>
  <link
    name="bucket_link">
    <inertial>
      <origin
        xyz="0.37286 0.15186 0.02067"
        rpy="0 0 0" />
      <mass
        value="70.075" />
      <inertia
        ixx="7.3387"
        ixy="1.013"
        ixz="0.001331"
        iyy="9.1696"
        iyz="0.0014523"
        izz="7.4771" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://my_arm_xacro/meshes/bucket_link.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0.79216 0.81961 0.93333 1" />
      </material>
    </visual>
    <collision>
      <origin
```

```
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://my_arm_xacro/meshes/bucket_link.STL" />
      </geometry>
    </collision>
  </link>
  <joint
    name="joint_4"
    type="revolute">
    <origin
      xyz="-0.027179 0 -1.7083"
      rpy="-1.5708 -0.060798 3.1416" />
    <parent
      link="wrist_link" />
    <child
      link="bucket_link" />
    <axis
      xyz="0 0 1" />
    <limit
      lower="0.30"
      upper="2.47"
      effort="2000"
      velocity="0.4" />
  </joint>
</robot>
```