Investigating Deep-Learning-based Solutions for Flexible and Robust Hand-Eye Calibration in Robotics

Ozan Bahadir

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Engineering
College of Science and Engineering
University of Glasgow

April 2023

Abstract

The cameras are the main sensor for robots to perceive their environments because they provide high-quality information and their low-cost. However, transforming the information obtained from cameras into robotic actions can be challenging. To manipulate objects in camera scenes, robots need to establish a transformation between the camera and the robot base, which is known as hand-eye calibration. Achieving accurate hand-eye calibration is critical for precise robotic manipulation, yet traditional approaches can be time-consuming, error-prone, and fail to account for changes in the camera or robot base over time.

This thesis proposes a novel approach that leverages the power of deep learning to automatically learn the mapping between the robot's joint angles and the camera's images, enabling real-time calibration updates. The approach samples the robot and camera spaces discretely and represents them continuously, enabling efficient and accurate computation of calibration parameters. By automating the calibration process and using deep learning algorithms, a more robust and efficient solution for hand-eye calibration in robotics is offered.

To develop a robust and flexible hand-eye calibration approach, three main studies were conducted. In the first study, a deep learning-based regression architecture was developed that processes RGB and depth images, as well as the poses of a single reference point selected on the robot end-effector with respect to the robot base acquired through the robot kinematic chain. The success of this architecture was tested in a simulated environment and two real robotic environments, evaluating the metric error and precision. In the second approach, the success of the developed approach was evaluated by transferring from metric error to task error by performing a real robotic manipulation task, specifically a pick-and-place. Additionally, the performance of the developed approach was compared with a classic hand-eye calibration approach, using three evaluation criteria: real robotic manipulation task, computational complexity, and repeatability. Finally, the learned calibration space of the developed deep learning-based hand-eye calibration approach was extended with new observations over time using Continual learning, making the approach more robust and flexible in handling environmental changes.Two buffer-based approaches were developed to eliminate the catastrophic forgetting problem, which is forgetting learned information over time by considering new observations. The performance and comparison of these approaches with the training of the developed approach in the first study using all datasets from scratch were tested on a simulated and a real-world environment.

Experimental results of this thesis reveal that: 1) a deep learning-based hand-eye calibration approach has competitive results with the classical approaches in terms of metric error (positional and rotational error deviation from the ground-truth) while eliminating data re-collection and re-training camera pose changes over time, and has 96 times better repeatability (precision) than the classic approach as well as it has the state-of-the-art result for it in comparison to the other deep learning-based hand-eye calibration approaches; 2) it also has competitive results with the classic approaches for performing a real-robotic manipulation task and reduces the computational complexity; 3) the leveraging deep-learning based hand-eye calibration approach with Continual Learning, it is possible to extend the learned calibration space over new observations without training the network from scratch with a lower accuracy gap (less than 1.5 mm and 2.5 degrees in the simulations and real-world environments for the translation and orientation components).

Overall, the proposed approach offers a more efficient and robust solution for hand-eye calibration in robotics, providing greater accuracy and flexibility to adapt to environments where the poses of the robot and camera base change according to each other over time. These changes may come from either robot or camera movement.The results of the studies demonstrate the effectiveness of the approach in achieving precise and reliable robotic manipulation, making it a promising solution for robotics applications.

Contents

List of Tables

List of Figures

Nomenclature

$\mathbf{A}$	a matrix
$\mathbf{A^{-1}}$	inverse of **A**
$\mathbf{A^T}$	transpose of **A**
$\mathbb{R}$	set of real numbers
$\mathbb{R}^3$	set of real numbers in 3 dimensional space
υ	a vector
$\hat{\upsilon}$	the unit vector of υ
$\|\upsilon\|$	norm of vector υ
$\upsilon_1 \cdot \upsilon_2$	dot product
$\upsilon_1 \times \upsilon_2$	vector product
$\mathbf{C}$	camera matrix
$\mathbf{K}$	camera calibration matrix
$\mathbf{R}$	an orthonormal rotation matrix
$\mathbf{T}$	homogeneous transformation matrix
$\mathbf{P}$	world point
$\mathbf{{}^{A}T_{B}}$	homogeneous transformation frame **B** with respect to frame **A**
$\mathbf{f}$	focal length
θ	angle , **rad**
$\mathbf{q}$	quaternion
$\mathbf{q}^\circ$	unit quaternion

$\vert\mathbf{q}\vert$	scalar norm of quaternion
$\mathbf{q}^*$	the complex conjugate of **q**
$\mathbf{q}^\circ(\upsilon)$	pure quaternion of vector v

Acknowledgements

I would like to express my sincere appreciation to Dr Gerardo Aragon-Camarasa and Dr Jan Paul Sibert for their exceptional guidance and mentorship throughout the past four years. Your dedicated supervision has been instrumental in shaping my academic journey and contributing to my growth as a researcher. Your insights, expertise, and unwavering support have been invaluable, and I am truly grateful for the opportunity to learn under your guidance. Thank you for your commitment and encouragement, which have been instrumental in my academic and personal development.

I want to extend my gratitude to Dr Ali Gooya for his organisation of my final Viva meeting. Additionally, I sincerely appreciate the efforts put forth by Dr Emma Li and Dr Randika Kosala Wathavana Vithanage in meticulously examining my PhD thesis. Their valuable feedback and insightful suggestions have significantly contributed to enhancing the quality of my work.

I would like to extend my sincere gratitude to Dr. José Cano Reyes and Prof. Phil Trinder for their invaluable feedback provided during my annual progress reviews. Their constructive insights and suggestions have significantly contributed to enhancing the robustness and quality of my research.

I am deeply grateful for the invaluable contributions and support extended by the Computer Vision and Autonomous Systems Group (CVAS) at the School of Computing Science, University of Glasgow. The enriching experiences and opportunities I have had within this esteemed community have significantly enhanced my research endeavours and my capabilities in presenting and academic writing. I extend my heartfelt appreciation for their unwavering encouragement and the platform they provided, which has been instrumental in fostering my academic and professional development.

I extend my heartfelt gratitude to my dear friends Abdulkadir Ciris, Kutlu Balci, and Hande Balci, who have been steadfast companions since the inception of my PhD journey, forming a bond akin to a family here in Scotland. Additionally, I wish to express my sincere appreciation to Ahmet Burak Ozyurt, Meltem Haktaniyan, Ceren Erdem, Ozgu Goksu, and Elifcan Beyazit, individuals with whom my paths crossed during my PhD endeavour. Your unswerving support and camaraderie have been a constant source of solace and motivation. Amid the rigours of this academic pursuit, your friendship has served as a reassuring presence, alleviating the challenges of the academic journey. Your willingness to lend a listening ear, share invaluable insights, and offer your aid have significantly enriched my experience. I am truly indebted for your presence and how you have made this journey more navigable through your companionship. The shared moments we have cherished together have been an unequivocal blessing, and I am profoundly grateful for the enrichment you have brought to my academic voyage.

I sincerely thank the Association of Turkish Alumni and Students in Scotland (ATAS), a charitable organisation, for its significant impact on my academic journey. Their well-organised social events not only provided me with the opportunity to meet a diverse and valuable network of individuals but also served as a means of alleviating the stress associated with my PhD pursuit. Additionally, their academic events facilitated connections with colleagues and researchers within the broader academic community of Scotland. Above all, I am grateful to ATAS for creating an environment that genuinely allows me to feel Scotland as my home.

Last but certainly not least, I wish to extend my heartfelt gratitude to my family, whose unwavering and unconditional support has been a constant source of strength throughout my entire life.

I also thank the Turkish Ministry of Education, which funded this project.

Declaration

Chapter 1

Introduction

In this thesis, novel hand-eye calibration methodologies have been developed using deep learning, which allow for the estimation of calibration parameters in dynamic robotic environments where the camera and robot base can change after data collection. This is in contrast to static environments, where the pair of the camera and robot base are fixed and exact in both the training and testing stages. The approach extends the learned calibration space through Continual Learning, enhancing its adaptability to new situations. The effectiveness of the proposed approach is demonstrated through a real-world robotic manipulation task involving pick-and-place operations. Unlike classical hand-eye calibration approaches that rely on metric error, this approach provides a more robust and accurate calibration process, resulting in improved manipulation performance.

1.1 A Brief Introduction to the Hand-eye Calibration

Robotic systems rely heavily on cameras to capture visual data that can be used for various tasks, such as object recognition, localisation, and manipulation. However, the camera must be accurately aligned with the robot's kinematic chain to make the most of this data. This process, known as hand-eye calibration (HEC), is a critical step in many robotic applications, such as pick-and-place operations [1], assembly lines [2], and quality control [3]. Despite significant progress in the static hand-eye configuration, where the camera and the robot base remain fixed after data collection and calibration estimation, achieving accurate and efficient dynamic HEC configurations remains a challenging problem. In this configuration, the camera pose changes with respect to the robot base after data collection and calibration parameters' estimation over time. Dynamic hand-eye calibration is essential in various robotic applications where the robot and the camera continuously move or operate in a changing environment.

Figure 1.1 shows a visualisation of the three spaces involved in robotic manipulation: the robot, camera, and object space. The camera captures an image of the scene and uses its extrinsic parameters to identify the object and determine its pose with respect to the camera base, a

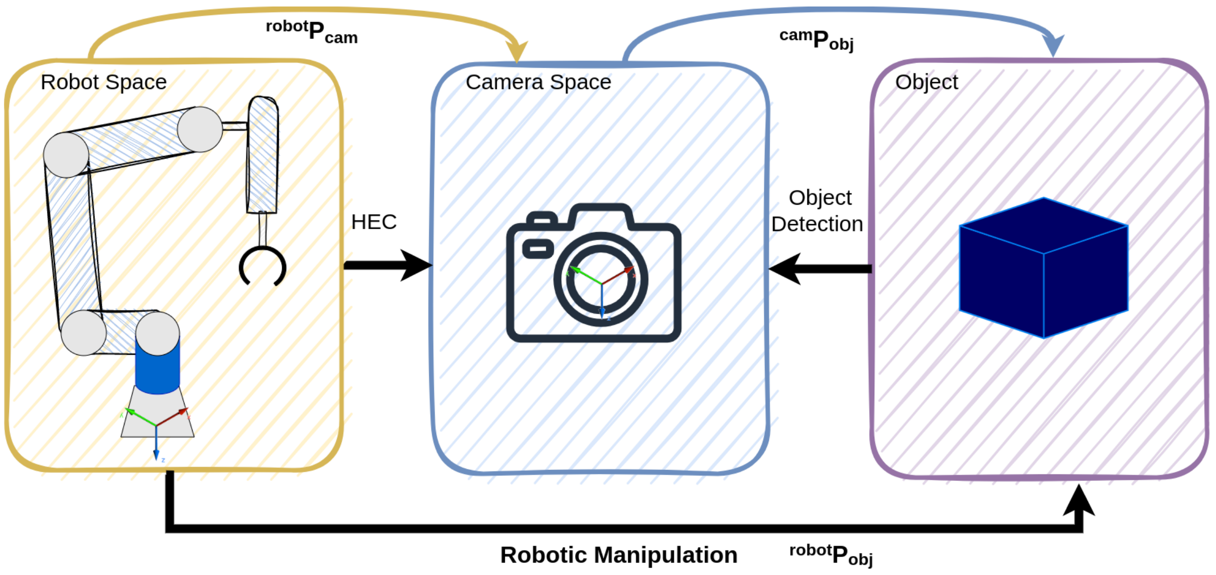


Figure 1.1: This figure depicts the relationship between the robot space, camera space, and object space.

process known as object detection. However, the information captured in the camera space must be transferred to the robot space to manipulate the object. This is achieved through hand-eye calibration, which establishes the transformation link between the two spaces.

Figure 1.1 depicts the transformation from robot space to object space using camera space, assuming that the robot space is fixed and serves as the world space. The object's pose is obtained via object detection, and its pose relative to the camera base frame is given by P_{obj}^{cam}. To manipulate the object, its pose must be defined with respect to the robot base frame.

Two cases exist for this transformation. In the first case, the camera or cameras can be embedded the robot joints, such as the head or the end-effectors. The poses of these cameras are linked to the robot kinematic chain and can be used directly to manipulate the object. In the second case, which is common in robotic systems, the camera may be placed externally. In the second case, hand-eye calibration is needed to determine the transformation between the camera and robot base frames for manipulating the object.

For the second case, there are two main scenarios: static and dynamic hand-eye calibration (depicted in Figure 1.2). In the static hand-eye calibration scenario, the camera and the robot base remain fixed for robotic manipulation after data collection and camera pose estimation with respect to the robot base. As for the dynamic scenario, the camera pose changes with respect to the robot base over time. In this scenario, the calibration approach must be updated the calibration parameters over time.

1.1.1 The importance

Hand-eye calibration is critical to robotic manipulation, allowing robots to accurately perceive their environment and perform precise manipulation tasks. Robots may need proper calibration

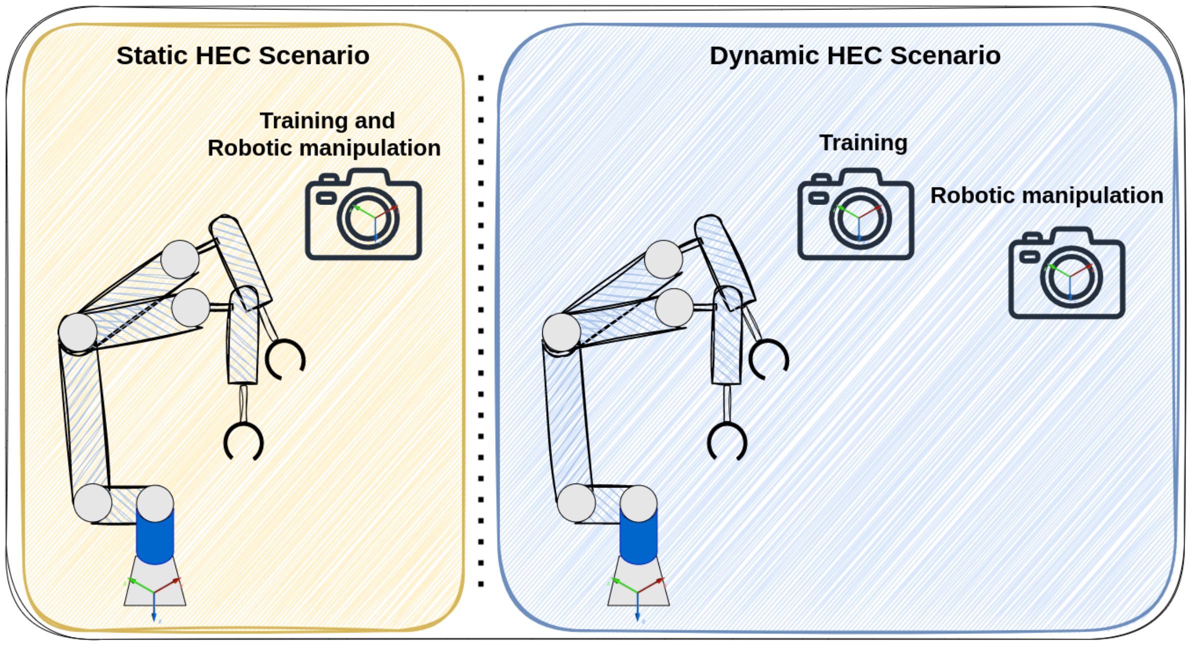


Figure 1.2: This figure depicts the static and dynamic hand-eye calibration scenarios. In the static scenario, the camera and the robot base remain fixed, and the calibration parameters estimated during training can be used for all robotic manipulations. In contrast, in the dynamic scenario, the camera pose changes with respect to the robot base during the robotic manipulation, requiring recalibration to maintain accurate alignment.

to locate and manipulate objects, leading to inefficiencies and inaccuracies.

This thesis focuses on two hand-eye calibration scenarios. The first is the static hand-eye calibration configuration, where the camera's pose and the robot's base frames remain fixed after data collection and calibration. This scenario is common in industries where robots perform repetitive tasks, such as pick-and-place operations. In such cases, classical hand-eye calibration approaches have successfully provided accurate calibration by collecting data and solving the hand-eye calibration equation offline.

The second scenario (dynamic robotic environments) is more complex and involves changes to the camera and robot base frames over time. In dynamic robotic environments, such as those operating outside enclosed spaces, changes to the camera and robot base frames over time can lead to degraded performance and safety risks. As an illustration, the robots can execute carrying operations within diverse camera fields of view positioned in external locales. To guarantee safety and stability in these operations, the hand-eye calibration parameters must be continuously updated online. A flexible and autonomous hand-eye calibration approach is essential to ensure that robots can adapt to these changes without requiring manual recalibration. One promising approach is to use deep learning and continual learning techniques to develop a system that can adapt to changing environments and tasks. With deep learning algorithms, the system can learn to map the visual features of the camera to the robot's pose, enabling the robot to adapt to changes in the camera and robot base frames. Continual learning allows the system to improve the accuracy of hand-eye calibration over time by updating the calibration parameters based on new data. Robots can operate in dynamic environments with improved accuracy, performance, and safety using this flexible and autonomous hand-eye calibration approach.

1.2 Current Literature and Challenges

1.2.1 Current Literature

There are several classic approaches to solving the HEC problem that are widely used in the robotics community. These approaches typically involve moving the robot to multiple different positions while recording the camera and end-effector poses with respect to the robot or world frame. The HEC problem can then be broken down into three main steps (depicted in Figure 1.3): robot calibration, camera calibration, and hand-eye calibration.

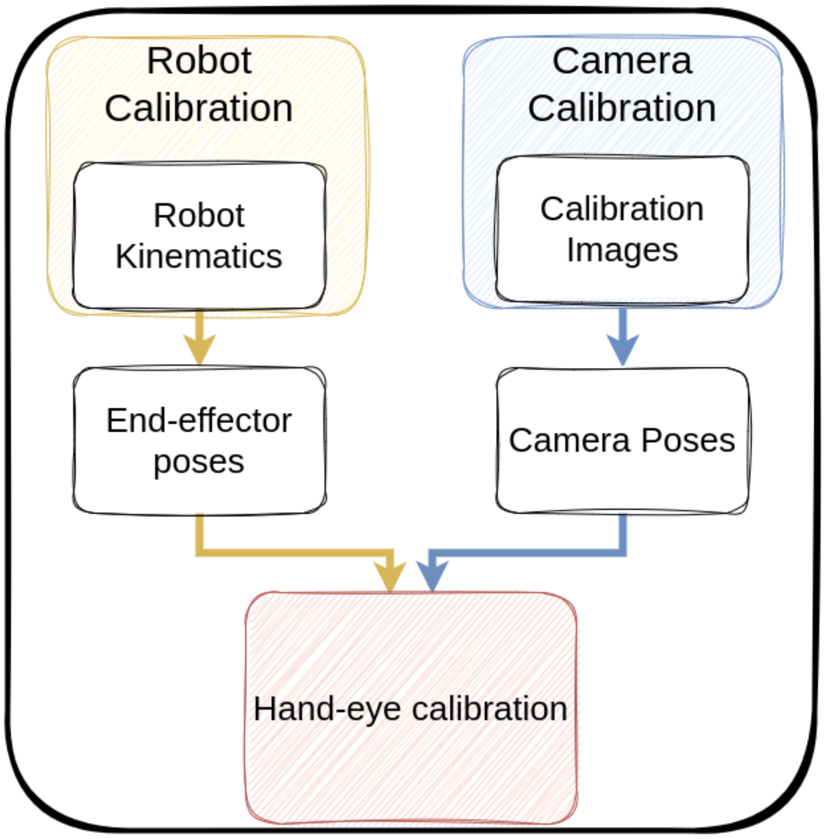


Figure 1.3: The figure outlines the key steps of classic hand-eye calibration: robot calibration for precise end-effector poses, camera calibration for accurate camera poses, and a hand-eye calibration model to estimate parameters from these poses.

The first step, robot calibration, involves determining the end-effector poses with respect to the robot or world base frame. This calibration is assumed to be known via the predefined robot kinematic chain, but it may become inaccurate over time due to wear and tear on the robot. Continuous usage and environmental conditions can gradually shift the robot's joints, changing their relative positions. These changes accumulate and eventually result in calibration discrepancies, affecting the precision and accuracy of the robot's movements and interactions. The magnitudes of positional errors typically range from a few millimetres to centimetres, while orientation errors can span from fractions of a degree to several degrees. These error ranges hinge on the application's intricacies, the robot system's complexity, and the exacting of the precision requirements. This issue is not typically addressed in classic approaches.

The second step, camera calibration, is used to obtain the camera's intrinsic and extrinsic parameters (the focal length, principal point coordinates, and lens distortion coefficients). Calibration targets [4, 5] or QR-code-based markers [6, 7] are used to capture images from different viewpoints, and these images are then used to calculate the camera's intrinsic parameters. The extrinsic parameters can be obtained using the known poses of the calibration targets or markers in the world frame. These extrinsic parameters can then be used to determine the camera's pose for different robot end-effector configurations.

The final step, hand-eye calibration, involves formulating linear or nonlinear equations based on the camera and robot calibration data and solving for the HEC parameters using mathematical optimisation techniques. Classic approaches include methods such as **AX=XB** [8–12] and **AX=YB** [13–17], which involve solving a linear system of equations, and the projection error and 3D reconstruction methods, which use nonlinear optimisation techniques. In **AX=XB** formulation, **X** is the unknown transformation between the camera base and the robot's base or end-effector, while **A** and **B** are observed linear transformations of end-effector poses with respect to the robot base and camera poses relative to the calibration target across n successive configurations. In **AX=YB** formulation, **Y** is the unknown transformation from the robot base to the world coordinate base, and **X** is the camera base to the robot base. **A** and **B** stand for the observable pose of the robot's end-effector and the camera's pose, respectively.

The approaches mentioned above have demonstrated satisfactory outcomes; however, they are offline, implying that their outcomes are only valid for camera positions used during calibration. When the camera position changes, all the data must be collected again for the new configuration, thus making them unsuitable for dynamic real-world robotic applications. To tackle this limitation, researchers have proposed deep learning-based approaches [18] [19] that eliminate the data collection process and allow the model to be re-trained when the camera pose changes, providing a more flexible hand-eye calibration model. Current hand-eye calibration methods have achieved significant success. However, a robust and flexible method is still required to recalibrate the system without data recollection and can be easily deployed into other robotic environments.

1.2.2 Challenges

The limitations of classic hand-eye calibration approaches are as follows:

- Firstly, they are offline methods, meaning that the hand-eye calibration parameters they generate are only applicable to the specific camera and robot configurations used during data collection. When the pose of the camera with respect to the robot base changes, these methods cannot automatically adjust to the new configuration. Consequently, they are not well-suited to dynamic robotic systems, where frequent recalibration may be required.

- Secondly, classic hand-eye calibration approaches require expertise to collect data and generate hand-eye calibration parameters. Moreover, the resulting calibration parameters may not always be accurate, which can necessitate multiple rounds of data collection and parameter tuning to obtain a satisfactory result.

- Finally, classic hand-eye calibration approaches lack flexibility in adapting to new environments, as the calibration space they learn is typically limited to a particular setup for external hand-eye configuration where an external camera observes the robot from a distance. As a result, they may not be suitable for applications where the robot must operate in diverse and changing environments.

- Current deep learning-based approaches applied to external hand-eye configuration predominantly leverage deep learning techniques for feature extraction rather than directly estimating the calibration parameters.

1.3 Aims and Objectives

1.3.1 Aim of the study

This thesis aims to address the limitations (detailed in section 1.2.2) of classical hand-eye calibration (HEC) approaches by developing a flexible and autonomous approach that leverages deep learning techniques. This thesis encompasses three main studies.

The first study (chapter 3) presents a deep learning-based hand-eye calibration approach that uses RGB and depth images and a reference point on the robot's end-effector to process data over various camera and robot configurations. Unlike classical approaches, this deep learning-based approach autonomously recalibrates itself to adapt to new camera poses within the 3D manifold data samples captured without requiring data re-collection or re-training. This feature makes it suitable for dynamic robotic environments where camera or robot poses may change over time for manipulation. Furthermore, this study addresses the first limitation of the current HEC approaches, which is their static nature.

The second study (chapter 4) in this thesis focuses on evaluating the performance of the deep learning-based hand-eye calibration approach developed in Chapter 3 in a real-world robotic manipulation task, specifically pick-and-place. Additionally, a comparison between the developed approach and a classic HEC approach was conducted regarding computational complexity, repeatability, and success of performing robotic manipulation. Computational complexity and repeatability are essential in assessing the suitability of the developed HEC approach for real-world applications. The performance of the manipulation shows the accuracy and precision of the HEC on a real task, in contrast to metric error. In addition, this study addresses two additional limitations of current HEC approaches: their computational complexity and the deviation of their success repeatability.

The third study (chapter 5) in this thesis focuses on developing a continuous deep learning-based hand-eye calibration approach that extends the learned calibration spaces through Continual Learning. This approach allows the deep learning-based HEC approach developed in Chapter 3 to adapt to new environments without requiring all the data to be stored and retrained from the beginning. As a result, the final HEC approach is flexible and can autonomously recalibrate itself even outside of the pre-learned calibration space. This makes the developed HEC approach suitable for dynamic robotic environments where recalibration is frequently required. Moreover, this HEC approach addresses the lack of flexibility limitation of the current HEC approaches.

1.3.2 The hypotheses and research questions

The main hypothesis of this thesis is:

A hand-eye calibration approach, which recalibrates external camera pose by observing a single reference point on the robot end-effector through a 3D vision system, enables a robot to perform robotic manipulation and grasping tasks by being adaptable and robust to environmental changes. Moreover, it has the same success on a real robotic manipulation task while reducing the computational complexity and increasing the repeatability compared to classical approaches.

On the other hand, each chapter has its hypothesis, which is detailed below.

Hypothesis in Chapter 3:

It is possible to carry out hand-eye calibration by tracking the known transformation of a single reference point on the robot's end-effector with respect to both the robot base and the camera via a 3D vision system. This HEC approach enables to get hand-eye calibration parameters without data re-collection within the learned calibration space while the repeatability score is better than the state-of-the-art deep learning-based approach, which is 10 millimeters.

To examine this hypothesis, the following research questions have been posited.

Q1 In contrast to closed-form hand-eye approaches, is it possible to find the geometric transformation between the camera and the robot where camera and robot calibrations are known in advance by using a neural network?

Q2 Is it possible to find the camera's pose with respect to the reference point by observing the motions of this reference point via a 3D vision system and using deep learning as a calibration model?

Q3 Is it possible to carry out hand-eye calibration where camera calibration is not known by observing the motions of the reference point via a 3D vision system and employing a deep learning-based regression architecture as a model?

Hypothesis in Chapter 4:

The deep learning-based HEC approach can achieve accurate calibration with reduced computational complexity and improved repeatability compared to classic methods. Specifically, this approach performs similarly to the classic hand-eye calibration approach on a real robotic manipulation task, such as a pick-and-place task, while reducing the number of attempts needed to obtain hand-eye calibration parameters and decreasing the data collection time. Furthermore, it achieves lower repeatability errors compared to the classic approach (Tsai's HEC approach [8]), which shows the precision of the HEC approach.

To examine this hypothesis, the following research questions have been posited.

Q4 How does the performance of the deep learning-based HEC approach compare to other state-of-the-art methods in the field of hand-eye calibration, in terms of both accuracy and computational efficiency?

Q5 What are the limitations of the deep learning-based HEC approach, and how can they be addressed to improve its practical application in various dynamic robotic environments?

Hypothesis in Chapter 5:

A Continual Learning-based hand-eye calibration system can extend the learned calibration space through new observations over time.

To examine this hypothesis, the following research questions have been posited.

Q6 Can hand-eye calibration be handled as a time sequence problem in terms of camera pose changes through Continual Learning?

Q7 Is it possible to extend the learned hand-eye calibration space via Continual Learning?

1.4 A Brief Overview of the Proposed Approaches

The thesis aims to develop a flexible and autonomous hand-eye calibration approach using deep learning, which can recalibrate itself without data recollection. To achieve this goal, the thesis conducted three main studies (presented in Figure 1.4) in simulated and real-world environments.

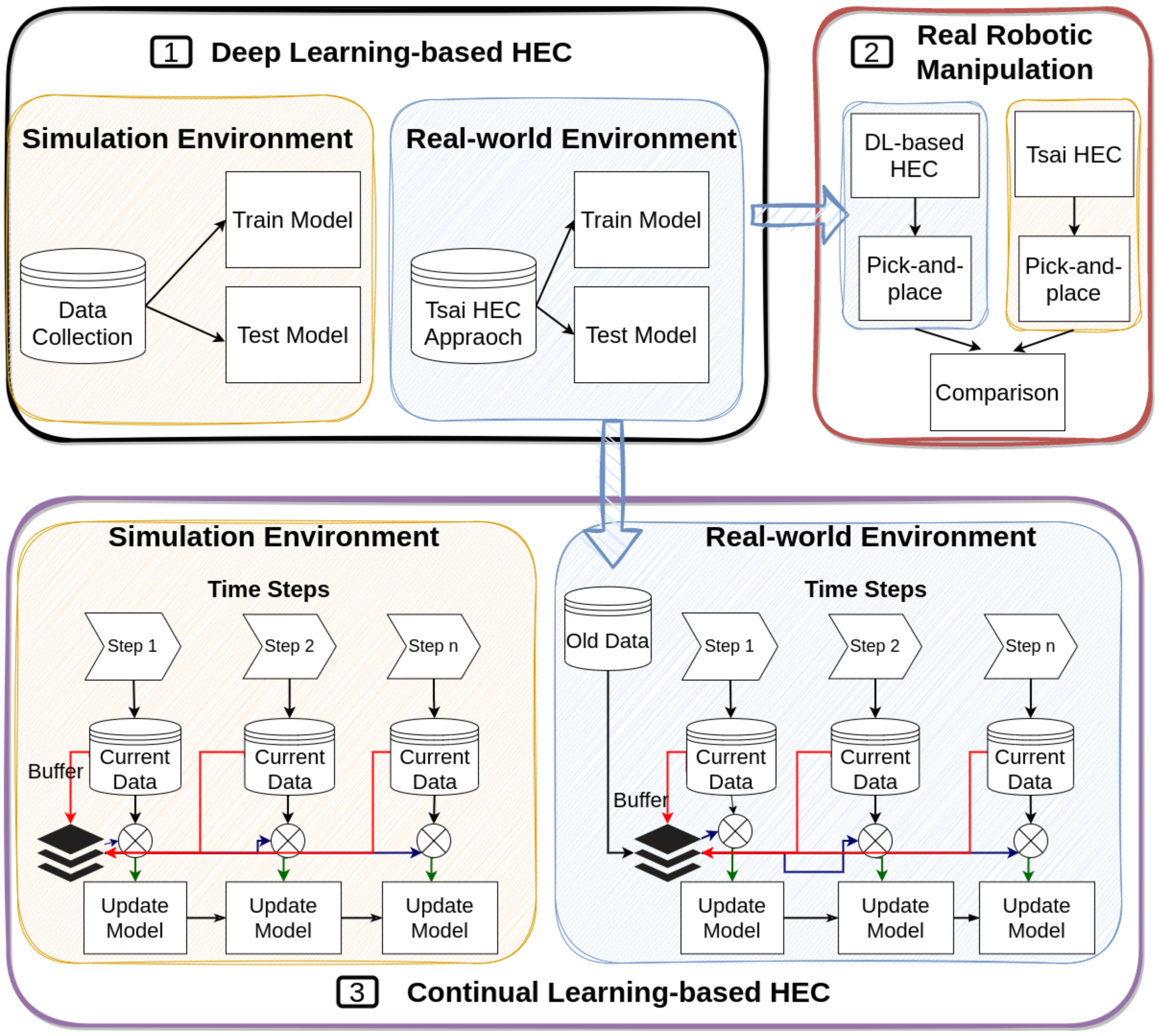


Figure 1.4: This figure depicts conducted three studies to develop a flexible and autonomous hand-eye calibration approach by using deep learning.

In the first study (Figure 1.4(1)), a deep learning-based HEC approach was developed by selecting a single reference point on the robot's end-effector through a 3D vision system. This approach allows the recalibration of the external camera poses without the need for data recollection. The experiments were conducted in both simulated and real-world environments to validate the effectiveness of the approach.

In the second study (Figure 1.4(2)), the deep learning-based HEC approach developed in the first study was tested on a real robotic manipulation task, specifically pick-and-place. Additionally, a classic HEC approach was used to make a comparison of the success of the approaches as well as their computational complexity and precision.

Finally, in the third study (Figure 1.4(3)), a continual learning-based HEC approach was proposed, which extends the learned calibration space through new observations over time. This approach aims to address the problem of catastrophic forgetting, which can occur when a deep learning model forgets previously learned information as it learns new information. The approach was evaluated through simulation and real-world experiments to demonstrate its effectiveness.

Overall, these three studies aim to develop a flexible and autonomous HEC approach using deep learning, which can be applied in dynamic robotic environments where the pose of the robot or camera changes over time to perform manipulation tasks.

1.5 The Significance of the Proposed Approaches

1.5.1 Contributions

The contributions of Chapter 3 are listed below:

- A deep learning-based regression architecture is proposed for estimating the camera pose by tracking a single reference point defined by the robot's kinematic chain. This architecture can be used as a hand-eye calibration model by automatically detecting the reference point using a 3D vision system.

- The developed deep learning-based HEC approach allows for estimating calibration parameters within seconds in the learned space during training, without the need for data recollection.

- The deep learning-based HEC approach achieves competitive results in terms of metric accuracy compared to classic HEC approaches that require data recollection for each camera pose. Additionally, the approach exhibits superior repeatability scores (precision) compared to state-of-the-art techniques, which is 7 and 3 times better than [18] and [19].

The contributions of Chapter 4 are listed below:

- Based on the experimental results, the deep learning-based HEC approach demonstrates comparable performance to the classic approach in executing a real-world robotic manipulation task.

- Furthermore, the deep learning-based HEC approach shows a reduction in computational complexity, as evidenced by the decreased number of attempts required to obtain calibration parameters and the reduced time needed for calibration when compared to the classic HEC approach.

The contributions of Chapter 5 are listed below:

- The results indicate that the hand-eye calibration problem can be addressed as a time-series problem, where the calibration parameters can be updated continuously over time as the camera pose changes.

- The proposed continual learning-based HEC approach utilises a buffer system to extend the learned calibration space with new observations without retraining starting from the beginning. This allows for the calibration parameters to adapt and improve over time, resulting in a more flexible and autonomous calibration approach.

1.5.2 List of publications

Published Papers

- Bahadir, Ozan, Jan Paul Siebert, and Gerardo Aragon-Camarasa. "A Deep Learning-Based Hand-eye Calibration Approach using a Single Reference Point on a Robot Manipulator." 2022 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2022. (Chapter 3)

Papers Under Review

- Bahadir, Ozan, Jan Paul Siebert, and Gerardo Aragon-Camarasa. "Performance Analysis of Hand-eye Calibration Approaches in a Real Robotic Manipulation Task." Engineering Applications of Artificial Intelligence. (Chapter 4)

- Bahadir, Ozan, Jan Paul Siebert, and Gerardo Aragon-Camarasa. "Extending Learned Hand-eye Calibration Space via Continual Learning." Robotics and Autonomous Systems. (Chapter 5)

1.6 The Outline of the Thesis

The thesis aims to address the limitations of existing hand-eye calibration (HEC) approaches by developing a flexible and autonomous deep learning-based HEC approach. Chapter 2 provides an overview of the HEC problem, current literature, and its limitations. Chapter 3 focuses on developing a deep learning-based HEC approach that can estimate calibration parameters without data recollection in the learned space; Chapter 4 evaluates the developed HEC approach on a real robotic manipulation task and compares its performance with a classic HEC approach; Chapter 5 proposes a continual learning-based HEC approach that can extend the learned calibration space over time. This chapter also investigates the approach's ability to address the catastrophic forgetting problem and enable the autonomous recalibration of the HEC system. The thesis concludes with Chapter 6, which summarises the main contributions and outlines potential future works. The proposed approach has the potential to be applied in various robotics and automation applications.

Chapter 2

Background and Literature Review

2.1 An Overview of Hand-eye Calibration Problem

Hand-eye calibration is the problem of finding the homogeneous transformation between the eye (camera) and the robot's end-effector [8]. Although this transformation is conceptually straightforward, it is essential for successful robot manipulation tasks as it enables us to relate camera observations to the robot's links or joints. The hand-eye calibration problem has been investigated for over 30 years via mathematical optimisation approaches [20]. These approaches have provided superior results under some particular constraints. The primary constraint of these approaches is that they are valid only for the current camera and the robot base. When the camera pose changes with respect to the robot base for the external hand-eye configuration, all data collection and optimisation should be repeated from scratch. With the increasing demand for flexible and adaptable robots that can quickly adapt to environmental changes, deep learning-based approaches have been proposed as a potential solution to the limitations of mathematical optimisation-based approaches. One of the primary challenges of deep learning-based approaches is the need for ground truth in the training stage. In addition, the hand-eye calibration problem has specific constraints, such as the orthogonality between each axis, that must be considered.

The hand-eye calibration problem can be decomposed into three main steps; camera calibration, robot calibration and the hand-eye calibration model. Besides these components, the success of the hand-eye calibration methods is highly affected by the adopted robotic environments and used technologies, i.e. cameras. To provide a comprehensive overview of the literature on hand-eye calibration, this thesis covers the adopted technologies, a brief background, camera calibration, robot calibration, classic hand-eye calibration models, and deep learning-based hand-eye calibration.

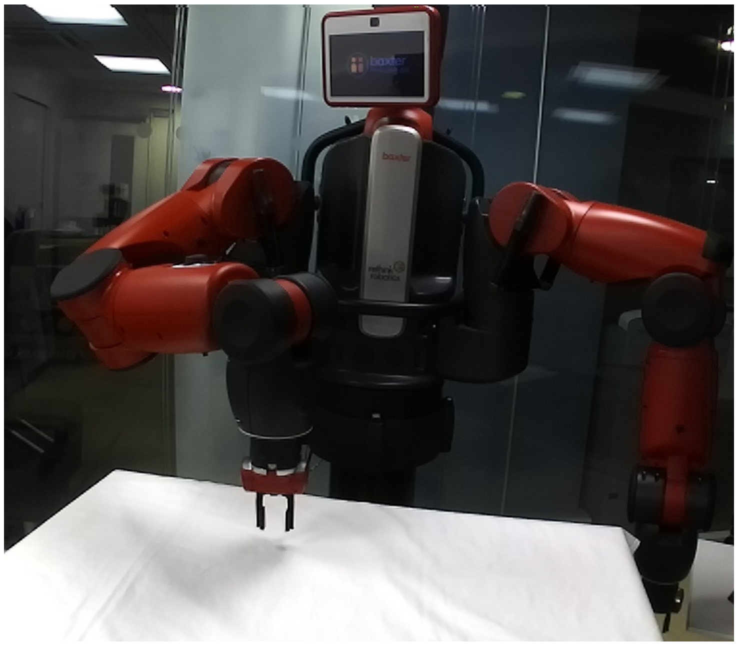


Figure 2.1: This figure shows the Rethink Baxter robot with two parallel grippers.

2.2 Hardware and Software

2.2.1 Robotic environments

In this thesis, two real-world robotic systems were used: The Rethink Baxter (Baxter) [21] and a Universal Robot (UR3) [22].

The Rethink Baxter robot

Baxter (depicted in Figure 2.1) has two arms with 7 degrees of freedom and three embedded cameras. One of the cameras is on the head, and the others are in the end-effector of the arms. It is compatible with two types of end-effectors: an electric parallel gripper and a vacuum cup gripper. In this thesis, we used the electric parallel grippers for both arms to manipulate objects. Baxter is also compatible with the Robot Operating System (ROS) [23].

UR3 robot

The UR3 (depicted in Figure 2.2) is a robotic arm with 6 degrees of freedom, and all its joints can rotate 360 degrees, and it has a 500 mm workspace range from its base. Although it has its programming tool and interface (PolyScope [24]), it is also compatible with ROS, and flexible for different types of grippers. In this thesis, we equipped the UR3 with a Shadow Modular Grasper (depicted in Figure 2.2) with three fingers and three joints for each finger (a total of 9 degrees of freedom).

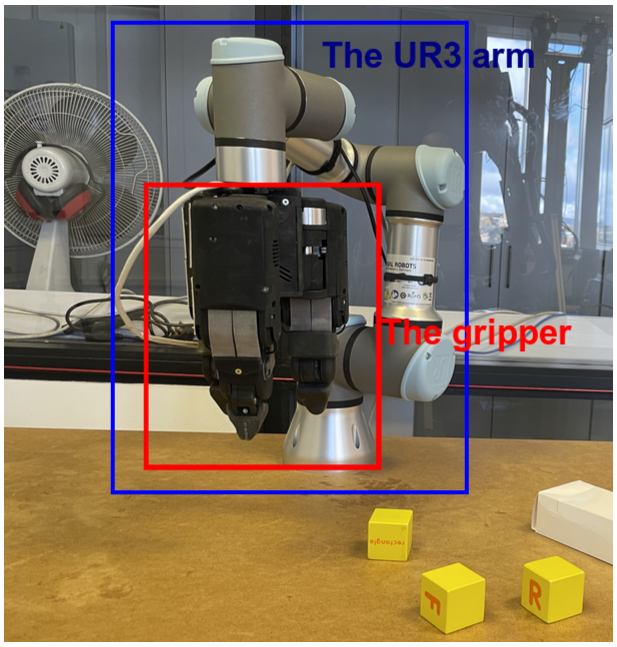


Figure 2.2: This figure depicts a Universal Robot 3 (UR3) equipped with a three-finger modular grasper.

2.2.2 Robot operating system (ROS)

The Robot Operating System (ROS) is an open-source robotic software framework [23] designed to mitigate large-scale software integration. ROS allows for communication between multiple machines and components, such as motors, robotic arms, and monitors. The framework is based on several key components, including nodes, messages, topics, and services [23]. Nodes are executable scripts which can be programmed in Python, C++, Octave [25] and LISP [26], and communicate with each other by using messages, which are the predefined typed data structures.To send messages, nodes publish topics, and to receive messages, nodes subscribe to topics. ROS also supports services for synchronous communication between topics that require request and response transactions. By facilitating communication and integration between components, ROS has become a popular framework for developing robotic applications. Examples of ROS-based robots include autonomous vehicles [27], drones, and industrial robots.

2.2.3 The StereoLabs camera (ZED)

The ZED camera (depicted in Figure 2.3) developed by Stereolabs [28] has been used for visual perception in this thesis. This camera is a stereo vision system, which consists of two RGB cameras separated by a fixed baseline. The depth map in this camera is calculated by the differences in matched points on two images from the left and the right camera. ZED camera enables the selection of different parameters that control the resolution of RGB image (up to 2k), the quality of depth map, and frame rate (up to 120 frames per second). Besides, it is compatible with ROS [23] and publishes different topics for visual perception, i.e. RGB images, depth

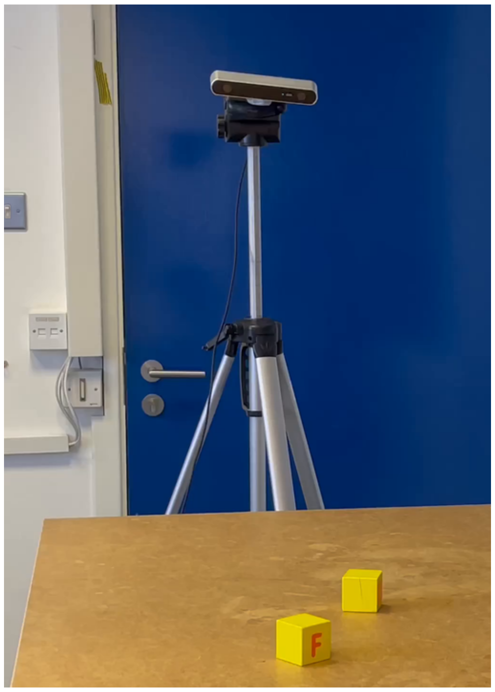


Figure 2.3: The ZED camera which has two RGB cameras (left and right)

maps, disparity images and camera intrinsic's parameters.

2.3 Background

2.3.1 Representation of position and orientation in 3D Space ($\mathbb{R}^3$)

The poses of point **P** with respect to the two reference frames (A and B) are shown in Figure 2.4, where $A_{\mathbf{p}}$ and $B_{\mathbf{p}}$ represent the positions of point **P** in the reference frames A and B, respectively. These reference frames have orthogonal axes in 3D Cartesian space. The displacement of reference frame B with respect to reference frame A is represented by vector $\mathbf{t}(x,y,z)$. The transformation from frame B to A includes both rotation and translation components, which ensure the orthogonality constraint between the axes of the two frames is maintained.

Orientation representation in the 3D Cartesian space ($\mathbb{R}^3$): Rotation in 3D is an operation which enables us to rotate the whole space with θ angle through counterclockwise along a fixed axis while the origin remains fixed. In 3D Cartesian space, this rotation can be represented via an 3×3 orthonormal matrix ($\mathbf{R}$) where $\mathbf{R}^{-1} = \mathbf{R}^T$ and $det(\mathbf{R})$=1.

Rotation matrices: The rotation matrix is a orthonormal matrix that represents a rotation in 3D space. It is a three-by-three matrix denoted by R($\hat{\mathbf{n}}$,θ), where $\hat{\mathbf{n}}$ is the unit vector of the rotation axis, and angle (θ) is the angle of rotation in a counterclockwise direction along the

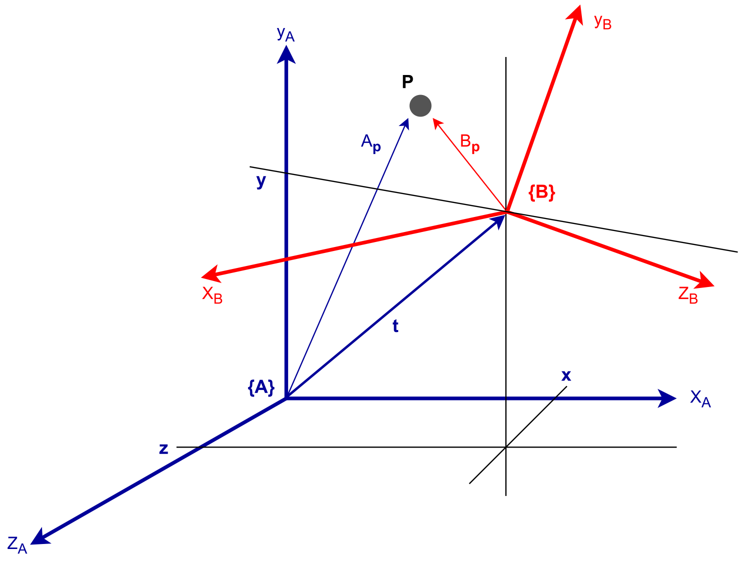


Figure 2.4: This figure shows two 3D coordinate frames ($\{A\}$ and $\{B\}$) and the pose of point **P** with respect to these frames.

axis. Equation (2.1) provides an example of this formulation, where a rotation is performed around the z-axis with a counterclockwise angle of θ.

$$R(\hat{\mathbf{z}},\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

The formulation in equation (2.1) can be extended via equation (2.2) for all three axes in 3D Cartesian space.

$$R(\hat{\mathbf{n}},\theta) = \begin{bmatrix} \cos\theta + n_1^2(1-\cos\theta) & n_1n_2(1-\cos\theta) - n_3\sin\theta & n_1n_3(1-\cos\theta) + n_2\sin\theta \\ n_1n_2(1-\cos\theta) + n_3\sin\theta & \cos\theta + n_2^2(1-\cos\theta) & n_2n_3(1-\cos\theta) - n_1\sin\theta \\ n_1n_3(1-\cos\theta) - n_2\sin\theta & n_2n_3(1-\cos\theta) + n_1\sin\theta & \cos\theta + n_3^2(1-\cos\theta) \end{bmatrix} \tag{2.2}$$

where,

$$\begin{gathered} \hat{\mathbf{n}} = (n_1, n_2, n_3) \\ n_1^2 + n_2^2 + n_3^2 = 1 \end{gathered} \tag{2.3}$$

In this formulation, the angle θ must be between zero to π:

- if theta is equal to zero, there is no solution (trivial solution),
- if theta is equal to π, R($\hat{\mathbf{n}}$,π) and R($-\hat{\mathbf{n}}$,π) yield the same solution,
- otherwise, there is a unique solution.

In this rotation representation, nine parameters must be estimated simultaneously under the orthogonality constraints among its axes. Besides, this representation is not intuitive; in other words, it is not easy to understand and interpret each element of the matrix presented in the equation (2.2).

Euler angles: Rotation in 3D Cartesian space can be represented by the product of three consecutive rotation matrices around different and orthogonal axes described in equation (2.4).

$$\begin{aligned} R(\hat{\mathbf{x}},\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\ R(\hat{\mathbf{y}},\theta) &= \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \\ R(\hat{\mathbf{z}},\psi) &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.4}$$

where, ϕ, θ and ψ are the Euler angles. In theory, there are 27 possible sequences to define a rotation with Euler angles. However, only 12 of these meet the constraint that two successive rotations are about different axes [29]. Equation (2.5) shows the three common sequences found in the literature, all of which can be described using a 3×3 matrix. Euler angles can be defined using two conventions: extrinsic and intrinsic. The intrinsic convention involves performing rotations using the coordinate system of the rotating object, while the extrinsic convention uses the external frame for rotations. Intrinsic rotations are performed in the same order as the formula, while extrinsic rotations are performed in the opposite order. Figure 2.5 shows the intrinsic convention of the Euler angle. The rotation begins with a ϕ angle, which maps $\mathbf{X}$ to x_1 and $\mathbf{Y}$ to y_1 while keeping the z-axis constant. The second rotation, which converts y_1 and z_1 to y_2 and z_2, is applied to the x_1 axis with a θ angle. *psi* angle is then applied via the z_2 axis, which transforms x_2 and y_2 into $\mathbf{X}$ and y, respectively.

$$\begin{aligned} R(\hat{\mathbf{n}},\theta) &= R_{ZXY}(\phi,\theta,\psi) = R(\hat{\mathbf{z}},\phi)R(\hat{\mathbf{x}},\theta)R(\hat{\mathbf{z}},\psi) \\ R(\hat{\mathbf{n}},\theta) &= R_{XYZ}(\phi,\theta,\psi) = R(\hat{\mathbf{x}},\phi)R(\hat{\mathbf{y}},\theta)R(\hat{\mathbf{z}},\psi) \\ R(\hat{\mathbf{n}},\theta) &= R_{ZYZ}(\phi,\theta,\psi) = R(\hat{\mathbf{z}},\phi)R(\hat{\mathbf{y}},\theta)R(\hat{\mathbf{z}},\psi) \end{aligned} \tag{2.5}$$

Despite the intuitive nature of Euler angles, the Gimbal lock problem occurs in all subsequences when one of the axes is rotated by a certain angle. The Gimbal lock occurs when two of the axes become aligned, resulting in the loss of one degree of freedom in describing the

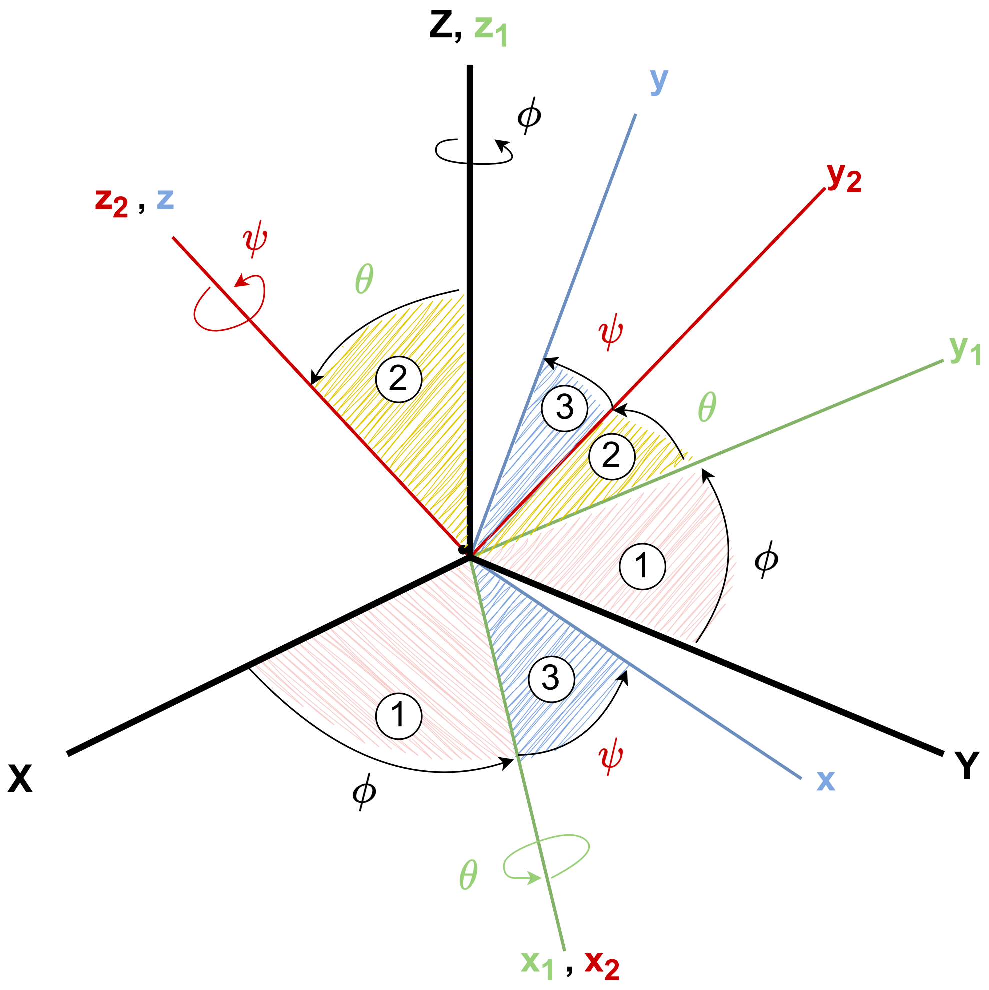


Figure 2.5: This figure shows ZXZ $(R(\hat{\mathbf{z}},\phi)R(\hat{\mathbf{y}},\theta)R(\hat{\mathbf{z}},\psi))$ Euler angle sequence application on XYZ reference frame to get xyz final frame.

rotation. For example, in the $R_{XYZ}(\phi,\theta,\psi)$ sequence, when θ approaches $\frac{\pi}{2}$, the second and third axes become aligned, making them indistinguishable [30].

Unit Quaternions: Quaternions are the three-dimensional extension of complex algebra [31]. A quaternion $(\mathbf{q})$ with scalar (q_0) and vectorial $(q_1,\ q_2,\ q_3)$ components can be expressed as follows,

$$\begin{aligned}\mathbf{q} &= q_0 + q_1 i + q_2 j + q_3 k \\ &= q_0 + \mathbf{q}_v\end{aligned} \tag{2.6}$$

where,

$$\begin{gathered} i^2 = j^2 = k^2 = ijk = -1 \\ ij = k \quad jk = i \quad ki = j \\ ji = -k \quad kj = -i \quad ik = -j \end{gathered} \tag{2.7}$$

The complex conjugate of the quaternion $(\mathbf{q})$, which is negating the vector part of the quaternion, is defined as follows :

$$\begin{aligned}\mathbf{q}^* &= q_0 - \mathbf{q}_v \\ &= q_0 - q_1 i - q_2 j - q_3 k\end{aligned} \tag{2.8}$$

Geometrically, the conjugate of a quaternion reflects the original rotation about the plane orthogonal to the axis of rotation. When we multiply the conjugate of the quaternion itself, we can get:

$$\begin{aligned}\mathbf{q}^*\mathbf{q} &= q_0 q_0 - \mathbf{q}_v \mathbf{q}_v \\ &= q_0^2 + q_1^2 + q_2^2 + q_3^2\end{aligned} \tag{2.9}$$

The norm of the quaternion ($|\mathbf{q}|$) equals to $\sqrt{\mathbf{q}^*\mathbf{q}}$:

$$|\mathbf{q}| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{2.10}$$

When the norm of the quaternion ($|\mathbf{q}|$) equals one, it is called the unit quaternion ($\mathbf{q}^\circ$). A unit quaternion ($\mathbf{q}$) can be described as follows:

$$\begin{aligned}\mathbf{q} &= q_0^2 + \mathbf{q}_v^2 = 1 \\ q_0^2 &= \cos^2\theta \\ \|\mathbf{q}_v\|^2 &= \sin^2\theta\end{aligned} \tag{2.11}$$

This implies that there is a θ angle between zero and π such that $\cos\theta = q_o$ and $\sin\theta = \|\mathbf{q}_v\|$. Then the unit quaternion ($\mathbf{q}$) can be also expressed as follow:

$$\mathbf{q} = \cos\theta + u\sin\theta \tag{2.12}$$

where,

$$u = \frac{\mathbf{q}_v}{\|\mathbf{q}_v\|} \tag{2.13}$$

In $\mathbb{R}^3$, any vector can be expressed as a pure quaternion (depicted in figure 2.6) where its scalar component (q_0) is zero: $\mathbf{q}^\circ(\upsilon) = 0 + xi + yj + zk$. A conjugate operator can be defined on a vector υ in $\mathbb{R}^3$ with the unit quaternion $\mathbf{q}$ (presented in equation(2.12)) as follow:

$$L_{\mathbf{q}}(\upsilon) = \mathbf{q}\upsilon\mathbf{q}^* \tag{2.14}$$

This operator represents the rotation of vector υ in $\mathbb{R}^3$ about axis u with θ angle. Unit quaternions are a commonly used representation for rotations in 3D space, as they are free from the issue of gimbal lock. However, the mapping from rotations to unit quaternions is not continuous, which can cause problems for numerical computations. This is because each rotation has two corresponding unit quaternions ($\mathbf{q}$ and $-\mathbf{q}$). This problem is called a double

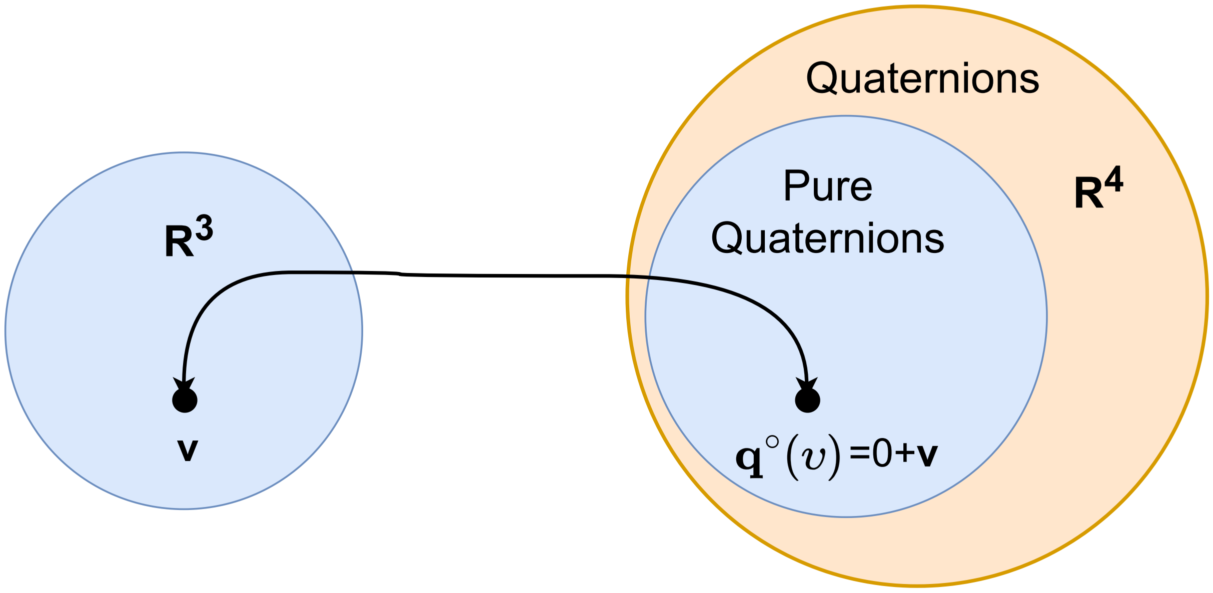


Figure 2.6: This figure shows the relationship of the vector υ in $\mathbb{R}^3$ and pure Quaternions space.

cover. To address this problem, some approaches have leveraged higher-dimensional spaces to represent rotations in a more continuous and unambiguous way.

According to Zhou *et al.* [32], deep learning methods may encounter to learn from discontinuous orientation representations, and it has been suggested that at least 5D or 6D parameters are needed to represent continuous (one-to-one mapping) orientations in 3D space. Peretroukhin *et al.* [33] proposed a 10D orientation representation based on a Quadratically Constrained Quadratic Program, which uses a symmetric 4×4 matrix with ten parameters to enable continuous rotation in 3D space. This representation allows for the eigenvalue decomposition of the matrix, which can be solved to obtain corresponding unit quaternion parameters. The use of a 10D representation in this context allows for a more continuous and unambiguous representation of camera pose orientation.

Linear Transformation in $\mathbb{R}^3$: The linear transformation of reference frame $\{B\}$ to $\{A\}$ (presented in figure 2.4) can be expressed as follow by using the homogeneous transformation representation:

$$\begin{pmatrix} A_x \\ A_y \\ A_z \\ 1 \end{pmatrix} \cdot \begin{bmatrix} {}^{A}\mathbf{R}_B & \mathbf{t} \\ 0_{1\times 3} & 1 \end{bmatrix} = \begin{pmatrix} B_x \\ B_y \\ B_z \\ 1 \end{pmatrix} \tag{2.15}$$

where

$$R = \begin{bmatrix} \mathbf{r_1} & \mathbf{r_3} & \mathbf{r_3} \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \quad \text{and} \quad t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

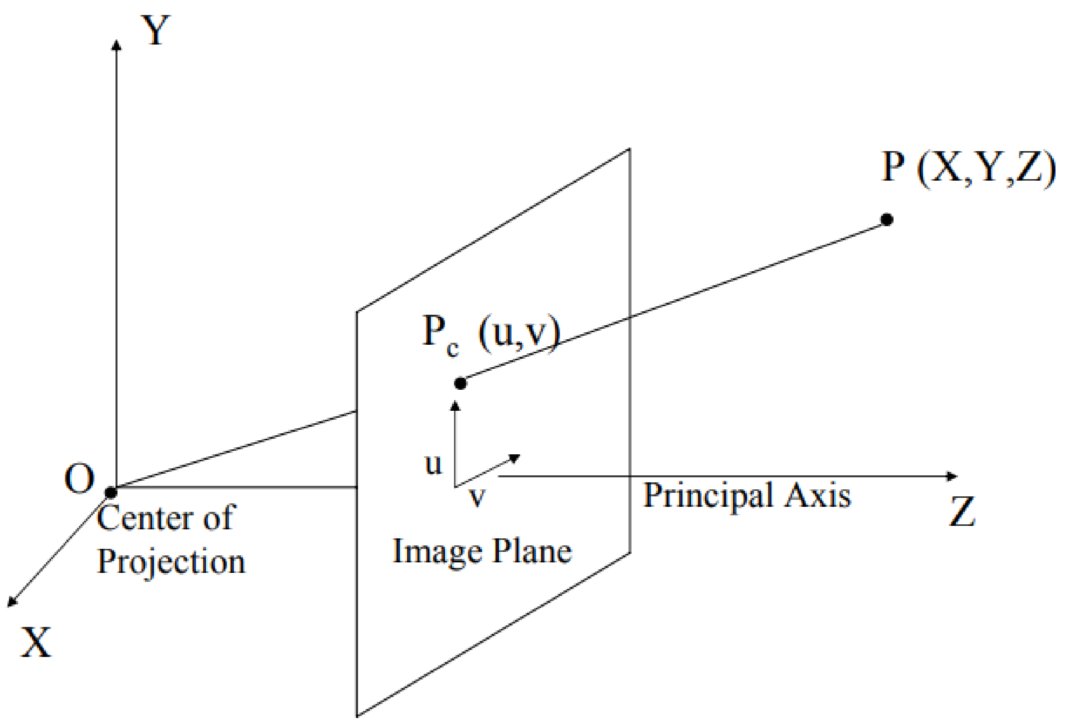


Figure 2.7: Pinhole Camera Geometry: **O** is the projection (camera) centre, and the principal axis lies towards the **Z** axis. **P** (X,Y,Z) and $P_c(u,v)$ represent the real-world (3D) and pixel (2D) coordinates of the same point.

Rotation (**R**) and translation (**t**) are two main components of linear transformation.

Hand-eye calibration problem can be solved by finding the transformation matrix that relates the end-effector's motion and the camera's motion with respect to a world reference frame. Linear transformation describes how the position and orientation of the camera and the robot's end-effector are related. By knowing the linear transformation between the two frames of reference, the relative position and orientation of the camera with respect to the robot's end-effector can be determined.

2.3.2 Image Formation

To enable robotic manipulation tasks based on camera images, it is necessary to transform pixel measurements in the image plane into world coordinates [34]. This process requires the acquisition of the camera's intrinsic and extrinsic parameters. The intrinsic parameters are used to transform 3D camera coordinates into 2D image coordinates and vice versa. The extrinsic parameters, which describe the camera's rigid transformation with respect to a fixed origin, transform 3D points from camera coordinates to world coordinates bidirectionally.

The projection of 3D points in the real world to 2D pixels in the image space needs a camera model including both the camera's intrinsic and extrinsic parameters. The pinhole camera model (depicted in Figure 2.7) is widely used in computer vision and robotics. The distance between

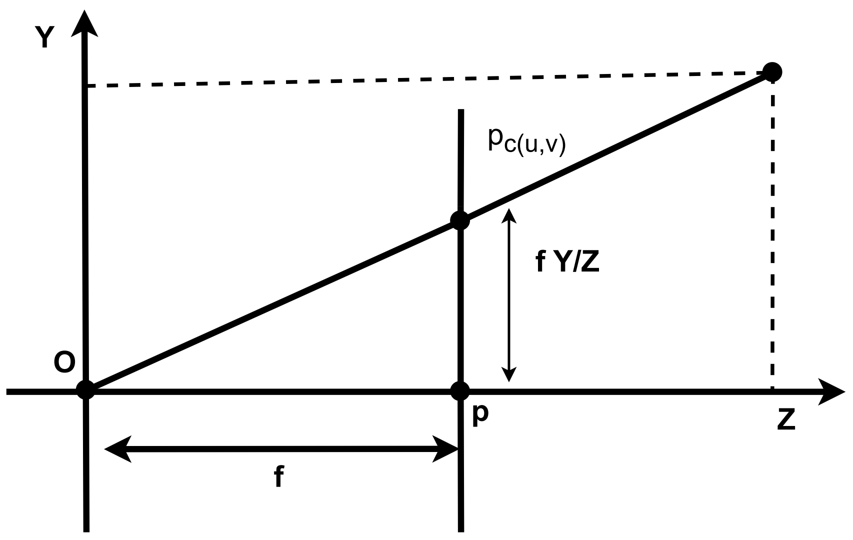


Figure 2.8: This figure shows the triangle similarity in the image plane.

the projection centre and the image plane is called the focal length (**f**). The projection of the world point (**P**) on the image plane is calculated by using the triangle similarity (in Figure 2.8) equations as follows:

$$\frac{f}{Z} = \frac{u}{X} = \frac{v}{Y} \tag{2.16}$$

which gives

$$u = \frac{fX}{Z} \quad \text{and} \quad v = \frac{fY}{Z} \tag{2.17}$$

The projection of the point **P** in the image plane P_c can be represented as follow:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{pmatrix} \tag{2.18}$$

Figure 2.9 shows the ideal and real image coordinate systems. The real image coordinate system considers the offset (u_0,v_0) between the ideal centre point **c** and the new centre **o**, and the non-orthogonal image coordinate system, which has θ angle among its axes. Let k_u and k_v be the orthogonal axes along u and v. We conclude that there is a transformation matrix (**H** detailed in equation (2.19)) which enables us to transfer point **p** (x,y) coordinates to the real image coordinates (u,v).

$$H = \begin{bmatrix} k_u & k_u \cot(\theta) & u_0 \\ 0 & k_v/\sin(\theta) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

Equation (2.18) can be extended by multiplying with matrix **H**. Equation (2.20) shows the camera intrinsic's matrix, which enables us to transform points for the camera to the image coordi-

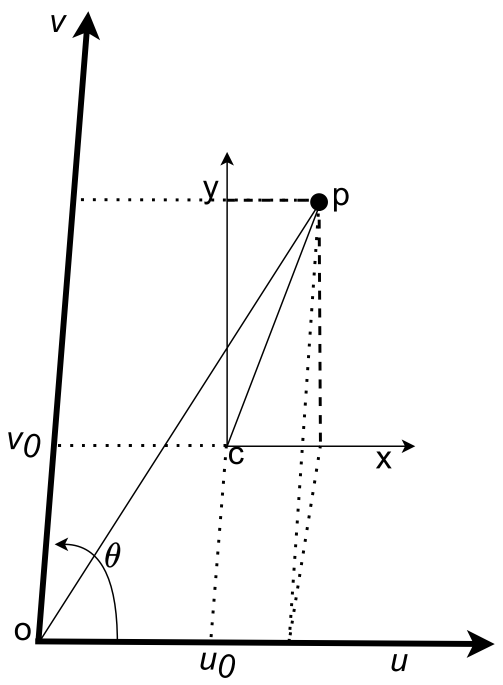


Figure 2.9: Image coordinate system: (c,x,y) is the ideal image coordinate system centred in the principal point, while (o,u,v) is the actual image coordinate system in which the centre is the upper left corner. θ shows the angle between the u and v axes.

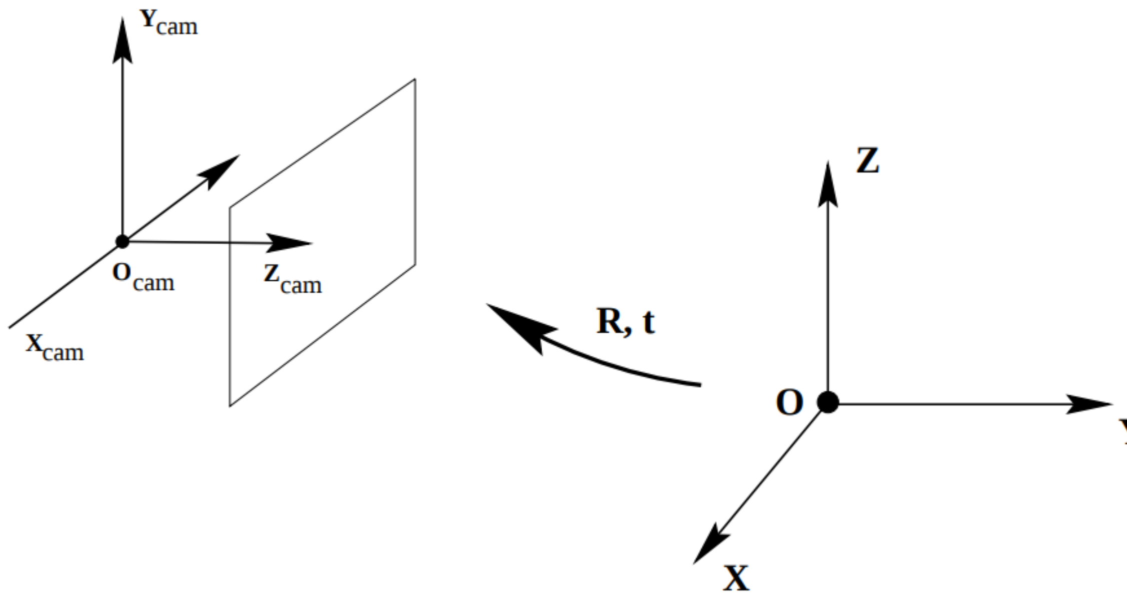


Figure 2.10: The euclidean transformation between the camera and the world coordinates. **R** and **t** are the 3x3 rotation and 3x1 translation matrices.

nate system. The matrix **K** can also be expressed in the homogenous coordinate system by using equation (2.21).

$$K = \begin{bmatrix} k_u & k_u \cot(\theta) & u_0 \\ 0 & k_v / \sin(\theta) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{bmatrix} fk_u & fk_u \cot(\theta) & u_0 \\ 0 & fk_v / \sin(\theta) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.20}$$

$$K = \begin{bmatrix} \alpha_x & s & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.21}$$

where

$$\alpha_x = fk_u$$

$$\alpha_y = fk_v / \sin(\theta)$$

$$s = fk_u \cot(\theta)$$

In this formulation, the camera's origin and the world's origin are the same. However, the camera and the world coordinate are different in most robotic manipulation or computer vision applications. Hence, the transformation between the camera and the world coordinate must be calculated. Figure 2.10 illustrates this transformation. The transformation consists of the rotation (**R**) and translation (**t**) components. Equation (2.22) shows the homogeneous transformation of the camera and world coordinate by using **R** and **t**. Equation (2.23) shows the straightforward camera model, including intrinsic and extrinsic camera parameters.

$$\begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.22}$$

where

$$P = \begin{bmatrix} {}^{cam}\mathbf{R}_{world} & \mathbf{t} \\ 0_{1\times 3} & 1 \end{bmatrix}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} \alpha_x & s & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{cam}\mathbf{R}_{world} & \mathbf{t} \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{pmatrix} X_{world} \\ Y_{world} \\ Z_{world} \\ 1 \end{pmatrix} \tag{2.23}$$

2.4 Camera Calibration

Two mainstream approaches have been widely employed in the literature to acquire the intrinsic and extrinsic parameters: marker-based approaches and self-calibration [20]. In marker-based approaches, a calibration target, such as a checkerboard and fiducial markers, is a specially designed object with a particular pattern whose shape and size are known. As for self-calibration, there is no calibration object, and the camera's parameters are estimated by extracting features in all images [20].

2.4.1 Marker-based approaches

The early camera calibration methods utilised 3D calibration objects to estimate the camera's intrinsic and extrinsic parameters by capturing their 2D pixel coordinates. The Direct Linear Transformation (DLT) method was employed in [35] and [36] to estimate these parameters. However, these methods did not consider the effect of lens distortion, and the accuracy of the calibration heavily relied on the precise manufacture of the calibration object [37]. Nonlinear optimisation-based methods were later proposed to handle the distortion problem. However, they require more parameters than the degree of freedom, which may lead to linear dependence among the parameters, making them sensitive to noise [4]. Additionally, these methods require reasonable initial values for the optimisation process to converge [38].

Tsai [4] is a pioneer of the 2D calibration target-based approach and proposed a two-stage camera calibration method combining linear estimation and nonlinear optimisation. Tsai defined the extrinsic parameters of the camera through the equation (2.22) and employed the Euler angles to represent the rotation component of the extrinsic parameters. Then, these parameters were estimated by using linear optimisation. As for internal camera parameters, a three-stage representation was used: ideal images coordinate via the simple pinhole camera model in equation (2.18), the second-order radial lens distortion, and the scale factor. The focal length ($\mathbf{f}$) was estimated using linear estimation in the first stage of the intrinsic parameters. Tsai then used nonlinear optimisation to estimate the lens distortion and scale factor, with the estimated focal length as an initial solution.

Zhang proposed a flat camera calibration approach using a 2D calibration target, the Checkerboard [5]. Zhang adjusted Equation (2.21) under the assumption of zero value of the Z axis. Zhang utilised the homography transformation from the calibration target to the image plane by adopting the tailored Equation (2.21). The camera's intrinsic parameters were calculated using the least squares error approach in this step. Then, the camera's extrinsic parameters were calculated via Equation (2.23) under the known camera's intrinsic parameters. Finally, Zhang refined the estimated parameters via nonlinear optimisation because the distance in the previous step is in parameter space and does not reflect the actual metric.

Although Tsai's [4] and Zhang's [5] approaches are the most widely used for camera calibration, their success highly depends on the pattern detection algorithm. Besides, they are offline approaches; hence, they are unsuitable for online camera calibration. To eliminate the effect of pattern detection for classic checkerboard calibration targets, QR-code-based approaches have been developed. ARToolkit, ARTag, and Aprilgrid are the prominent QR-code for camera calibration and robot localization [6]. ARToolkit markers consist of a black square with different internal symbols. Abdullah and Martinez [7] proposed ARToolkit marker-based camera calibration. They employed the epipolar geometry to get the camera's intrinsic parameters via equation (2.21) by using extracted 3D points in ARtoolkit on at least three different camera configurations. Then they extracted the camera's extrinsic parameters using the known intrinsic parameters. ARTag markers are 2D black square shapes, but unlike ARToolkit, white squares are inside. These white squares provide more stable marker detection and identification, unlike grey colour segmentation in ARToolkit. Aprilgrid markers have the same pattern as the ARTag. Olson [39] developed Aprilgrid-based camera calibration by matching the 3D centre of the segmented squares with their image coordinates via Direct Linear Transformation.

Camera calibration is an essential part of the hand-eye calibration problem, which involves determining the relationship between the camera and the robot's end-effector. Checkerboard and chessboard-based approaches have dominated the literature on camera calibration, but these targets are not very flexible when handling changes in camera pose. Specifically, when the camera's position or orientation changes, the calibration target must be re-attached to the robot or re-placed in the external space to collect all data from scratch. In contrast, QR code-based approaches are more flexible in handling camera pose changes. However, they can be susceptible to occlusion, sensitivity to lighting conditions, and dependence on QR code size in some applications. This thesis uses neither a classic calibration target nor a QR code for camera calibration due to these limitations.

2.4.2 Self-calibration approaches

Although calibration target-based approaches have dominated the literature and have had great success, they are not suitable for online camera calibration. Hence, the estimation of the camera parameters directly from the images has gained attention over several decades, known as self-calibration [40] [41]. In these methods, n key points ($n \geq 6$) are selected on the initial input image, and these key points are tracked while changing the perspective of the view by either moving the camera or the object in the scene. Then, the camera parameters are calculated by using the differences between these key points on different images. However, these approaches assume that the camera or the robot's motion is known.

The literature on self-camera calibration is relatively sparse, likely due to the requirement for a significant amount of motion and texture in the environment to detect and track key points accurately. However, in this thesis, an approach for explicit and implicit camera calibration, detailed in Chapter3, can be classified as a form of self-calibration. In contrast to traditional self-calibration approaches, the developed approach does not rely on key points detection and tracking. Instead, a single reference point is selected on the robot end-effector to serve as a reference frame.

2.5 Robot Calibration

For hand-eye calibration approaches, the assumption is that the robot is calibrated. However, a robot calibration solution can become outdated because of the displacement of the robot joints due to wear and tear. This source of errors affects the hand-eye calibration success because of the difference between the end-effector's actual pose and the controller's nominal pose. Hence, robot calibration is vital for hand-eye calibration. Robot calibration can be divided into four steps [42]: robot kinematic model selection, pose measurement approaches, optimisation methods, and pose redefinition. Although approaches such as the product of exponents (POE) [43] and quaternion model (QM) [44] exist for robot kinematic model representation, Denavit-Hartenberg (DH) parameters [45] and its extension [46] (which eliminates the singularity problem) are the most popular in the literature. For pose measurement, laser-based approaches [47] have emerged as a powerful method in the robot calibration literature. However, the cost of these specialised devices is considerable. Hence, planar contact [48, 49], visual observation [50], self-touch [51] and the combinations of them [52–54] have arisen as powerful methods in the robot calibration literature.

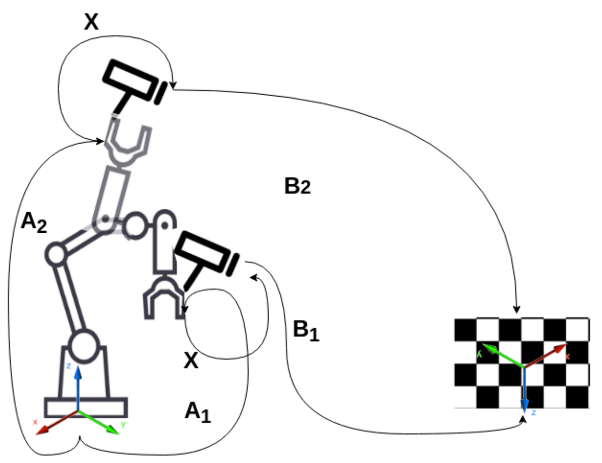
In conclusion, hand-eye calibration approaches rely on robot calibration to achieve accurate results. It is essential to apply the proper calibration techniques to make sure that the robot's joints are correctly aligned because wear and tear might cause robot calibration to become outdated. In this thesis, we did not employ any robot calibration approach. However, the proposed deep learning-based hand-eye calibration approach (detailed in Chapter3) can alleviate this problem because it is based on camera observation from different points of view, and these observations are linked with the robot workspace.

2.6 Classic Hand-eye Calibration

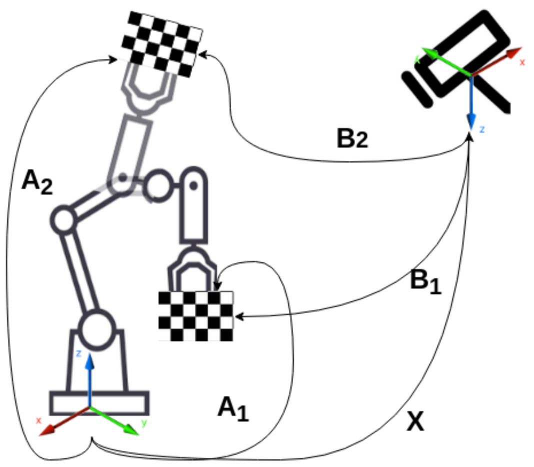
This section details the classic hand-eye calibration approaches regarding adopted formulations and metric error. $AX = XB$, $AX = YB$ and reprojection error are the mainstream methods for classic hand-eye calibration approaches. Additionally, each subsection contains solution approaches for these formulations. Table 2.1 shows the classification of the approaches in terms of the formulations and solution methods. The main limitation of these classic approaches is that they are not flexible against changes in the camera pose. More detailed limitation for each formulation is presented in subsections.

Table 2.1: An Overview of the Classic Hand-eye Calibration Approaches

Approach	**Formulation**	**Solution Method**	**Orientation Representation**
Tsai [8]	AX=XB	Separation	Euler angles
Chou and Kamal [9]	AX=XB	Separation	quaternion
Andreff et al [10]	AX=XB	Separation	Axis angle and quaternion
Horaud [11]	AX=XB	Simultaneous	quaternion
Daniilidis [10]	AX=XB	Simultaneous	Dual quaternion
Zhao [55]	AX=XB	Simultaneous	Rotation matrice and Quiaternion
Zhuang *et al.* [13]	AX=YB	Separation	quaternion
Dornika and Haroud [14]	AX=YB	Separation	quaternion
Shah [15]	AX=YB	Separation	Kronecker delta
Dornika and Haroud [14]	AX=YB	Simultaneous	quaternion
Li et al [56]	AX=YB	Simultaneous	Dual quaternion
Tabb and Yousef [16]	AX=YB	Simultaneous	Euler angles
Zhao [17]	AX=YB	Simultaneous	Kronecker delta and Dual quaternion
Zhi and Schwertfeger [57]	AX=XB	reprojection error	Kronecker delta
Ali *et al.* [58]	AX=XB and AX=YB	reprojection error	quaternion
Tabb and Yousef [16]	AX=YB	reprojection error	quaternion, Euler angles And the axis angle



(a) eye-in-hand



(b) external camera configuration

Figure 2.11: This figure shows the eye-in-hand (a) and external camera (b) configurations.

2.6.1 AX=XB Formulations

$\mathbf{AX} = \mathbf{XB}$ is the most preferred formulation of the hand-eye calibration problem in the classic approaches because of its simple and easily understandable nature. Figures 2.11(a) and 2.11(b) show the geometric interpretation of this formulation for two primary visual modalities. Figure 2.11(a) represents the eye-in-hand configuration, in which the camera is attached to the robot end-effector. In contrast, figure 2.11(b) shows the external camera configuration, where a calibration target is attached to the robot end-effector. In this formulation, **X** is the unknown transformation between the camera base and the robot's base or end-effector. **A** and **B**, which are observable, are the linear transformations of the end-effector's poses with respect to the robot base and the poses of the camera with respect to the calibration target in n successive configurations, respectively. Equation (2.24) shows the formulation of these linear transformations (**A** and **B**).

$$\begin{gathered}
A_1 \cdot \mathbf{X} \cdot B_1 = A_2 \cdot \mathbf{X} \cdot B_2 \\
A_2^{-1} \cdot A_1 \cdot \mathbf{X} \cdot B_1 \cdot B_1^{-1} = A_2^{-1} \cdot A_2 \cdot \mathbf{X} \cdot B_2 \cdot B_1^{-1} \\
A_2^{-1} \cdot A_1 \cdot \mathbf{X} = \mathbf{X} \cdot B_2 \cdot B_1^{-1} \\
\mathbf{A} = A_2^{-1} \cdot A_1 \quad \text{and} \quad \mathbf{B} = B_2 \cdot B_1^{-1}
\end{gathered} \tag{2.24}$$

$$\mathbf{A} \cdot \mathbf{X} = \mathbf{X} \cdot \mathbf{B} \tag{2.25}$$

where

$$\begin{bmatrix} R_A & t_A \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_X & t_X \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_X & t_X \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_B & t_B \\ 0 & 1 \end{bmatrix} \tag{2.26}$$

Equation (2.25) shows the general formulation of $\mathbf{AX} = \mathbf{XB}$ formulation and how it is derived. Equation (2.26) shows the homogenous transformation-based matrix formulation. Equation (2.25) can be decomposed into two components represented by equations (2.27) and (2.28). There are two mainstream solution approaches: separately and simultaneously. In separate solutions approaches, equation (2.27) and equation (2.28) are considered independently, while simultaneous approaches solve these equations at the same time.

$$R_A R_X = R_X R_B \tag{2.27}$$

$$R_A t_X + t_A = R_X t_B + t_x \tag{2.28}$$

Separation method

Tsai [8] proposed a separation-based hand-eye calibration approach by adopting the $\mathbf{AX} = \mathbf{XB}$ formulation. To represent the orientation component, he converted equation (2.27) to equation (2.2). He then solved equation (2.2) to compute R_X using the least squared error method. The translation component was then calculated by substituting the computed R_X into equation (2.28). While Tsai's method produces satisfactory outcomes, it requires expertise for data collection and has limitations on the distance between the camera and the calibration target. Additionally, a solution does not exist for 0 and π because Tsai's approach uses Euler angles as described in section 2.3.1. To address these issues, Chou and Kamal [9] employed quaternion to represent the rotation component in the $\mathbf{AX} = \mathbf{XB}$ formulation. They decomposed the quaternion-based rotation matrix (R_X) using the Singular Value Decomposition (SVD) method and solved the equations via the closed-form method. Finally, they computed the translation component by replacing the computed R_X into equation (2.28).

Calibration target-based approaches typically rely on the relationship between $3D$-to-$2D$ key points on a calibration target to determine the camera poses for n frames. In contrast, self-calibration approaches estimate the camera pose for each frame using $2D$-to-$2D$ key point matching. Andreff *et al.* [10] proposed a self-calibration approach based on the **AX=XB** formulation within the structure-from-motion paradigm. Unlike previous approaches, Andreff *et al.* [10] used $2D$ key point matching from n different frames to estimate the camera pose. Two methods were developed in [10]: calibration targets for key points **(M1)** and self-calibration **(M2)**. While the calibration target-based method produces similar results to those achieved by [8], [11], and [12], the accuracy of the self-calibration method **(M2)** is lower, with a 17-fold decrease compared to **(M1)**, and compared to those achieved by [8], [11], and [12].

Simultaneous method

Separation methods are commonly used for hand-eye calibration, but they can suffer from error transfer, where the estimation error in the rotation matrix is transferred to the translation vector. To address this issue, simultaneous solution approaches have been developed. Unlike separation methods, simultaneous solution methods jointly estimate the rotation and translation matrices, thereby preventing error transfer.

Horaud and Dornika [11] tailored $\mathbf{AX} = \mathbf{XB}$ formulation to eliminate explicit camera calibration as follows:

$$\mathbf{MY} = \mathbf{M}^{'}\mathbf{YB} \tag{2.29}$$

where **Y**is the unknown transformation between the robot's end-effector to the calibration target frame. Daniilidis [12] enhanced Horaud [11] approach by representing the rotation components of the transformation through dual quaternion. Thereby, he transformed the hand-eye calibration problem into a linear equation system. Then he employed the SVD to find the rotation and translation components of the unknown transformation. [12] is the first approach that finds hand-eye calibration parameters simultaneously using linear optimisation.

In equation (2.21), the initial perspective transformation matrix between the calibration target and the camera frame, denoted as **M'**, is explicitly generated and kept constant for *n* frames. On the other hand, **M** represents the perspective transformation matrix for *n* consecutive frames and varies for each frame. **B** is the transformation of the robot's end-effector with respect to the robot base in *n* successive movements, as in the $\mathbf{AX} = \mathbf{XB}$ formulation. The rotation component of the transformation is represented using quaternion, and the translation and rotation components are estimated using nonlinear objective functions and the Levenberg-Marquardt nonlinear optimisation method. Horaud and Dornika [11] eliminate transferring errors from the rotation to translation in separation methods. In addition to, it makes a more flexible hand-eye calibration system enabling different camera models. However, it does not provide the transformation between the robot's end-effector and the camera because **Y** in [11] is the transformation between the robot's end-effector and the calibration target frame. Hence, it requires a linear optimisation step to find **X** using the estimated **Y**.

Daniilidis [12] further improved Horaud's approach by using dual quaternion to represent the rotation components of the transformation, leading to a linear equation system. The rotation and translation components of the unknown transformation are estimated using the SVD, and the hand-eye calibration parameters are found simultaneously using linear optimisation.

Zhao [55] developed a convex optimisation-based hand-eye calibration approaches with two different rotation representations: rotation matrices and quaternion. Unlike previous nonlinear optimisation approaches, they employed L_∞ norm-based objective function that eliminates defining the good initial solutions in previous approaches. Their quaternion-based approach has competitive results with classic nonlinear optimisation-based approaches while eliminating the dependence on good initial values.

2.6.2 AX=YB Formulations

The end-effector motion's order significantly impacts the performance of $\mathbf{AX} = \mathbf{XB}$ formulation-based techniques. This implies that even little modifications to the robot's trajectory can cause large errors in the calibration. As a result, $\mathbf{AX} = \mathbf{YB}$ has emerged as a novel formulation to address the issues in the hand-eye calibration problem. In this formulation, $\mathbf{Y}$ is the geometric transformation from the robot base to the world coordinate base, and $\mathbf{X}$ is the hand-eye transformation. $\mathbf{A}$ and $\mathbf{B}$ stand for the pose of the robot's end-effector and the pose of the camera, respectively.

Equation (2.31) shows the mathematical model of the **AX=YB** formulation. This model can be decomposed into two components, like in the **AX=XB** formulation. Equations (2.32) and (2.33) show the rotation and translation components.

$$\mathbf{A} \cdot \mathbf{X} = \mathbf{Y} \cdot \mathbf{B} \tag{2.30}$$

where,

$$\begin{bmatrix} R_A & t_A \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_X & t_X \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_Y & t_Y \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_B & t_B \\ 0 & 1 \end{bmatrix} \tag{2.31}$$

$$R_A R_X = R_Y R_B \tag{2.32}$$

$$R_A t_X + t_A = R_Y t_B + t_Y \tag{2.33}$$

Separation method

Zhuang *et al.* [13] proposed a separation-based hand-eye calibration approach by representing rotation with unit quaternion. They solved the rotation component (equation (2.32)) using SVD. Then they solved equation (2.33) by substituting $\mathbf{R_y}$ estimated previously and using linear optimisation. However, their method transfers errors from the rotation to the translation same as the **AX=XB** formulation with separation solution approaches. Besides, their method suffers from a singularity problem when the angle between two successive calibration frames is zero or π. Dornaika and Horaud [14] tailored [13] approach using a closed-form solution. For equation 2.32, they established a positive quadratic objective function with a unit Quaternion the same as in [13]. They applied these constraints to their objective function with two Lagrange multipliers to guarantee unit Quertarnions for $\mathbf{R_X}$ and $\mathbf{R_Y}$. After minimising this function, they used the least squared error method to solve the equation (2.33). Shah [15] employed the Kronecker product method to represent rotation presented in equation (2.32). The author obtained the rotation matrices $\mathbf{R_X}$ and $\mathbf{R_Y}$ through the use of singular value decomposition (SVD). Subsequently, Shah used the least squared method to calculate translations for $\mathbf{t_X}$ and $\mathbf{t_Y}$ with estimated rotations.

Simultaneous method

Dornaika and Horaud [14] presented a simultaneous solution-based approach in addition to the closed-form method. They devised a nonlinear objective function which consists of four components with Lagrange multipliers. The first component is the estimation error of the rotation. The rotation was represented by the rotation matrix, which has 18 parameters. The second component is the sum of translation errors (6 parameters). The other components are the orthogonality constraints for $\mathbf{R_X}$ and $\mathbf{R_Y}$. The success of this approach highly depends on the selection of the Lagrange multipliers. Moreover, it requires good initialising for the nonlinear optimisation process to get the global solution.

Li *et al.* [56] proposed two methods to obtain hand-eye calibration parameters simultaneously with closed-form methods. In their first method, they represented rotation with dual quaternion and simultaneously obtained rotation and translation components using SVD. Furthermore, the authors utilised the Kronecker product to transform the problem into a new linear equation system, which is solved using the least squares method.

Tabb and Yousef [16] proposed an iterative approach with AX=YB formulation to obtain hand-eye calibration parameters simultaneously. This approach consists of three main steps: initialisation, iterative optimisation, and refinement. In the initialisation step, an initial guess for the hand-eye calibration parameters is obtained using a closed-form method. In the iterative optimisation step, the authors iteratively refine these parameters by solving linear equations relating to the robot and camera measurements. In the refinement step, the authors further improve the calibration accuracy by using a nonlinear optimisation algorithm to minimise the difference between the predicted and actual measurements.

Zhao [17] proposed two semi-convex optimisation approaches to simultaneously obtain rotation and translation of unknown **X** and **Y**. The author converted the **AX=YB** formulation into a semi-convex optimisation problem utilizing the Kronecker product in their first approach. In the second approach, the semi-convex optimisation problem was reconstructed using the dual quaternion representation.

2.6.3 Reprojection error-based approaches

The **AX=XB** and **AX=YB** methods for hand-eye calibration are limited in their ability to handle noise and outliers coming from camera calibration. To eliminate this problem, reprojection error-based hand-eye calibration approaches have been developed. In these approaches, **X** and

Y are obtained by using the reprojection error detailed in equation (2.34) without explicit **A** and **B**.

$$error = \sum_{i=1}^{n} \sum_{i=m}^{m} \left\| p_{ij} - \overline{p}_{ij} \right\|^2 \tag{2.34}$$

where n and m are the total numbers of frames (stations) and key points.

The equation represents the summation of the squared Euclidean distances between the observed image coordinates p_{ij} and the corresponding estimated coordinates $\overline{p}_{ij}$ for all key points m across all frames n.

Zhi and Schwertfeger [57] developed a projection error-based hand-eye calibration approach with **AX=XB** formulation. They first extract features on the camera scene and match them in n frames. The RANSAC algorithm [59] was used to eliminate outliers. They transform **AX=XB** formulation into a linear equation system by using Kronecker delta and skew-symmetric matrix. Then they obtained the initial hand-eye calibration parameters via the least-squared linear optimisation method. To refine these initial hand-eye calibration parameters, they applied bundles adjustment [60] and triangulation [61] several times. They do not use any calibration target in this method. However, this method assumes that the camera's intrinsic parameters are known.

Ali *et al.* [58] proposed several approaches for hand-eye calibration using unit quaternion-based rotation representation. They considered **AX=XB** and **AX=YB** formulations with different error metrics, including the real error metric (homogeneous transformations) and the projection error. To improve the accuracy of their projection error-based methods, they proposed a nonlinear objective function that incorporates the camera's internal and external parameters. In their paper, Ali *et al.* provided a comprehensive review and comparison of existing methods for simultaneous robot-world-hand-eye calibration with their proposed approaches. To evaluate the performance of their methods, they created six datasets consisting of three real-world and three simulation scenarios with varying image resolutions and numbers of collected poses. Their approaches performed well in some specific scenarios and sub-evaluation metrics, outperforming other hand-eye calibration methods. However, these approaches were not as flexible and robust for all scenarios, and their performance varied depending on the calibration dataset and the specific evaluation metric. Specifically, their projection error-based methods struggled to handle noisy data and could be sensitive to initialisation.

Tabb and Yousef [16] conducted a study to investigate the impact of different rotation representation methods on the accuracy of hand-eye calibration. They developed two nonlinear objective functions that consider the camera's intrinsic and extrinsic parameters, which can be used for separate or simultaneous solution approaches. They employed quaternion, Euler angles, and axis-angle representations to parameterise these objective functions. Their study pointed out that the choice of rotation representation significantly impacted the accuracy of both real metrics and reprojection errors. Specifically, they found that the quaternion-based method performed best, while the axis-angle method performed the worst. However, they noted that the performance of each method was highly dependent on the quality of the initial solution. This is because their nonlinear objective functions require nonlinear optimisation solvers, and the initial solution heavily influences the success of these solvers.

2.7 Deep Learning-based Hand-eye Calibration

While mathematical optimisation-based (classic) hand-eye calibration approaches have been successful for fixed external camera configurations, they lack flexibility when the camera's pose changes with respect to the robot base or end-effector. Deep learning has emerged as a promising approach for developing flexible hand-eye calibration systems because of its ability to generalise from data and fast approximation capabilities when there are enough training samples. Compared to classical approaches, deep learning-based hand-eye calibration methods can accommodate changes in camera configuration and provide more accurate and reliable calibration results. These methods typically require a large amount of data for training, which can be time-consuming and expensive to collect. However, recent data generation techniques and hardware acceleration advancements have made obtaining the necessary data more accessible and cost-effective.

The literature on deep learning-based hand-eye calibration approaches is relatively sparse compared to classical approaches. This is attributed to the fact that, in the early stages of hand-eye calibration research, the application areas, such as robotic manipulation, 3D reconstruction, and augmented reality, did not require a high degree of flexibility, and classical approaches yielded satisfactory results for these stationary systems. Robotic systems require a degree of flexibility, resulting in flexible hand-eye calibration systems. To meet these requirements, deep learning-based approaches have gained attention and are being developed to provide flexible and online calibration systems that can adapt to varying environmental conditions and system dynamics.

Lambrecht [62] proposed a deep learning-based approach to make a flexible hand-eye calibration approach. He employed the faster R-CNN (Region-based Convolutional Network method) [63] method to find the bounding box of key points. They selected the robot joints as key points because their 3D pose is known through the robot kinematic chain. After obtaining 2D key points, their corresponding 3D and the camera's intrinsic parameters were given to the Perspective-N-Point (PnP) [64] algorithm to find hand-eye calibration parameters. PnP is an algorithm enabling to derive the camera pose, encompassing both translation and orientation, from a set of n 3D points in real-world space and their corresponding 2D projections within the image plane. This derivation relies on the geometric relationships inherent to these n points. This method uses deep learning as a feature detection algorithm. Moreover, although it makes a more flexible hand-eye calibration system, empirical results demonstrate a relatively diminished efficacy compared to conventional approaches. Lee *et al.* [18] presented the same method as in [62].However, they devised an encoder-decoder-based network to obtain 2D key points on the robot joints. Unlike [62], they trained their network only with simulation data and employed domain adaptation techniques to deploy the trained method on real robotic environments.

While [62] and [18] employ deep learning as a feature detection algorithm, it is possible to use deep learning architectures as a direct method for estimating hand-eye calibration parameters. Valassakis *et al.* [19] developed three end-to-end deep learning-based architectures for this purpose. In their first method, they used a deep learning-based regression architecture to process RGB images and estimate hand-eye calibration parameters directly. Their network produced nine parameters, the first three representing the translation component and the remaining six converted to the unit quaternion using a 6D representation [32]. In their second approach, they employed a deep learning architecture to estimate 2D key points on the robot's end-effector and then used the PnP algorithm to obtain hand-eye calibration parameters. In their final approach, they used three networks for the depth map, segmentation mask of the robot's end-effector, and initial hand-eye calibration parameters. These networks, together with the camera's intrinsic parameters and the model of the robot end-effector, were combined, and the Iterative Closest Point (ICP) [65] approach was used to refine the hand-eye calibration parameters estimated by the networks. All three approaches were trained on simulated and real-world environments, and the direct regression architecture produced the best results, which implies that end-to-end deep learning-based architectures have a vast potential to develop flexible hand-eye calibration systems. This paper is the state-of-the-art result of deep learning-based hand-eye calibration, and their results are comparable with the classic approaches. Their results also indicate that orientation representation is crucial for deep learning architecture, similar to in classical approaches. Continuity and the one-to-one relationship between the orientation representation space and the 3D Cartesian Space must be ensured to train deep learning architectures smoothly.

Indeed, the predominant paradigms within deep learning-based methodologies [18, 62] for external camera configuration in hand-eye calibration involve using deep learning for feature extraction. Integrating auxiliary algorithms complements this practice to deduce the requisite calibration parameters. Consequently, any inaccuracies in feature detection can potentially be propagated to the subsequent additional algorithm, influencing the overall performance of these approaches.

2.8 Continual Learning

Classic deep-learning architectures have exhibited superior performance to human capabilities in specific tasks, notably evidenced in domains such as Atari games [66] and object recognition [67]. However, their training paradigm is predominantly offline, characterised by the availability of the entire training dataset at the inception of the training process. Subsequently, the trained model remains static and is not subject to continuous updates during testing. Consequently, their adaptability to novel streaming data scenarios is constrained, particularly in the face of evolving data distribution dynamics over time [68]. As human beings, we are capable of learning new tasks sequentially while preserving previously known concepts. However, when the stream data are added to offline trained models, classic deep learning approaches face the catastrophic forgetting problem, which means that learned concepts are gradually forgotten. This catastrophic forgetting problem is also referred to as stability-plasticity [69], where stability and plasticity refer to retaining previous knowledge and the ability to integrate new observations.

Continual Learning (CL) has emerged as a powerful method to address the stability-plasticity dilemma in machine learning. CL algorithms enable the model to learn from a continuous stream of new data, which can be categorised into three primary scenarios [70]. The first scenario is task-incremental learning, where the model learns isolated tasks sequentially over time from new data. For instance, the model may learn sub-tasks such as reaching, picking, and placing objects incrementally in robotic manipulation tasks. The second scenario is domain-incremental learning, where the model adapts to new observations with a different distribution than the original training data. This scenario is characterised by concept drift, where the input data distribution changes over time. An example of this scenario could be adapting an object detection model for autonomous driving to different weather conditions. The third scenario is class-incremental learning, where the model needs to classify an increasing number of classes over time. In this scenario, the model must simultaneously deal with concept drift and learn new class labels for the new observations.

CL approaches can be classified into three categories to handle the aforementioned scenarios: parameter isolation (architectural design), replay-based (memory), and regularisation-based approaches. Parameter isolation (architectural design) methods aim to handle the stability-plasticity problem by dedicating each task to different parameters. These methods can be categorised as dynamic and static models. In dynamic models, there is no limit to architecture size, and when a new task is introduced, the network is expanded. Meanwhile, the weights associated with previous tasks are frozen [71] or copied [72] to overcome the stability (catastrophic forgetting) problem. Static models are fixed-size architectures, and each task has its dedicated weights [73, 74]. Replay methods sampled previously used raw data while updating the model with new stream data to overcome the stability problem. The most straightforward approach is naive memory [75], which randomly stores a subset of the previous data. Besides, the nearest-mean classifier [76], reservoir sampling which is a uniform distribution-based random selection [77], and their combination [68] are the main approaches for selecting samples from previous data. Regularisation-based approaches modify the loss functions to penalise changes in important weights. This method has two main strategies: estimation importance of the parameters and knowledge distillation. The parameters importance estimation strategy aims to detect important weights by employing Elastic Weight Consolidations [78, 79], Memory Aware

Synapses [80], or Deep Model Consolidation [81]. The second strategy is to use the knowledge distillation function, which enforces preserving previously consolidated knowledge while learning a new task [82, 83].

To best of the author's knowledge, there is no study for hand-eye calibration using Continual Learning. However, Wang *et al.* [84] developed a Continual Learning-based visual localisation (camera pose estimation) approach. In [84], they trained their network sequentially with a novel buffer system. They combined reservoir and class-balance [85] buffer methods to overcome the catastrophic forgetting problem. The reservoir method enables them to sample previous data with uniform distribution, while the class-balance method ensures that selected samples represent all scenes.

By inspiring this study [84], it is possible to design a deep learning-based hand-eye calibration approach using Continual Learning, which extends its learned calibration space using new observations over time. The buffer system could store previous hand-eye calibration parameters and corresponding data, and the network could be trained sequentially outside of the learned space while also revisiting and fine-tuning previously observed space.

2.9 Discussion

The literature review summarises and compares the existing hand-eye calibration approaches. Section 2.2 presents the adopted hardware, software and technology. A brief background is provided to understand the hand-eye calibration problem in Section 2.3. Hand-eye calibration consists of three main components: camera, robot, and hand-eye calibration approaches.

Section 2.4 details the current camera calibration approaches. The marker-based approaches, such as checkerboard, chessboard, and QR-codes, have been widely adopted at the early stage of the research. Checkerboard and chessboard calibration targets are not suitable for flexible camera calibration scenarios because they must be attached to the robot end-effector or placed in world space for data collection steps. After data collection, they must be removed, and when the camera pose changes, all these steps must be repeated from scratch. Although QR-code-based calibration targets are more flexible than the classic calibration targets, they may be occluded with the robot joint while robotic manipulation is performed. Moreover, QR-code-based approaches is too sensitive to their size and highly affected by lighting conditions.

Self-calibration-based approaches have been developed over the last decades, and they have huge potential to develop flexible and autonomous calibration systems. However, their current success is not comparable with the marker-based approaches. These methods use the environment's feature points or natural landmarks to estimate the camera's intrinsic parameters. However, they require careful data collection, pre-processing, and post-processing steps to ensure accurate results. These methods are more flexible than marker-based approaches but are highly dependent on the environment and the camera's field of view. They also require more computational power, and the accuracy highly depends on the feature detection algorithms. Deep learning-based approaches have been recently introduced to overcome these limitations and have shown promising results.

Section 2.5 provides an overview of the literature on robot calibration, an important aspect often assumed to be known in hand-eye calibration research. However, accurate robot calibration is crucial for the success of hand-eye calibration approaches that rely on the **AX=XB** and **AX=YB** formulations. Deviations in robot calibration can introduce significant errors into the hand-eye calibration results, making it essential to address this issue. Deep learning-based approaches have the potential to mitigate such errors by leveraging large amounts of data and learning from them, thereby reducing the dependence on precise robot calibration knowledge

Section 2.6 provides a comprehensive overview of the evolution of classic hand-eye calibration literature and presents an insightful analysis of the nature of the problem. The **AX=XB** formulation established the relationship between the end-effector and camera configurations and was the first and most straightforward method proposed to solve the hand-eye calibration problem. Separation-based and simultaneous solution approaches have been proposed to obtain the hand-eye calibration parameters using this formulation. However, separation-based approaches tend to transfer errors from the orientation component to the translation component, while simultaneous solution approaches require good initialisation to converge to the optimal solution. Moreover, **AX=XB** formulation is sensitive to the motion of the calibration target or the robot's end-effector, which limits its flexibility in handling camera pose changes. To address this limitation, **AX=YB** formulation-based approaches have been developed, which do not depend on the calibration target or robot motion. However, these approaches also have separation and simultaneous solution methods, which have their drawbacks, similar to **AX=XB** formulation-based approaches. Additionally, reprojection error-based approaches have been proposed to minimise errors resulting from camera calibration. Despite their advantages, classic hand-eye calibration approaches lack flexibility in dealing with camera pose changes.

In conclusion, each formulation has its own advantages and limitations. Researchers have proposed various approaches to address the challenges associated with hand-eye calibration. However, the lack of flexibility in handling camera pose changes remains a significant limitation of classic hand-eye calibration approaches.

Section 2.7 provides a detailed description of deep learning-based hand-eye calibration approaches. Early attempts of using deep learning involved using it as a supplementary tool to extract, detect, and track key points. However, Valassakis *et al.* [19] introduced an end-to-end deep learning-based hand-eye calibration approach, which demonstrated that deep learning can serve as a hand-eye calibration model.

Moving on, Section 2.8 provides an overview of the Continual Learning literature, which has a high potential to expand the learned calibration space via new observations while preserving previously acquired knowledge. Although no study has investigated hand-eye calibration problems using a Continual Learning approach, the lifelong learning paradigm of Continual Learning can aid in developing more flexible and autonomous hand-eye calibration systems.

Chapter 3

Deep Learning-Based Hand-eye Calibration Approach

3.1 Introduction

This chapter introduces a novel approach to solve the hand-eye calibration problem using deep learning. The approach is designed to enable a hand-eye calibration system that can recalibrate itself without the need for data recollection. To achieve this, the hand-eye calibration problem was divided into three sub-tasks. In the first sub-task, deep learning was used to estimate the hand-eye calibration parameters by processing known transformations of a single reference point selected on the robot end-effector with respect to both the robot and camera base, assuming that the camera and robot calibration were already known. In the second sub-task, deep learning was used for the camera calibration, using the single reference point pose with respect to the robot base and corresponding RGB and depth images in n different robot end-effector configurations and m different camera configurations, with the assumption that only the robot calibration is known. Notably, this approach does not require a calibration target. Finally, deep learning was utilised to estimate the hand-eye calibration parameters by tracking a single reference point using a 3D vision system, utilising RGB and depth images and the pose of the reference point over different camera and robot end-effector configurations as inputs.

The effectiveness of the approaches mentioned above were evaluated through experiments conducted in two robotic testbeds (a Universal Robot and a Rethink Baxter), as well as a simulated environment, as illustrated in Figure 3.1. In the simulation environment, a stereo vision camera was placed in 108 different locations, and for each camera position, 50 different end-effector configurations were executed to collect samples. In the real-world experiments, 24 and 19 locations were chosen to cover the camera space, while 100 and 85 robot end-effector configurations were executed to cover the robot space for the Universal Robot 3 and the Rethink Baxter Robot, respectively.

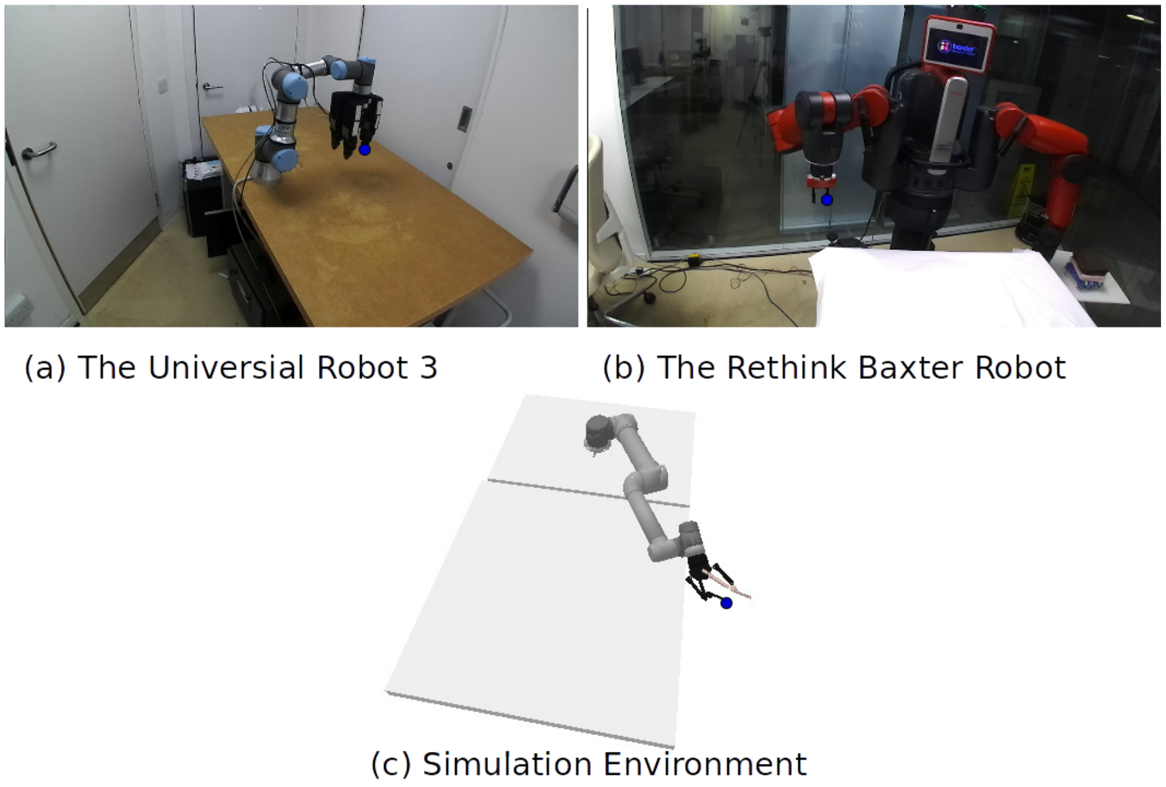

(a) The Universial Robot 3
(b) The Rethink Baxter Robot
(c) Simulation Environment

Figure 3.1: (a) and (b) depict the two real-world robotic testbeds and (c) present the simulated environment used in this study, as observed from the camera view.

3.2 Motivation and Objectives

Early approaches that have successfully solved the hand-eye calibration problem relied on offline data collection to acquire the required poses of the camera and robot's end-effector to perform hand-eye calibration. The main limitation of offline hand-eye calibration approaches is that they are not flexible against camera pose changes with respect to the robot base after obtaining the geometric transformation model between the camera with respect to the robot's world coordinate frame. To overcome this problem, online hand-eye calibration approaches [86] [18] have been developed over the last decade, the most recent leveraging deep learning. Although current hand-eye calibration methods have significant success, a robust and flexible method that recalibrates the system without data recollection and is easily deployable to other robotic environments has not been proposed as described in Chapter 3.

This chapter hypothesises that *a hand-eye calibration approach, which recalibrates external camera pose by observing a single reference point on the robot end-effector through a 3D vision system, enables a robot to perform robotic manipulation and grasping tasks by being adaptable and robust to environmental changes. Moreover, it has the same success on a real robotic manipulation task while reducing the computational complexity and increasing the repeatability*

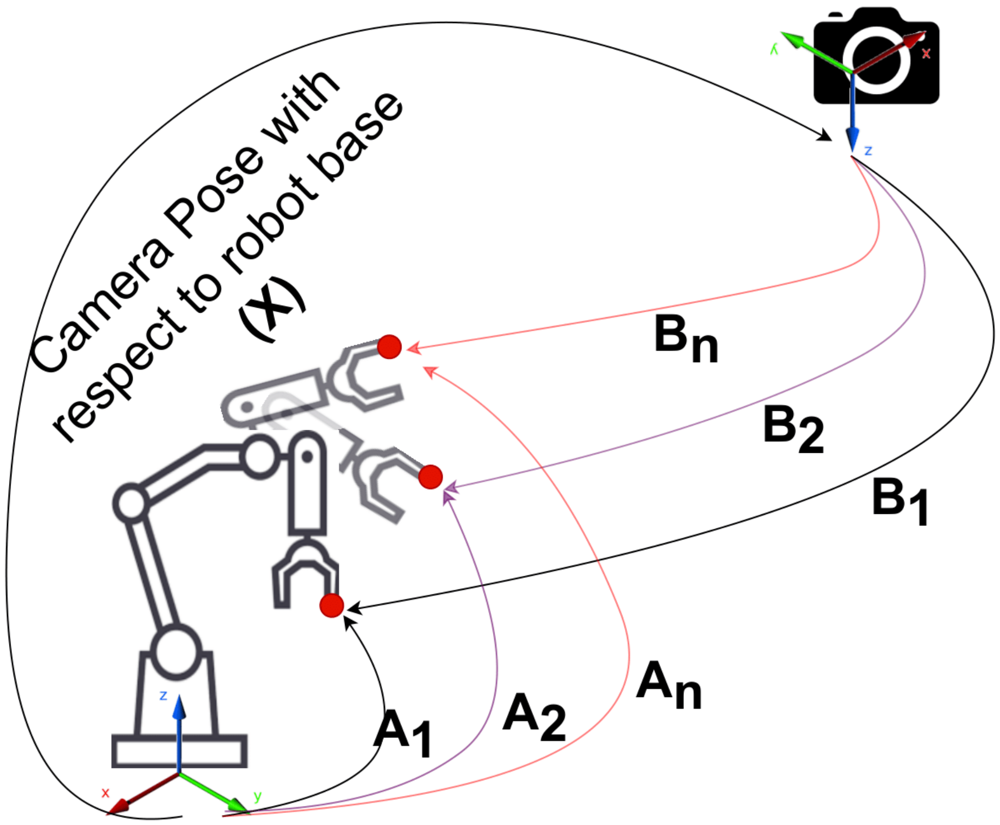


Figure 3.2: The geometric representation of the proposed hand-eye calibration approach. The homogeneous transformation between the robot base and the camera is represented by $\mathbf{X}$. $\mathbf{A_{1....n}}$ represents the pose of the reference point with respect to the robot base, while $\mathbf{B_{1....n}}$ are the camera pose with respect to the reference point.

compared to classical approaches.

The contributions of this chapter are threefold:

1. It is demonstrated that it is possible to estimate the camera pose by tracking a single reference point defined by the robot's kinematic chain and automatically detecting this reference point using a 3D vision system.

2. A deep learning-based hand-eye calibration approach with single RGB and depth images has competitive accuracy for position and orientation errors than classical hand-eye calibration methods and the state-of-the-art.

3. It is shown that it is possible to carry out hand-eye calibration without explicit camera extrinsic calibration (i.e. camera pose).

3.3 Methodology

Figure 3.2 shows the hand-eye calibration formulation in this paper. **A**, **B**, and **X** are the poses of the reference point with respect to the robot base, the pose of the camera with respect to the reference point, and the geometric transformation between the camera and the robot base, respectively. The poses of the end-effector or reference points and the corresponding poses of the camera in consecutive frames should be known for hand-eye calibration. Therefore, three primary research questions were posited to evaluate the feasibility of deep learning-based approaches as a solution for hand-eye calibration

Q1 In contrast to closed-form hand-eye approaches, is it possible to find the geometric transformation between the camera and the robot where camera and robot calibrations are known in advance by using a neural network?

Q2 Is it possible to find the camera's pose with respect to the reference point by observing the motions of this reference point via a 3D vision system and using deep learning as a calibration model?

Q3 Is it possible to carry out hand-eye calibration where camera calibration is not known by observing the motions of the reference point via a 3D vision system and employing a deep learning-based regression architecture as a model?

The experiments were conducted in a simulation environment for all research questions, while for **Q3**, experiments were also conducted in the real world due to the unavailability of ground truth data for **Q1** and **Q2**.

The objective of **Q1** was to assess the feasibility of a deep learning-based approach as a hand-eye calibration model. In this experiment, it was assumed that the robot and camera calibration were known, and a deep learning-based regression architecture was designed to take the known transformations of the reference point (**A**) and camera pose (**B**) as inputs and output the unknown hand-eye transformation (**X**).

The applicability of a deep learning-based regression model for camera calibration was explored in **Q2**. In this experiment, the extrinsic camera parameters were unknown, unlike in **Q1**. These parameters were estimated with respect to the reference point by tracking the reference point using a 3D vision system. Additionally, the regression model used in **Q1** was extended to input the pose (**A**) of the reference point with respect to the robot's world frame and the corresponding RGB and depth images. This architecture outputted the pose (**B**) of the camera with respect to the robot's world reference frame.

Finally, **Q3** aimed to carry out hand-eye calibration directly using a deep learning-based regression architecture. In this configuration, the robot calibration and the camera's intrinsic parameters were assumed to be known, while the camera pose and the geometric transformation between the camera and the robot base were unknown. A deep learning-based regression architecture was trained using poses of the reference point (**A**) in *n* frames and corresponding RGB and depth images as inputs. The architecture estimated the unknown geometric transformation (**X**). The experimental setup and network architecture used for **Q3** are illustrated in Figure 3.1.

3.4 Simulation Experiments

3.4.1 Data Generation

A virtual Universal Robot 10 (UR10) was utilised in the PyBullet simulation environment, along with a stereo vision camera. The data for the experiments was generated by placing the camera over 108 different locations (depicted in Figure 3.3) in the PyBullet environment, and the robot end-effector was moved to 50 different configurations for each camera pose. For the first research question (**Q1**), the reference point poses with respect to the robot (**A**) and the camera base (**B**), and the homogeneous transformation between the robot base and the camera (**X**) were recorded for each end-effector configuration. For the second research question (**Q2**), RGB and depth images were captured along with the reference point poses with respect to the robot base (**A**), and the homogeneous transformation (**B**) between the camera and the reference point (camera's extrinsic parameters with respect to the reference point) for each end-effector configuration. Finally, for the third research question (**Q3**), RGB and depth images were captured for each end-effector configuration along with the geometric transformations of **A** and **X**.

The transformation matrices **X**, **A**, and **B** consist of a seven-element array where the first three elements represent the position, and the last four elements are the orientation parameters (unit quaternion) of the transformation. The RGB images are $16-bit$ with a resolution of $1024x1024$ pixels, while the depth images are also $16-bit$ with a resolution of $1024x1024$ pixels. Table 3.1 summarises the generated data in the simulation environment for each research question.

Table 3.1: Generated data in the simulation environment

Simulation Environment UR 10		
Research Question	**Q1**	**Q2,Q3**
Camera Configurations	108	108
End-effector Configurations	50 for each cam.	50 for each cam.
Data	5300 A,B and X (ground-truth)	5300 RGB and depth images, A, B, and X (ground-truth)

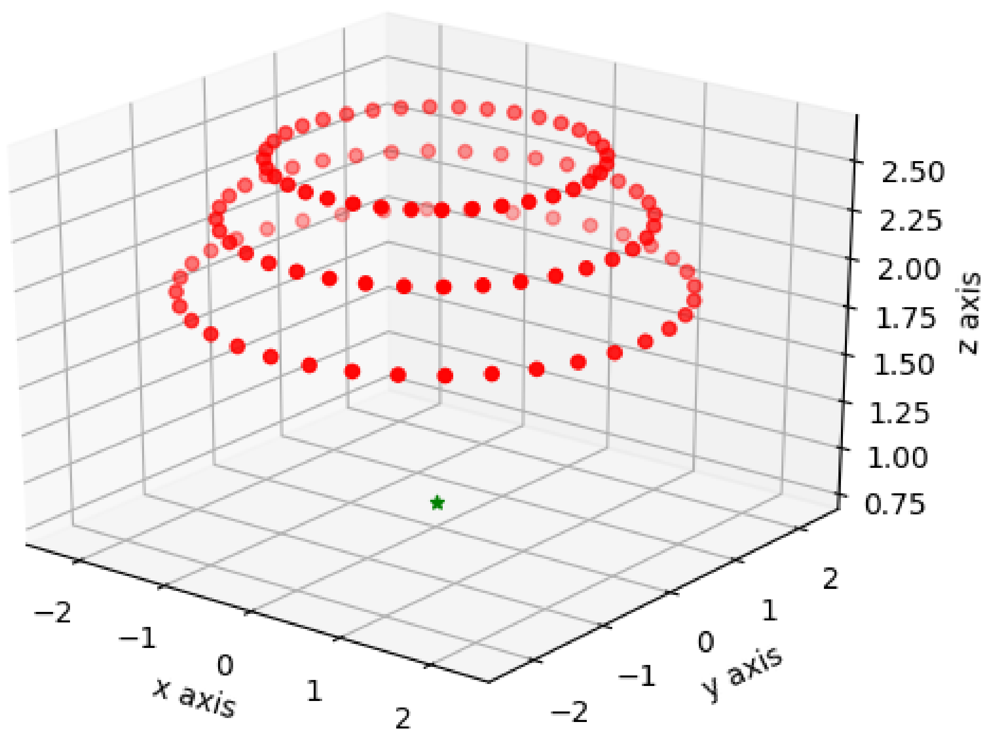


Figure 3.3: This figure shows the 108 camera positions (red dots) and the robot centre (green star) in the simulation environment.

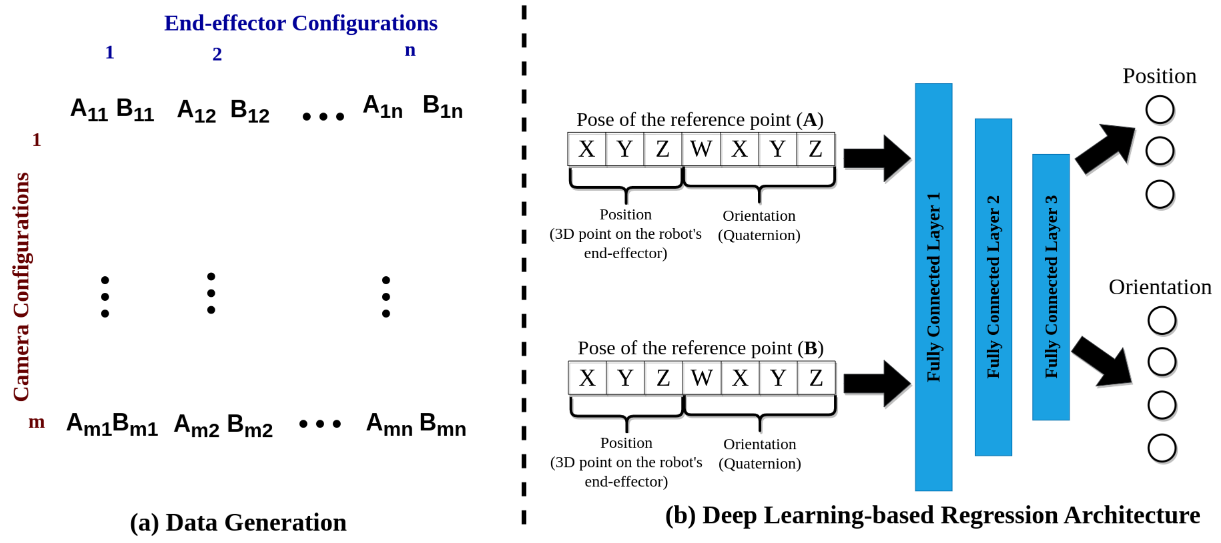

End-effector Configurations
1
2
n
Camera Configurations
1
m
A11 B11 A12 B12 ... A1n B1n
Am1Bm1 Am2 Bm2 ... Amn Bmn
(a) Data Generation
Pose of the reference point (A)
X Y Z W X Y Z
Position (3D point on the robot's end-effector)
Orientation (Quaternion)
Pose of the reference point (B)
Fully Connected Layer 1
Fully Connected Layer 2
Fully Connected Layer 3
Position
Orientation
(b) Deep Learning-based Regression Architecture

Figure 3.4: This figure depicts the research design for addressing **Q1**. Subfigure (a) shows data collection with camera positions and end-effector configurations. Subfigure (b) presents the distinct neural network for position and orientation, providing outputs for unknown hand-eye transformations (**X**).

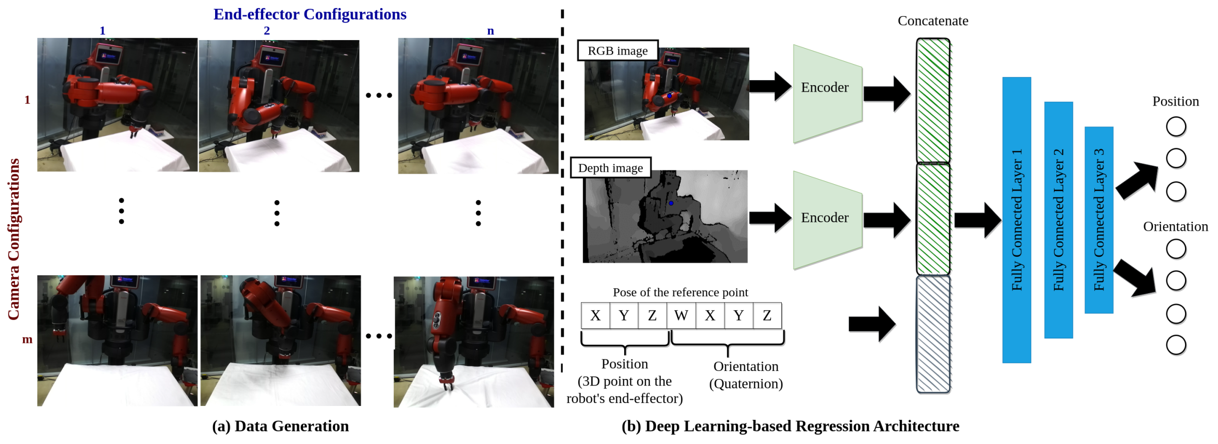


Figure 3.5: This figure outlines the research design for **Q2** and **Q3**. In subfigure (a), data collection involves diverse camera positions and end-effector setups. Subfigure (b) showcases the deep learning architecture using U-Net encoders for RGB and depth data. Concatenated with known **A** transformation, the network separately focuses on position and orientation. Trained network outputs **B** (**Q2**) or **X** (**Q3**).

3.4.2 Network Architectures

A deep learning-based regression architecture (as shown in Figure 3.4(b)) was developed for **Q1** to estimate the geometric transformation between the robot base and the camera. The architecture was designed with separate training for the position and orientation components of the transformation due to their differing solution spaces, i.e., translation is in millimeter space while orientation is in radian space. The network consists of three fully connected layers with a Rectified Linear Units (ReLU) [87] activation function and is trained using meters and radians for position and orientation, respectively. The architecture starts with 512 neurons, chosen through empirical experimentation, and systematically halves for each layer except the last layer, which is three and four for position and orientation, respectively. The architecture was trained for 50 epochs using Adam optimiser [88], with a learning rate of $1e-3$ and a batch size of 100.

For **Q2** and **Q3**, a deep learning-based regression architecture was also developed. This architecture takes as inputs RGB and depth images of dimensions *256x256*, as well as the poses of the reference point with respect to the robot base. The architecture is inspired by the U-Net architecture [89], with the encoder part tailored to extract features from RGB and depth images separately (as shown in 3.5(b)). The encoders are composed of two consecutive Conv2D-3x3 layers, each with a stride of 1 and no padding, succeeded by a subsequent Maxpool-2x2 layer for dimensionality reduction. The output is then subjected to the ReLU activation function. The encoded features for RGB and depth images are then concatenated with the known transformation **A**, and a fully connected network with three hidden layers is employed to estimate the translation or orientation component of the unknown transformation **B** and **X** for **Q2** and **Q3**, respectively. The network is trained separately for position and orientation for both **Q2** and **Q3** due to differences in their solution spaces. The network is trained for 20 epochs with 4 mini-batches and a learning rate of $1e-3$ using Adam optimiser [88].

The direct representation of the camera's position with respect to the robot base in 3D Cartesian space is obtained from the last three neurons in the deep learning-based regression architectures for position estimation. However, for orientation estimation, the last ten neurons are converted to a 4×4 symmetric matrix **S(θ)** (as detailed in Equation 3.1), which is then passed through a Quadratically Constrained Quadratic Program-based model (presented in [33]) to produce a unit quaternion representing the camera's orientation in 3D Cartesian space. The double cover problem between the quaternion and 3D Cartesian Space (detailed in chapter 2.3.1) was resolved by handling this representation.

$$S(\theta) = \begin{bmatrix} \theta 1 & \theta 2 & \theta 3 & \theta 4 \\ \theta 2 & \theta 5 & \theta 6 & \theta 7 \\ \theta 3 & \theta 6 & \theta 8 & \theta 9 \\ \theta 4 & \theta 7 & \theta 9 & \theta 10 \end{bmatrix} \tag{3.1}$$

3.4.3 Loss Function and Metric

The models for position estimation were trained using the Mean Square Error (MSE) metric. The accuracy of the solution in the original units was evaluated using the Root Mean Square Error (RMSE). The RMSE values for position estimation were converted to millimetres for interpretation. For orientation estimation, the chordal loss function (equation 3.3) described in [33] was utilised.

$$d_{chord}(R, R_{gt}) = ||R_{gt} - R||_F \tag{3.2}$$

To interpret the orientation parameters, they were converted to degrees using equation 3.4.

$$L_{chord}(R, R_{gt}) = d_{chord}(R, R_{gt})^2 \tag{3.3}$$

In equation 3.2, R and R_{gt} represent estimated and ground-truth unit quaternions, respectively, and F is the Frobenius norm.

$$angle = 4 * \sin(0.5 * min(||R_{gt} - R||_F, ||R_{gt} + R||_F))^{-1} * 180/\pi \tag{3.4}$$

3.4.4 Experimental Results

The networks were trained three times for each research question using the parameters mentioned in Section 3.4.2. 84 and 24 camera poses were used as training and testing sets. The mean and standard deviation of all research questions' position and orientation results are presented in Table 3.2. A direct comparison of **Q2** with **Q1** and **Q3** is not possible because **Q2**

Table 3.2: Experimental Results in Simulation

	Position Error (mm)		**Orientation Error (degree)**	
	Mean	**1 Std**	**Mean**	**1 Std**
Q1	0.415	0.016	0.036	0.003
Q2	5.44	0.343	0.563	0.111
Q3	1.69	0.11	1.91	0.28

enables acquiring the camera's extrinsic parameters only, while **Q1** and **Q3** are hand-eye calibration models.

The positional and orientational errors for **Q1** were 0.42 mm and 0.036 degree, respectively. These results indicate that a deep learning-based regression architecture could serve as a hand-eye calibration model in situations where the camera and robot calibration are already known. As for **Q3**, the positional and orientational errors were 1.69 mm and 1.91 degree, respectively, which is slightly worse than **Q1**. However, the model can implicitly solve unknown camera calibration, making the hand-eye calibration system more flexible and deployable in real-world scenarios.

Regarding **Q2**, the position error was 5.44 mm (standard deviation, 0.343 degrees), while the orientation error was 0.56 degrees (standard deviation 0.112 degrees). These results are competitive, but using multiple reference points, as in our approach, could potentially improve the success rate of explicit camera pose estimation. However, the estimation of camera extrinsic parameters falls outside the scope of this thesis, as we only considered the hand-eye calibration problem. Further research could investigate the impact of multiple reference points on camera pose estimation to provide more evidence supporting **Q2**. It should be noted that camera extrinsic parameters in **Q2** are with respect to the reference point, unlike **Q3**, where the camera extrinsic's parameters are with respect to the robot base (hand-eye calibration).

It should be noted that the developed deep learning-based regression architecture is not closed-form optimisation and it is an iterative optimisation approach. Hence, even in the simulation environment, it is not expected that it directly produces precisely zero error for the testing set.

3.4.5 Ablation Study

Table 3.3 presents the translational and orientational errors of a fusion model used for simultaneous estimation using mean squared error (MSE). In contrast to the previous model, which estimates these components separately, this model estimates them together. However, the results indicate that the errors in the fusion model are worse than those in the separate solution approach, which are presented in Table 3.2. Specifically, the orientation error in the fusion model is 4.416 degrees, which is twice of the separate model (1.79 degrees). This difference reveals that considering translation and orientation parameters simultaneously is challenging for deep learning models because the solution spaces for translation and orientation are different, as mentioned in Section 3.4.2.

It is crucial to carefully consider and systematically design deep learning models, especially when tackling complex tasks such as hand-eye calibration. While the simultaneous estimation of the translation and orientation components of calibration parameters might appear straightforward, it is essential to understand each parameter's inherent constraints and solution spaces and choose the most appropriate approach based on the specific task requirements. This ablation study found that separately estimating translation and orientation parameters produced better results than the fusion model, highlighting the importance of thorough analysis and deliberate selection of deep learning model architectures for optimal performance.

Table 3.3: Mean and standard deviation of the fusion approach, which estimates the translation and orientation parameters simultaneously, where training and testing are 87 and 21 camera poses, respectively.

Fusion		
Components	**Mean**	**STD**
Translation (mm)	2.02	0.126
Orientation (degree)	4.416	0.685

Figure 3.6 illustrates the translation error for unseen camera poses as a function of the training and testing dataset sizes. The horizontal axis (x-axis) denotes the size of the training dataset, while the testing dataset size equal to the difference between 108 (total camera poses) and the training dataset size. The results indicate that as the size of the training dataset increases, the translation error decreases. The optimal training size for representing 108 camera configurations is 45, which has an error of approximately 3.65 mm. This error is nearly the same as that of the training size 87, which has an error of approximately 1.69 mm.

After using around 85% of the training dataset (about 80 camera poses), Figure 3.6 shows a clear drop. This drop indicates that the balance between training and testing poses is not even, with fewer camera poses used for testing. While the decrease in testing error might seem positive, it is important to note that this might not accurately predict how the model will perform with new, unseen data. The uneven distribution of testing poses could introduce bias and affect how well the model can handle different situations. So, while the testing error goes down, it is essential to consider the imbalance in the testing data distribution.

Figure 3.7 compares the testing orientation error across different training sample sizes for two representations: pure quaternion and 10D quaternion. The results indicate that the 10D representation outperforms the pure quaternion representation, with an error that is 2.5 times smaller for each training sample size. This performance gap can be attributed to the double cover problem, which arises from the fact that pure quaternion and 3D Cartesian space are not one-to-one mappings, and this problem is resolved in the 10D representation. More information on this problem can be found in Chapter 2.3.1.

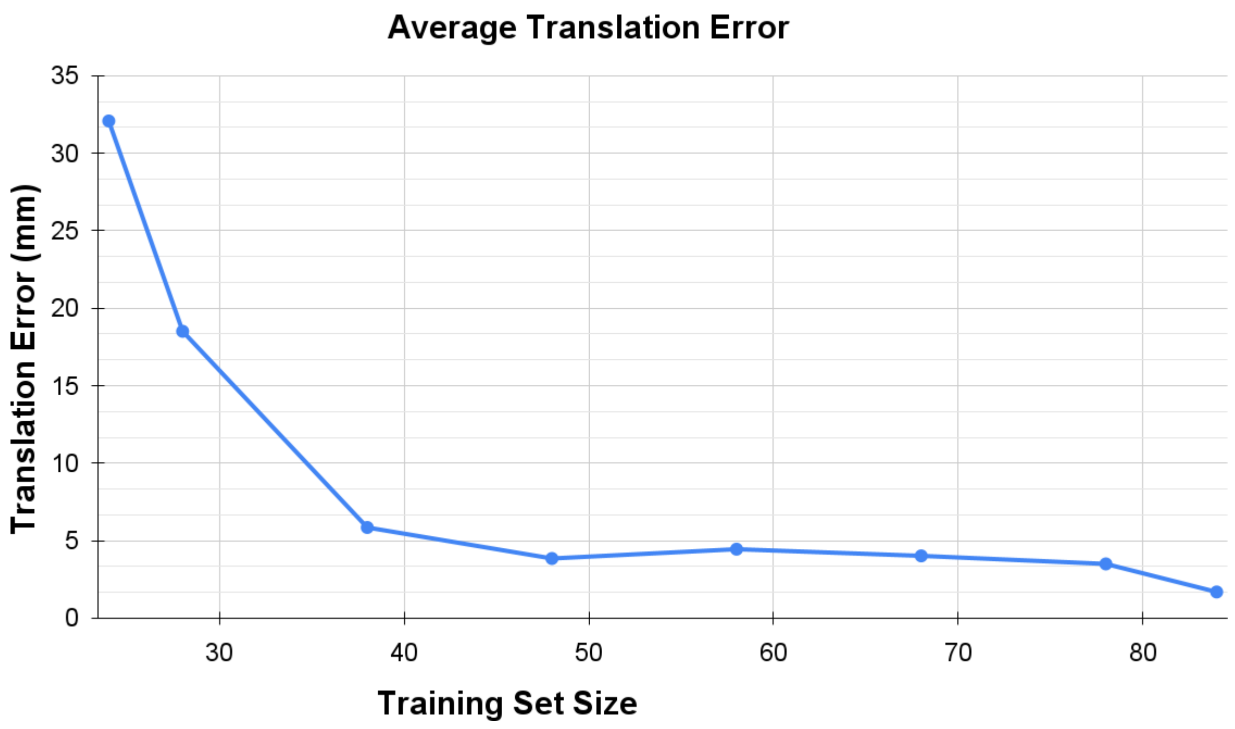


Figure 3.6: This figure displays the testing error of the translational component as a function of the training dataset size.

3.5 Real-world Experiments

3.5.1 Data Generation

Universal Robot (UR3) robotic testbed The UR3 robot was equipped with a Shadow Modular Grasper [90] featuring nine degrees of freedom (three fingers and three joints for each finger), of which the last link of one of these fingers was selected as the reference point. The UR3's repeatability was reported to be 0.1 mm. Next, the Stereolab's ZED camera [28] was placed in 24 different locations around the robot's workspace, and the Robot Operating System (ROS) [23] was employed to control the robot and connect the camera. Tsai hand-eye calibration method [8] was used for each camera pose to obtain a baseline for the geometric transformation between the robot base and the camera. Tsai hand-eye calibration was repeated five times for each camera configuration, and the best calibration result was used as the input to the regression neural network. The rationale for repeating the calibration five times was due to the dependency of Tsai's calibration on the quality of pose sequences. Table 3.4 shows the average of the best of these five hand-eye calibration attempts for the UR3.

Subsequently, the robot was allowed to plan for 100 different robot end-effector poses for each camera location, and RGB and depth images were captured using the camera while recording the poses of the reference point with respect to the robot base using the ROS Transform Library (TF) [91]. In total, 2400 RGB and depth images of *1920×1080* image resolution were captured.

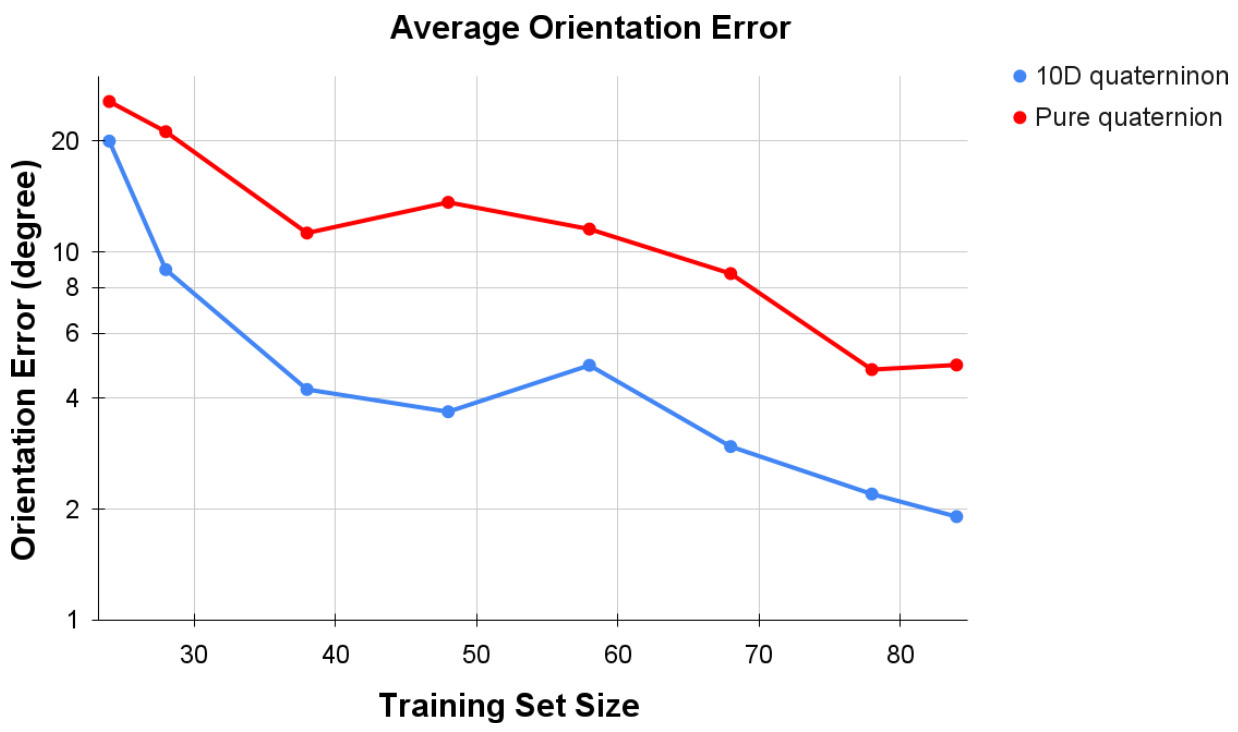


Figure 3.7: This figure shows the performance of the pure and 10D quaternion representation according to the training size.

Table 3.4: Average mean and std of reprojection error (pixel) for m camera configuration by employing Tsai hand-eye calibration method (ground-truth) on UR3 testbed

UR3 (pixels)	
Mean	**1 Std**
1.04	0.42

The Rethink Baxter robotic testbed The Baxter robot with electric parallel grippers was used as a second real robotic environment. Baxter has reported repeatability in the left and right arm of 2.9 mm and 3.3 mm, respectively [92]. As it can be observed, the UR3 has better repeatability than Baxter. This repeatability difference allows us to quantitatively measure whether our approach can accommodate wear and tear, as Baxter's compliant actuation system inherently introduces errors in the robot's end-effector positioning.

The Seterolab's ZED camera was placed in 19 configurations, and Tsai hand-eye calibration was employed to get a baseline for each camera pose. Tsai's hand-eye calibration was carried out five times for each camera configuration, and the best calibration was used as the input to the neural network (detailed in Table 3.5). The right finger of the right gripper in Baxter was selected as the reference point for **Q3**. The ROS and TF library were employed to control the robot and get the reference point transformations. For each camera configuration, the end-effector on the right arm of the Baxter robot was allowed to move to 85 different poses for each camera location. RGB and depth images were captured, and the geometric transformation between the robot base and the reference point was recorded. In total, 1615 RGB and depth images were collected with

Table 3.5: Average mean and std of reprojection error (pixels) for *m* camera configuration by employing Tsai hand-eye calibration method (ground-truth) on the Baxter testbed

Baxter (pixels)	
Mean	**1 Std**
2.41	0.97

an image resolution of *1920×1080* pixels.

3.5.2 Network Architecture

A deep learning-based regression architecture is developed for **Q3**, similar to the architecture designed for the simulation environment in Section 3.4.2 (Fig.3.5(b)). However, to handle real-world complexity, the number of double Conv2D-3x3 layers was increased from two to five. While the shallow network proved to be highly effective in the simulation environment, the deep network was deemed more suitable for real-world experiments involving various noise sources, such as illumination and image acquisition. The network was trained for 20 epochs using the Adam optimiser with four mini-batch and a learning rate of $1e-4$ for position and orientation parameters estimation. To account for variation from the optimiser, the network was trained three times with the same seed, and all training repetitions were recorded in the results.

3.5.3 Real-world Results

Table 3.6 summarises the position and orientation errors based on ground-truth coming from Tsai's hand-eye calibration approach [8] for **Q3** in the Baxter and UR3 robotic testbeds. The positional and orientational parameters derived from Tsai's hand-eye calibration approach have been treated as the reference standard. Following multiple iterations of data collection, this methodology demonstrates commendable outcomes within specific camera and robot base configurations. Our deep learning-based regression architecture demonstrated 2.07 mm and 5.8 degrees of error for translation and orientation, respectively, in the test set for the UR3 robotic environment. In the case of Baxter, our approach yielded 4.54 mm and 9.2 degrees of error in the test set for translation and orientation, respectively. As anticipated, the difference between the UR3 and Baxter calibration results was attributed to Baxter's inherent inaccuracy. Nevertheless, our approach doubled the error but remained below similar approaches presented in the literature (refer to Table 3.7).

Table 3.6: Experimental Results for Baxter and UR3 robots

	Position Error (mm)		**Orientation Error (degree)**	
	Mean	**1 Std**	**Mean**	**1 Std**
Baxter	4.543	0.081	9.2	1.35
UR3	2.07	0.331	5.8	1.02

The indirect error function presented and used in [8, 19] is employed to make a fair comparison with the literature because the results are highly dependent on the experimental setup, the robot and the camera. However, the use of this error function shows the repeatability of a hand-eye calibration approach that is independent of the robotic environment. Specifically, the indirect error function is given in equation 3.5.

$$\varepsilon_{std} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{n}\sum_{i=1}^{n}\|\bar{x}-\mu\|_2^2\right)^{\frac{1}{2}} \tag{3.5}$$

where k, m, n, and μ are the neural network training repetitions (see Section 3.5.2), the total number of unseen camera configurations, the total number of end-effector configurations for each camera configuration, and the mean of predicted camera positions or orientations.

Table 3.7 presents a comparative analysis encompassing the baseline, the state-of-the-art approaches, and the proposed approach, focusing on the metric of repeatability error. The values depicted therein signify the mean deviation of positional error from the corresponding mean position within a given camera and robot pose configuration for a specified range of camera poses (m in total). The first two rows summarise the best Tsai's hand-eye calibrations in the experiments for the Baxter and UR3 robots. The repeatability error of the Tsai's approach [8] is 96 times higher than that of the DL-based approach. This significant difference can be attributed to two main factors. Firstly, the classic approach transfers the rotational error into the translational error, as it solves the rotation component (detailed chapter 2.6) first and then substitutes the solution of the rotation to solve the translation component. Secondly, the success rate of this approach is highly dependent on the selection of calibration target poses. Lee *et al.* [18] and [19] report an indirect error of 27.4 mm and 10.4 mm with respect to their sample mean, respectively. This represents a difference of 23.45 mm and 6.45 mm with respect to the approach in the UR3. It should be noted that Lee *et al.* [18] and [19] do not use a reference point that is aligned to the robot's kinematic chain but instead relies on the captured images to infer the hand-eye calibration parameters. The approach explicitly defines a reference point from which a neural network can create an embedding that considers the robot's end-effector and the kinematic chain (i.e. robot calibration in chapter 2.5).

The achieved reduction in repeatability error, as observed through the proposed DL-based approach, implies a commendable enhancement in the calibration accuracy. While this enhancement aligns with the trend observed in similar DL-based calibration approaches, there remains the need to delve into the complexities inherent to these methodologies, including their potential limitations when applied in dynamic environments. The substantial repeatability error observed in Tsai's approach across both robotic testbeds implies susceptibility to yielding imprecise calibration parameters. This underscores that multiple iterations are essential to acquire valid hand-eye calibration parameters even within a fixed camera and robot base configuration. Consequently, this inherent characteristic renders Tsai's approach ill-suited for dynamic hand-eye calibration scenarios, necessitating the continual or iterative updating of camera poses to accommodate evolving conditions.

Table 3.7: Comparison with the state-of-the-art

Method	**Position**($\varepsilon_{\mathbf{std(mm)}}$)
Tsai-Baxter	(420.13±175.55)
Tsai-UR3	(382.8±150.98)
Keypoint+PnP [18]	(27.4±4.7)
Direct Regression [19]	(10.4±4.0)
Our approach-Baxter	(9.66±1.58)
Our approach-UR3	(3.95±1.25)

3.6 Conclusion

This chapter has verified the following hypothesis:

A hand-eye calibration approach, which recalibrates external camera pose by observing a single reference point on the robot end-effector through a 3D vision system, enables a robot to perform robotic manipulation and grasping tasks by being adaptable and robust to environmental changes. Moreover, it has the same success on a real robotic manipulation task while reducing the computational complexity and increasing the repeatability compared to classical approaches.

A deep learning-based hand-eye calibration approach was described in this chapter, which enabled the estimation of the unknown geometric transformation between the robot base and an external camera without requiring data recollection after network training. The motion of a single point on the robot's end-effector was tracked through a 3D vision system during experiments. The chapter also proposed three research questions (as detailed in Section 5.3). The first research question (**Q1**) demonstrated the use of a deep learning-based regression model instead of mathematical models (e.g. [8]) for hand-eye calibration. In this approach, it was assumed that robot and camera calibration were known, and the unknown transformation was directly estimated via the deep learning-based regression model. The second research question (**Q2**) investigated a tailored regression approach with the addition of two encoders for RGB and depth images to find the camera's extrinsic parameters when camera calibration is unknown. Finally, the third research question (**Q3**) employed the network architecture to estimate the unknown transformation between the camera and the robot base. All these research questions were tested in a simulation environment, and the experimental results demonstrated that a deep learning model can be used for hand-eye calibration. To further validate the approach, experiments for **Q3** were performed on two real robotic environments (a Universal Robot 3 and the Rethink Baxter robot), which yielded an overall position error of 2.07 and 4.54 mm and orientation error of 5.8 and 9.2 degrees, respectively. These results showed the robustness of the approach to camera pose changes while having only a repeatability error of 3.95% and 9.6% for unseen camera poses that were not used during network training.

An autonomous and flexible hand-eye calibration system is crucial in robotics applications where robots need to adapt continuously to new environments and tasks without requiring significant manual intervention. Such systems can benefit various industries, such as manufacturing, logistics, and healthcare, by enabling robots to learn from vast amounts of data and improve their accuracy and precision. This is particularly useful in dynamic and changing environments where robots must perform various tasks on different production lines or adapt to changes in layouts [93]. By continuously calibrating their hand-eye system, robots can accurately track their position and orientation and perform tasks with incredible speed and efficiency while reducing the risk of errors or accidents.

Chapter 4

Pick-and-Place Pipeline

4.1 Introduction

This chapter examines the performance of classic and deep learning-based hand-eye calibration approaches in a real robotic manipulation task, specifically pick-and-place. Three evaluation criteria were proposed to assess the task's functionality: computational complexity, repeatability score, and success rate in performing the task. Tsai's [8] method, known for its high success rate and extensive usage in the literature, was employed as the classic hand-eye calibration approach. Meanwhile, Bahadir *et al.*'s method [94] (developed in Chapter 3), which was trained using ground truth data from Tsai's method, served as the deep learning-based approach.

The camera was positioned in ten different locations for the experiments, and both Tsai's and the deep learning-based calibration approaches were carried out to obtain the camera's pose. A simple pick-and-place experiment was then designed using three objects of varying shapes and sizes (cube, box, and cup), as shown in Figure 4.1. Colour segmentation was utilised to determine the 6D pose of the objects accurately and quickly. This method rapidly identifies objects in an image when the colour codes are manually defined. Moveit Motion Planning [95] was utilised to pick and place the objects in the scene. To assess the success rate of both approaches, the pick-and-place task was divided into four main steps: reaching, touching, picking, and placing. The success rate of each of these steps and the computational costs of the hand-eye calibration using both methods were presented.

4.2 Motivation and Objectives

Several mathematical models have been proposed in the literature to represent the hand-eye calibration problem, including $AX = XB$, $AX = ZB$, projection error, and 3D reconstruction (detailed in Chapter 2). These models have shown successful results but are offline approaches, meaning their results are only valid for the current camera configuration. Therefore, they are not suitable for real-world robotic applications where the camera pose may change. Deep learning-

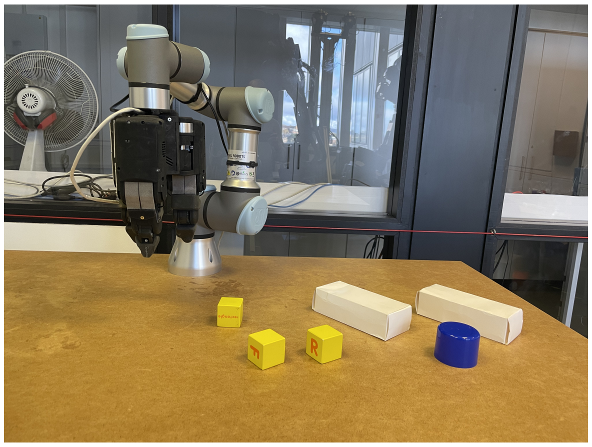


Figure 4.1: The figure shows the experimental set-up, which consists of a Universal Robot equipped with a three fingers gripper. The robot is mounted on the table, and three objects with unique colours have been placed in the robot's workspace.

based approaches have been proposed to address this limitation, eliminating the data collection process and re-training the model when the camera pose changes. However, these approaches require ground truth data and are trained in a supervised manner.

After collecting data and solving an optimisation problem or training the proposed deep learning architecture using a learning signal, an evaluation metric is necessary to judge the developed approach. However, there is no direct evaluation metric for hand-eye calibration because it is difficult to know precisely where the camera is with respect to the robot base in the real world. Hence, average rotation and translation error, projection error, 3D reconstruction error, and repeatability error [18] are the most preferred evaluation metrics in the hand-eye calibration literature [20]. However, the ideal scenario is performing a robotic manipulation task and judging the developed hand-eye calibration model by using the success rate of the accomplished task. Thus, this chapter considers an ideal evaluation scenario involving performing a real pick-and-place task with different objects to compare classic and deep learning-based hand-eye calibration approaches' accuracy, strengths, and limitations. This is the first study that considers a real robotic manipulation task to compare hand-eye calibration approaches in the literature.

Evaluating a hand-eye calibration method on a real robotic task is essential because it provides a more realistic and practical assessment of its performance. This is because the ultimate goal of hand-eye calibration is to enable the robot to perceive and manipulate objects in the real world accurately. Hence, assessing the success of a hand-eye calibration method in terms of its ability to perform a real-world robotic manipulation task is a better indication of its practical usefulness than simply evaluating it based on metrics such as average rotation and translation error, projection error, 3D reconstruction error, and repeatability error. Therefore, testing the success of hand-eye calibration on a real robotic task can help researchers and practitioners to make informed decisions about the practical applicability of the method.

This chapter examines the following research questions *to compare the classic and deep learning-based hand-eye calibration approach in terms of computational complexity, repeatability and successful performance on a real robotic manipulation task.*

- How does the performance of the deep learning-based HEC approach compare to other state-of-the-art methods in the field of hand-eye calibration, in terms of both accuracy and computational efficiency?
- What are the limitations of the deep learning-based HEC approach, and how can they be addressed to improve its practical application in various dynamic robotic environments?

4.3 Methodology

Figure 3.2 displays a real-world pick-and-place pipeline, which is employed to examine the success rate, strengths, limitations, and computational cost of both a classic and a deep learning-based hand-eye calibration approaches. This section is divided into two subsections. Subsection 4.3.1 outlines the process of determining hand-eye calibration parameters using [8] and [94] (chapter 3), while the object detection method adopted is presented in 4.3.2.

4.3.1 Hand-eye Calibration Methods

The classic hand-eye calibration approach selected for this study is [8], which is commonly used in the literature due to its simplicity and success under specific constraints, such as a specific distance between the camera and the robot base and changing orientation maximally between 0 to 89 degrees for each axis. For the deep learning-based hand-eye calibration approach, [94] (Chapter 3) was chosen as it is state-of-the-art, and [8] is used to provide ground truth for [94]. An overview of these approaches is presented in the remainder of this subsection.

Tsai's Hand-eye Calibration [8]

Figure 4.2 illustrates the hand-eye system used in this study. To calibrate the system, the calibration target attached to the robot's gripper is sampled at least eight times with different translations and orientations. The corresponding poses of the robot's end-effector and the camera's extrinsic parameters are collected for each configuration using OpenCV camera calibration [96]. Tsai's approach [8] is employed to solve the hand-eye calibration problem using the $AX = XB$ formulation (Equation (2.26) detailed in Chapter 2). These matrices are then decomposed into their orientation and translation components using Equations (2.28 and 2.27 detailed in Chapter 2). Finally, the orientation component of calibration parameters is solved using a linear optimisation method, and the translation component is calculated using the solution of the orientation.

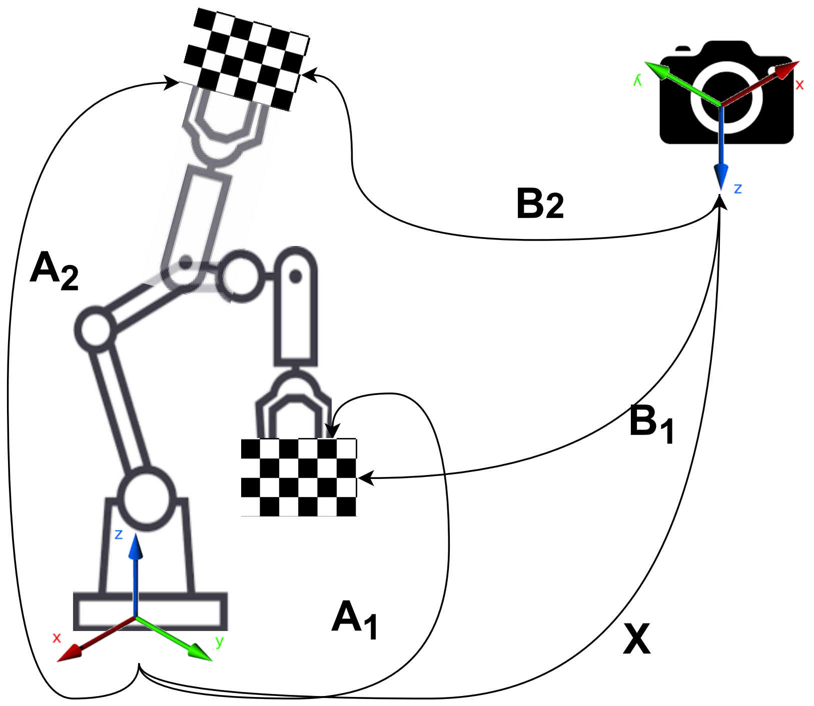


Figure 4.2: This figure shows the adopted hand-eye calibration system. A camera is placed in an external location, and a checkerboard is attached to the robot's end-effector. $\mathbf{A}_{1,2}$ describe the poses of the end-effector with respect to the robot base for different configurations, while $\mathbf{B}_{1,2}$ shows the poses of the camera with respect to the calibration target for the corresponding configurations. $\mathbf{X}$ presents the static transformation between the robot base and the camera.

Evaluation of the ascertained calibration parameters is conducted through the utilisation of the reprojection error. A solution is regarded as the ground truth for the given camera and robot base configuration when this error is confined to less than 2 pixels. Notably, this error value, measured in pixels, signifies successful hand-eye calibration outcomes suitable for applications within industrial robotics [97].

Deep Learning-based Hand-eye Calibration

The developed deep learning-based hand-eye calibration approach in Chapter 3 is used [94]. This approach employs an RGB image, a depth image, and the known transformation of a single point on the robot's end-effector to obtain the hand-eye calibration parameters. Unlike [8], which uses a calibration target and explicitly handles the camera extrinsic's parameters for each end-effector pose, [94] uses a single reference point, and the camera extrinsic's parameters are handled implicitly. It should be noted that there is no detection for this single reference point using images, as detailed in Chapter 3. Its pose with respect to the robot base is given as input to the network directly by using the robot kinematic chain.

To collect data, the robot's end-effector is sampled in n different locations to span the robot's workspace for each of the m camera configurations. Tsai's hand-eye calibration approach [8] is performed at least five times, and the best calibration in terms of reprojection error is used as the

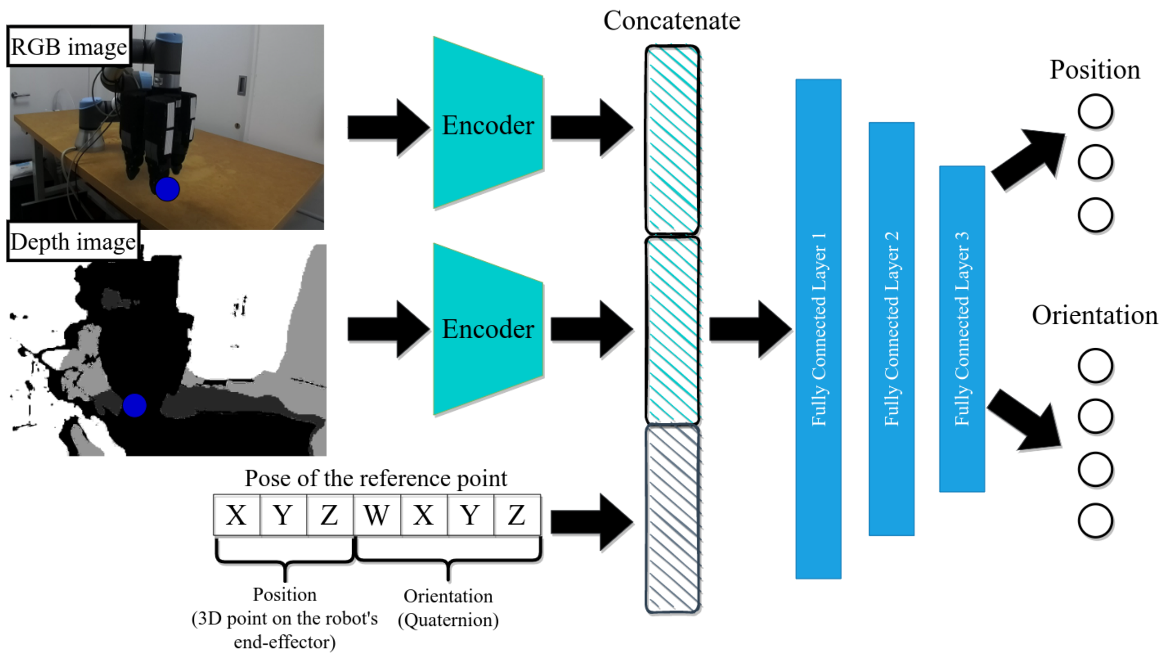


Figure 4.3: This figure presents the designed hand-eye calibration architecture, with two encoders extracting features from RGB and depth images separately. The extracted features are concatenated with the pose of the reference point (marked as the blue circle) with respect to the robot base and passed to three fully connected layers. The network's output is either translation (3D) or orientation (4D).

ground truth. The pose of the reference point with respect to the robot base and corresponding RGB and depth images are also collected.

After data collection, a deep learning regression architecture is designed, as shown in Figure 4.3. The network has two separate encoders for RGB and depth images and is trained separately to obtain translation and orientation parameters. Mean Square Error is the loss function for the translation parameters, represented in the 3D Euclidean space. A 10D quaternion representation [33] is employed for the orientation parameters, enabling the one-to-one mapping from the quaternion space to the 3D Cartesian space.

In the experiments conducted in this study, the deep learning-based regression architectures developed in Chapter 3 were used for pick-and-place operations. The trained models from Chapter 3 were used directly in this study without re-training or fine-tuning.

4.3.2 Object Detection

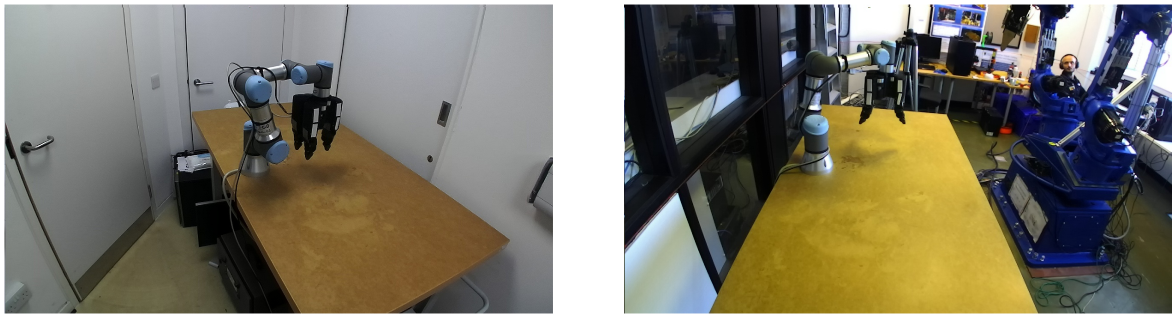
Colour segmentation is a prominent method for object detection because it provides fast and online detection when the threshold for the objects is pre-defined. In this study, an RGB image is first passed through several filters to eliminate noise, such as blur and discrepancies. Next, the image is transformed from BGR to HSV colour space, which enables us to change the saturation and brightness of the image, as well as the colour information. Then, the threshold value for each object is determined, and the image is passed through three morphological operations (erosion, dilation, and erosion) to determine specific features of the objects. The pre-defined colour thresholds specific to the illumination and scene environments are used to obtain the mask of the objects, and the masked image is passed to the OpenCV `findContours` function [98] to obtain the border of the objects. Finally, the output of the contour is sent to the OpenCV `fitEllipse` function [98] to obtain the smallest ellipse fitting this contour, which includes the centre of the masked objects and the angle with respect to the origin of the image plane.

4.4 Experimental setup

A Universal robot (the UR3) with a three-finger grasper has been mounted on a table. Then a Stereolab (ZED) [99] camera was placed in ten different configurations. For each camera configuration, we run several Tsai's hand-eye calibrations to get the camera pose with respect to the robot base because the quality of this approach depends on the collected data. Algorithm 1 presents the steps used for Tsai's approach for one camera configuration. An error threshold (reprojection error of 2 pixels is used in this study because it is found that this value produces reliable HEC results) was given as input to Algorithm 1. Then, the robot arm was moved to at least eight configurations using the ROS Moveit Library [95]. End-effector poses and corresponding checkerboard images were collected by subscribing to the TF library [91] and ZED Camera via ROS [100]. The 2D corner points of the checkerboard were obtained using the OpenCV `findChessboardCorners` function, which takes images and predefined 3D coordinates of the checkerboard's corner with respect to the world coordinates. The estimated corner points in the images were refined by the OpenCV `cornerSubPix` function to obtain more precise calibration. Finally, the camera pose was calculated using the OpenCV `calibrateCamera` function, which takes the checkerboard's 3D and 2D corner points. Hand-eye calibration parameters were then calculated by processing the camera poses and corresponding end-effector poses via Tsai's approach. This operation was repeated until the error from Tsai's approach was less than the defined threshold (reprojection error of 0.9). During the experiments, the estimated hand-eye calibration parameters and time were recorded.

After obtaining the hand-eye calibration parameters using Tsai's approach, the deep learning-based hand-eye calibration method developed in Chapter 3 was applied to the same camera configuration outlined in algorithm 2. This method does not require any calibration target or explicit camera extrinsic calibration. RGB and depth images were captured using the Zed camera, and the pose of the reference point relative to the robot base was recorded using the TF library [91]. The captured images and poses were then fed into a trained network in a different environment to obtain the hand-eye calibration parameters.

Figure 4.4 illustrates the training and testing environments. It should be noted that the neural network underwent training within a distinct environment. It is subsequently applied to the pick-



(a) Training environment (b) Testing environment

Figure 4.4: This figure shows the difference between the training (a) and testing (b) environments.

Algorithm 1 Classic hand-eye calibration algorithm

```
Require: Attach the checkerboard to the robot gripper
Require: $Treshold \geq 0$
  $X_{best}, X \leftarrow \emptyset$                    ▷ Hand-eye calibration parameters,$T_x, T_y, T_z, R_x, R_y, R_z$
  $n \leftarrow 0$
  $Error \leftarrow \infty$
  while $Error \geq Treshold$ do
      $Images \leftarrow \emptyset$
      $EndEffectorPoses \leftarrow \emptyset$
      for $i \leftarrow 1$ to 8 do
          $Images \leftarrow image_i$
          $EndEffectorPoses \leftarrow pose_i$          ▷ Collect images and end-effector's poses
      end for
      CameraPoses= Camera calibration (Images)
      T = Perform Tsai Hand-eye calibration (EndEffectorPoses, CameraPoses)
      $Error \leftarrow T_0$
      $X \leftarrow T_1(T_x, T_y, T_z, R_x, R_y, R_z)$
      $n \leftarrow n + 1$
  end while
  $X_{best} \leftarrow X$
  return $X_{best}$, $n$
Require: Detach the checkerboard from the robot gripper
```

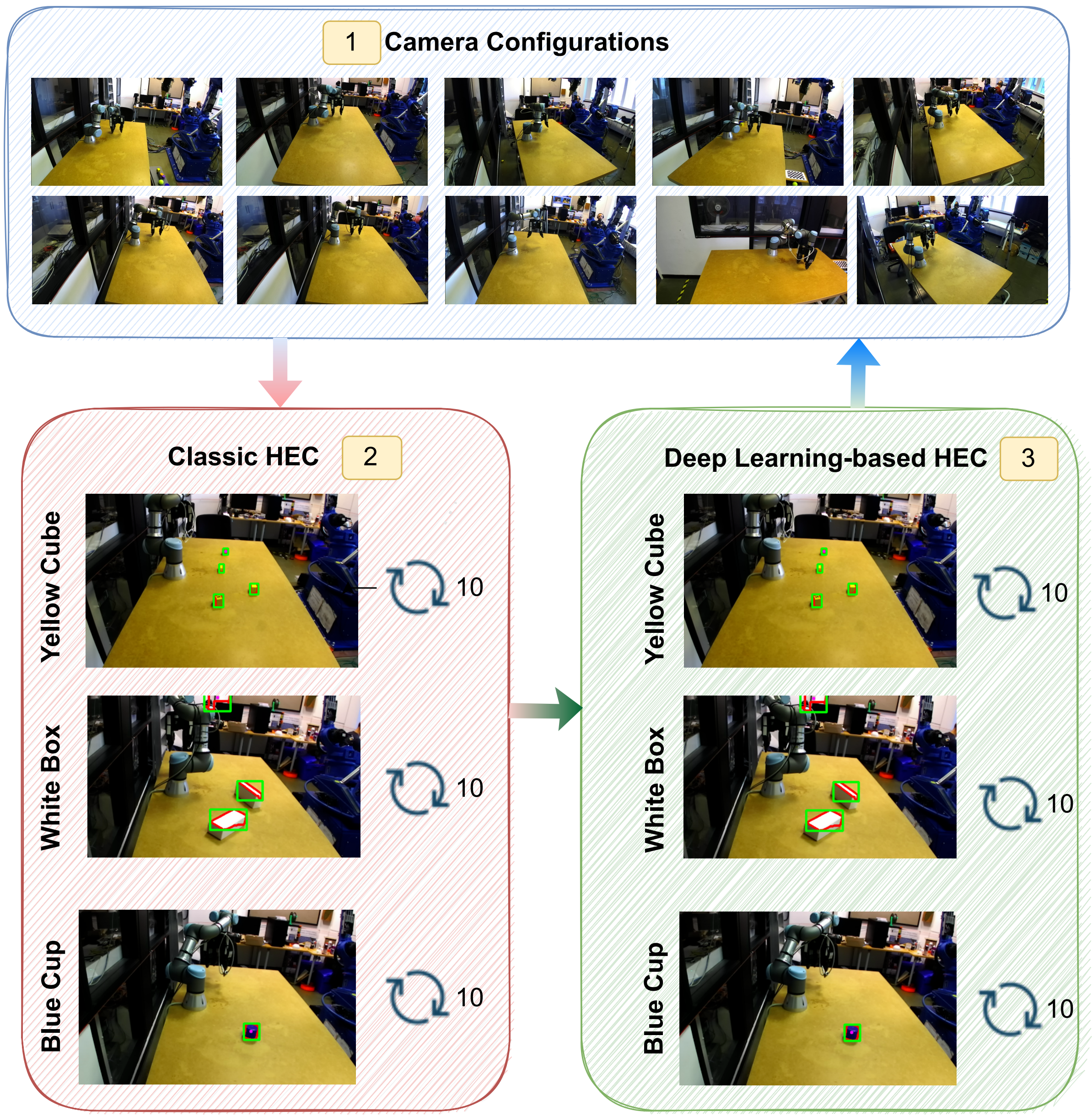


Figure 4.5: This figure outlines the pick and place pipeline used to evaluate two hand-eye calibration methods. (1) displays the camera configuration count. (2) and (3) depict experiments with classic and DL-based approaches, respectively. Each row in (2) and (3) represents object sub-experiments (cubes, boxes, cups), repeated 10 times.

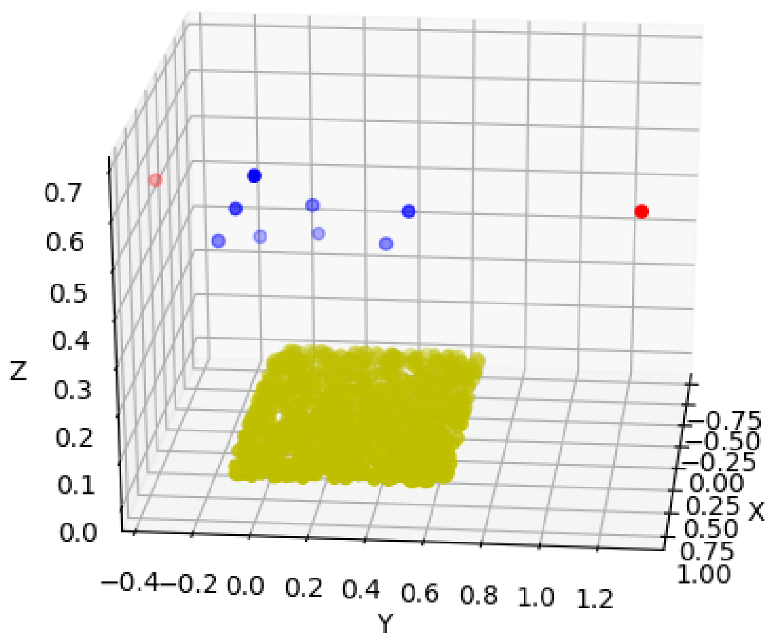


Figure 4.6: This figure depicts 3D camera pose locations and the table. Yellow represents the table area. Blue points are within the network's trained space, while red points are outside. Axes are labelled X, Y, and Z in meters.

and-place task within a new environment without requiring retraining or fine-tuning procedures. However, a vital prerequisite remains that the relative configuration of the camera and robot base aligns with the learned configuration from the preceding environment.

Algorithm 2 Deep learning-based hand-eye calibration algorithm

```
$X \leftarrow \emptyset$
Capture an RGB and depth image
Record corresponding reference point's pose
Get hand-eye calibration parameters processing these data
**return** $X$
```

Figure 4.5 provides an overview of the experimental setup, while Figure 4.6 depicts all camera configurations used in the experiments. The experiment began by placing the camera in an external location, followed by acquiring the hand-eye calibration parameters using the classic approach via algorithm 1 (shown in the bottom left image in Figure 4.5). Next, three sub-experiments were conducted sequentially for each object (yellow cubes, white boxes, and blue cups), as shown in Figure 4.1. After completing all sub-experiments in step 2 (Figure 4.5), the hand-eye calibration parameters were estimated using the DL-based approach via algorithm 2 (depicted in step 3 in Figure 4.5). All sub-experiments were then performed using the newly obtained hand-eye calibration parameters. Finally, the camera configuration was changed, and the above steps were repeated until all camera locations (depicted in Figure 4.6) were visited.

The sub-experiments involve placing one or multiple objects on the table, with the assumption that the objects are separated and their contours do not overlap. An RGB image is then captured using the ZED camera and passed through the object detection module (as detailed in section 4.3.2) via ROS. The detected objects' 3D world coordinates and orientations are calculated by feeding the 2D pixel coordinates obtained from the object detection module into the point cloud and subscribing to the corresponding depth image via ROS. Next, these 3D coordinates are transformed from the camera frame to the robot reference frame using the estimated hand-eye calibration parameters. Finally, the robot is commanded to pick and place the closest object with respect to the robot base using the ROS Moveit library.

The experimental procedure involved the following steps:

- The camera was placed in the first configuration and the hand-eye calibration parameters were obtained using the classic approach.
- Yellow cubes were placed within the robot's workspace, and pick-and-place operations were performed ten times. This was repeated for the white box and blue cup objects.
- Once all objects had been tested with the classic hand-eye calibration method, the same experiments were conducted using hand-eye calibration parameters obtained through the deep learning-based approach for the same camera configuration.
- This process was repeated for the other nine camera configurations; in total ten camera configurations are considered in this study for each HEC approach. .

In total, the pick-and-place procedure was carried out 600 times, taking into account the ten camera configurations, two hand-eye calibration approaches, and three objects with ten repetitions for each object.

4.5 Results

In this study, computational complexity, repeatability, and success rate of performing pick-and-place were employed as performance criteria. Table 4.1 compares the average computational complexity of the classic and deep learning-based hand-eye calibration approaches for ten camera configurations. The first row shows the number of collected images for one camera configuration. The classic approach requires eight RGB images, while the DL-based approach needs only two images (one RGB and one depth). The second row presents the average number of attempts to obtain the hand-eye calibration parameters above and the acceptance level (i.e., 0.9 reprojection error). The average attempt to obtain hand-eye calibration parameters for the classic approach is 8.3, meaning data has to be collected from the beginning eight times for one camera configuration. However, the second approach requires only one attempt to estimate these parameters. Finally, the last row compares the average time to perform these methods to get hand-eye calibration parameters. It was observed that data collection and finding solutions

Table 4.1: Comparison of computational complexity

	Classic HEC	**DL-based HEC**
Number of Image	8 RGB	**1 RGB + 1 Depth**
Average Attempt	8.3±3.37	**1±0**
Average (minutes)	83	**0.06**

with the classic approach for one experiment takes 10 minutes because the calibration target's position and orientation have to be manually changed for each repetition. Apart from calibration target manipulation, attaching and detaching the calibration target was also considered for time calculation. Hence, this time (10 minutes) was multiplied by the average attempts for one camera configuration to calculate the average computational cost of the classic approach. The average computational cost for the DL-based approach is four seconds because it only requires one RGB and depth image to estimate hand-eye calibration parameters.

The repeatability of the DL-based approach (as utilised in [19] and [94](developed in Chapter 3)) was tested by evaluating the standard deviation of the estimated hand-eye calibration parameter for camera poses. For completeness, this equation is:

$$\varepsilon_{std} = \frac{1}{k}\sum_{k=1}^{K}\frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{n}\sum_{i=1}^{n}\|\bar{x}_{kji}-\mu_{kj}\|_2^2\right)^{\frac{1}{2}} \tag{4.1}$$

where k, m, and n refer to the total trained models, camera configurations, and testing images, respectively. While DL-based approaches estimate hand-eye calibration parameters for each image, classic approaches estimate these parameters by processing a set of images. Thus, the standard deviation of the estimated hand-eye calibration parameter for several attempts in a fixed camera pose is used to calculate the repeatability function for classic approaches. Therefore Equation 4.1 was modified for the classic approach as follows:

$$\varepsilon_{std} = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{n}\sum_{i=1}^{n}\|\bar{x}_{ji}-\mu_{j}\|_2^2\right)^{\frac{1}{2}} \tag{4.2}$$

where m and n represent the number of camera configurations and hand-eye calibration attempts for one camera pose. To compute the repeatability score, only the translation component of the hand-eye calibration parameters is considered. This is because directly taking the element-wise average of the rotation component, represented by Euler angles, rotation matrix, or Quaternions, is not possible. For example, the rotation matrix representation must consider orthogonality constraints, or the Quaternions representation requires that the norm of it must be 1.

Table 4.2 presents the repeatability of the three methods for ten camera configurations using Equations 4.1 and 4.2. This study has two deep learning-based approaches to repeatability scores because the developed regression architecture in Chapter 3 ([94]) and the current study

Table 4.2: Repeatability Comparison

Approach	**Position**($\varepsilon_{\mathbf{std(mm)}}$)
Tsai	382.8±150.98
DL-based in this study	**4.95±1.5**
Bahadir [94]	3.95±1.25

are in different environments. The repeatability score of the current experiment environment was calculated by collecting images of each camera configuration during experiments. The repeatability score of Chapter 3 ([94]) is used to show transferring performance of the developed deep learning-based HEC approach from one environment to another.

The repeatability error for Tsai's approach was determined by calculating the standard deviation across all calibration attempts (eight times) for one camera configuration. This measures how much the translation component varies among calibration attempts for the same configuration. The results show that Tsai's repeatability error is high. Tsai's approach propagates error from the orientation component to the translation component because it solves the problem sequentially. Additionally, the success of Tsai's approach is heavily contingent on the selection of calibration target poses, which, if not chosen optimally, can introduce imprecision into the calibration results.

The classic approach exhibits a repeatability error of 96 and 77 times higher than the deep learning-based HEC approaches in terms of environments in [94] and this study, respectively. This difference can be attributed to two primary reasons. Firstly, the classic approach transfers the rotational error into the translational error as it solves the rotation component in Equation 2.27 before determining the translation component by substituting the solution of the rotation. Secondly, the success rate of the classic approach heavily relies on the selection of calibration target poses. While a procedure for selecting these positions has been proposed in [8], it can be challenging to follow this procedure by merely observing the calibration target movements, and expertise is required to collect image samples.

On the other hand, the DL-based approach utilises a deep learning architecture, as explained in Section 4.3.1, which was trained in a different environment than the one in which the experiments were conducted (as shown in Figure 4.4 (left)). Thus, the repeatability error of this approach is only 1.2 times higher than that of [94] and Chapter 3. However, this difference is negligible when accounting for the environmental change, as described in Section 4.4 and depicted in Figure 4.4. The results suggest that the DL-based approach can handle the environmental change and rely on the robot's features to estimate hand-eye calibration parameters.

The success rate of the two methods, classic and DL-based hand-eye calibration, were compared on a real-world pick-and-place pipeline. The performance criteria were divided into four sub-tasks: reaching, touching, picking, and placing. These sub-tasks were sequential, meaning that if reaching failed, the other sub-tasks also failed. Reaching was considered the simplest task and was used to evaluate the estimated hand-eye calibration parameters. However, a successful reaching operation did not directly imply that the following operation could also be performed successfully. Touching was used to determine if the estimated parameters were good enough to touch the objects. Picking was the main evaluation criterion, indicating the success of the estimated hand-eye calibration parameters. Finally, placing was the last task evaluated in this study.

Table 4.3: The results of the classic hand-eye calibration approach on the pick-and-place task.

Classic hand-eye calibration					
	Reach	**Touch**	**Pick**	**Place**	**Objects**
Mean	**100**	**95**	62	**51**	**Cube**
Std	0	7.07	6.32	14.49	
Mean	**100**	**100**	87	84	**Box**
Std	0	0	4.83	6.99	
Mean	**100**	**99**	**74**	**60**	**Cup**
Std	0	3.16	10.75	20	
Mean	**100**	**98**	74.33	**65**	**Total**
Std	0	3.41	7.3	13.83	

Table 4.4: The results of the deep learning-based hand-eye calibration approach on the pick-and-place task.

DL-Based hand-eye calibration					
	Reach	**Touch**	**Pick**	**Place**	**Objects**
Mean	**100**	93.75	**66.25**	50	**Cube**
Std	0	5.18	7.44	16.04	
Mean	**100**	**100**	**88.75**	**85**	**Box**
Std	0	0	9.91	9.26	
Mean	**100**	98.75	73.75	52.5	**Cup**
Std	0	3.54	7.44	19.09	
Mean	**100**	97.5	**76.25**	62.5	**Total**
Std	0	2.90	8.26	14.79	

Table 4.3 presents the mean and 1-standard deviation of experiments performed on three objects using the classic hand-eye calibration approach for ten camera configurations. Table 4.4 shows the corresponding results for the DL-based hand-eye calibration approach, but it only includes the first eight camera configurations due to space constraints. These configurations are shown as blue points in Figure 4.6, while the last two configurations, represented as red points

in Figure 4.6, are outside of the space from which the network was trained. We performed the pick-and-place pipeline for these two configurations using DL-based hand-eye calibration, but only reaching was successful, while the other sub-tasks failed.

The results in Tables 4.3 and 4.4 indicate no meaningful difference between the success rates of reaching and touching operations for every object and both hand-eye calibration approaches. However, due to unstable picking operations, there is a gap between the success rates of picking and placing for every object in both approaches. The Shadow Modular Grasper with three fingers was used in the experiments, and collisions among the fingers could happen when an object was successfully picked, preventing the robot from placing it. This issue can be addressed by developing a more stable grasping approach, but it is beyond the scope of this PhD thesis. Furthermore, the success rates differ for each object in both approaches due to the object's size and materials. For instance, the white box has the highest success rate for pick-and-place tasks in both classic and DL-based hand-eye calibration approaches (84% and 85%, respectively), followed by the blue cup (60% and 52.5%) and yellow cube (51% and 50%)

DL-based hand-eye calibration approach surpasses the results of the classic approach in all sub-tasks in the white box object and picking sub-task for the cube (76.5%). As for the other result (performances of picking and placing tasks for yellow cubes and the blue cup), DL-based hand-eye calibration has competitive results compared to the classic approach. Incidentally, it is crucial to consider that the classic HEC approach is the baseline of the DL-based approach, which means that the best result of the classic approach for a camera pose is used as ground truth to train the DL-based approach.

In conclusion, DL-based hand-eye calibration has comparable results for a real-world pick-and-place pipeline and observes better results for specific sub-tasks. For example, it performs the picking sub-task for all objects 2.25% better than the classic approach. Besides, the DL-based approach is much better than the classic hand-eye calibration approach in terms of computational complexity and repeatability. This means the DL-based approach is more flexible than the classic approach and eliminates data re-collection in camera pose changes. The main limitation of the DL-based hand-eye calibration is that the camera must be placed in a location that lies within the sampled 3D space. Moreover, DL-based hand-eye calibration is trained in a supervised manner and with respect to the best Tsai's hand-eye calibration parameter, which means its success highly depends on the quality of these calibrations.

4.6 Conclusions

The chapter compares the performance of classic and DL-based hand-eye calibration approaches in a real-world pick-and-place pipeline, using three criteria: computational complexity, repeatability score, and success in performing a robotic manipulation task. The camera was placed at ten different locations, and the hand-eye calibration parameters were estimated for each method.

The time and number of attempts required to obtain a valid hand-eye calibration for one camera location were recorded, and pick-and-place tasks were carried out for three objects using each camera configuration and hand-eye calibration method, resulting in 600 trials.

The results indicate that the DL-based approach outperforms the classic approach in terms of computational complexity and repeatability score (77 times better). Although the DL-based approach performs slightly worse overall than the classic method, it performs better than the classic method in some sub-experiments, such as the white box pick-and-place task, where the success rate is 1% higher than the classic approach. Moreover, the DL-based approach outperforms the classic method by 4.2%, 1.75%, and 1.98% in picking sub-tasks for the yellow cube, white box, and total, respectively.

The chapter findings suggest that DL-based hand-eye calibration approach is more adaptable to camera pose changes than traditional methods. The results indicate that obtaining good calibration parameters using the classic approach requires multiple attempts and is challenging to deploy when the camera pose changes due to external factors. The DL-based hand-eye calibration [94](developed in Chapter 3) approach's performance decreases when the camera is placed outside the trained 3D space.

The chapter raises the question of how to make DL-based approaches more adaptable outside the learning environment. Large-scale data capturing, which samples all space around 360 degrees of the robot base with a distance at the beginning of the training network, can eliminate this problem. On the other hand, to address this issue, Chapter 5 plans to update the trained model using Continual Learning on new data outside the learning space. The Continual Learning-based approach can also provide a more flexible hand-eye calibration system, which addresses the layout changes, the distance between the camera and the robot base changes while eliminating the adaptability problem outside the learning space. Furthermore, the Continual Learning-based approach can provide an online hand-eye calibration system which controls the hand-eye calibration parameters while performing a robotic manipulation task and updates the trained model when the novel observation has arrived (camera poses).

Chapter 5

Continual Learning for Hand-Eye Calibration

5.1 Introduction

This section introduces a novel approach based on Continual Learning principles that extends the domain of hand-eye calibration. The experiments conducted in Chapter 4, specifically in the context of pick-and-place tasks, empirically validated the efficacy of the deep learning-based Hand-Eye Calibration (HEC) method introduced in Chapter 3. This method exhibits significant adaptability, particularly in response to environmental alterations, such as changes in the background. This effectively mitigates the impact of environmental variations, including lighting changes often encountered in industrial robotic environments. However, it should be noted that this adaptability is effective when the relative camera and robot base pose remain within the learned space.

When the camera is placed outside of this learned relative space, observation from Chapter 4: shows that the efficacy of our proposed approach diminishes. This scenario can arise when the layout of an assembly line is altered. In this scenario, the relative pose of the camera and robot space can be changed, and the new relative calibration space must be learned to update the calibration parameters continuously. Learning this new calibration can be addressed by extending previously learned calibration space using the continuous learning paradigm. This approach preserves the pre-established relative calibration space while seamlessly incorporating the exploration of novel relative calibration domains. To this end, three Continual Learning-based approaches are proposed: naive CL, reservoir rehearsal, and reservoir rehearsal with camera pose selection.

The effectiveness of these approaches was evaluated through experiments conducted in a simulated and real-world environment. In the simulation environment, a stereo vision camera was positioned in 108 different locations, and 50 different end-effector configurations were run for each camera position to collect samples. In the real world, the camera was positioned in

24 different locations, and 100 end-effector configurations were run for each camera pose. The dataset was partitioned into subsets for Continual Learning-based training. The success of the approach was evaluated based on the accuracy of hand-eye calibration in both environments. Finally, the performance of the three Continual Learning-based approaches was compared with batch training (detailed in Chapter 3) for both environments. The results demonstrate that the presented approach outperforms the other approaches and achieves superior accuracy in both simulated and real-world environments.

5.2 Motivation and Objectives

Recent advancements in deep learning architectures, such as convolutional neural networks, have led to the development of hand-eye calibration systems [18, 19, 94] capable of estimating different camera poses with respect to a single robot base, thus providing greater flexibility and adaptability to industrial manufacturing systems. However, these deep learning-based approaches have limitations; since they are trained offline on fixed datasets, they may not generalise well to new situations or environments. In contrast, humans are able to continuously update and refine their internal model of hand-eye coordination based on new experiences. In robotic manipulation, this type of continuous learning can be achieved through Continual Learning (CL) [101], a paradigm in deep learning where a model is trained on a continuous stream of data over time. By utilising CL, robots can learn and adapt in a similar way to human learning, enabling them to perform more complex tasks and operate in a broader range of environments.

One of the main challenges in CL is the catastrophic forgetting problem [101], which means that when new data is processed, the previously learned information is degraded over time. To overcome this problem, three main strategies have been proposed: regularisation [78, 79, 81], modular architectural design [71, 72] and rehearsal model (buffer) [75–77]. The regularisation methods aim to prevent catastrophic forgetting by adding terms in the objective function to control changes in model weights. Modular architecture design methods mitigate the catastrophic forgetting problem by dedicating sub-modules for different tasks or expanding network architecture when a new task is defined. Rehearsal approaches replay stored old data (some parts) to the model again periodically to prevent the catastrophic forgetting problem.

This study used replay buffer systems to prevent catastrophic forgetting problems for several reasons. In the case of hand-eye calibration, replay buffer systems can be used to store previously learned calibration parameters and corresponding data, allowing the network to continue learning sequentially outside of the learned space while also revisiting and fine-tuning previously observed space. Replay buffers can ensure that a diverse data set is stored and used for training, which can help prevent the catastrophic forgetting problem. This is achieved by periodically sampling from the replay buffer and including new and old data in the training process. Another advantage of using replay buffer systems in hand-eye calibration is that they can help reduce the storage capacity and memory usage required for the calibration process. Since the calibration parameters and data are stored in the replay buffer, there is no need to keep all of the data in memory during training, which can help reduce the overall memory usage of the system. Additionally, replay buffers can enable efficient storage and retrieval of the data, allowing for faster and more efficient training of the network.

Overall, replay buffer systems in hand-eye calibration can help prevent catastrophic forgetting problems, ensure a diverse set of data is used for training, and reduce storage capacity and memory usage, making it an attractive approach for continual learning in hand-eye calibration.

This chapter examines the following research questions to develop *a Continual Learning-based hand-eye calibration system, which extends the learned calibration space through new observations over time*.

- Can hand-eye calibration be handled as a time sequence problem in terms of camera pose changes through Continual Learning?
- Is it possible to extend the learned hand-eye calibration space via Continual Learning?
- How does the continual learning-based HEC approach address the catastrophic forgetting problem over time?

This chapter provides the first study investigating the hand-eye calibration problem through Continual Learning.

5.3 Methodology

The hand-eye calibration problem was approached as a regression problem using deep learning. In chapter 3, an end-to-end deep learning-based approach for hand-eye calibration was developed and tested in a simulated environment and two real robotic environments, namely the Rethink Baxter (depicted Figure 2.1 in Chapter 2) and a Universal Robot 3 (depicted Figure 2.2 in Chapter 2). The deep learning-based approach has been extended to Continual Learning-based hand-eye calibration. Three methods were employed to extend the deep learning-based approach to Continual Learning-based hand-eye calibration (detailed in Table 5.1): naive, reservoir buffer with class balance, and reservoir buffer with class balance with camera pose elimination.

Two Replay-buffer-based methods were adopted to address the catastrophic forgetting problem, which is the challenge of maintaining previously learned knowledge while adapting to new knowledge. As noted in [84], these techniques have shown promising results in regression tasks for domain-incremental scenarios, particularly relevant for Continual Learning-based hand-eye calibration. The goal was to mitigate the issue of catastrophic forgetting, where previously learned information is degraded over time when new data is processed, by storing previous calibration parameters and corresponding data in the replay buffer and training the network sequentially while revisiting and fine-tuning previously observed space. By utilising these methods, the study aimed to improve the stability and robustness of the hand-eye calibration model.

Table 5.1: An overview of the adopted Continual Learning-based approaches in the hand-eye calibration problem

Approach	Description	Time Step	Advantage	Drawback
Naive CL	Update the networks weights by using the training data in the current time step	An area which contains a group of camera poses	Handle the HEC problem with CL	Suffers from the catastrophic forgetting problem
The reservoir buffer with class balance	Updates the network weights by leveraging both the training data in the current time step and samples from past observations.	An area which contains a group of camera poses	Eliminating the catastrophic forgetting problem	It is an offline approach
The reservoir buffer with class balance and camera pose selection	Updates the network weights by leveraging both the training data in the current time step and samples from past observations. Additionally, it employs a threshold to eliminate camera poses that do not contain novelty.	One camera pose	It provides an online CL-based HEC approach	

5.3.1 The formulation of the HEC problem as Continual Learning problem

Continual Learning involves processing data over time to update the model with new observations, which can be formulated as a time sequence problem. This thesis used a group of camera poses, including different robot end-effector configurations, as time steps. The camera poses were grouped based on their proximity to split the dataset into subsets for training and testing in the naive and reservoir buffer with class balance approaches. For instance, in real-world experiments, camera poses on each side of the table was considered a subset. The data generation and split details are presented in the experimental design subsections.

In the reservoir buffer with class balance and camera pose elimination approach, each camera pose was used as a time step, unlike the previous two CL-based approaches. This method controls whether the pose contains novelty and provides an online approach. In other words, these methods store a representative subset of the data, including camera poses not encountered during the previous training iterations. Hence, this method ensures that the model remembers the previously learned information and adapts to new information effectively and online.

The implementation details of the CL-based approaches are presented in Sections 5.4 and 5.5 for the simulated and real-world environments, respectively.

5.3.2 Deep Learning-based Hand-eye Calibration

The developed deep learning-based approach for hand-eye calibration in Chapter 3 served as the baseline for the Continual Learning-based approaches in this thesis. This approach involves processing n different camera configurations and m different end-effector configurations for each camera pose to cover a wide range of hand-eye calibration spaces. A regression architecture based on deep learning is utilised to predict the camera's pose with respect to the robot base, separately for the translation and orientation components of the calibration parameters. The architecture takes RGB and depth images as input, along with the pose of a reference point selected on the robot's end-effector with respect to the robot base. It then produces the translation and orientation components of the calibration parameters as output.

5.3.3 Naive approach

The traditional approach of updating a trained model without any buffering is known as the naive approach. The second row in Table 5.1 illustrates the Naive CL approach. This approach uses a small batch size to update the network weights with new observations each time step, consisting of a set of camera poses detailed in the second column of Table 5.1. The naive approach trains the network with the dataset in the current time step and passes the learned weights to the new time step. Moreover, the naive CL approach evaluates the performance of the trained model for each time step by using all testing subsets. This evaluation shows the performance of the Naive CL approach in both the observed and non-observed spaces.

5.3.4 The reservoir buffer with class balance

The reservoir buffer with class balance approach employs two distinct techniques to tackle the catastrophic forgetting problem (detailed in third row of the Table 5.1). Specifically, this approach combines the use of reservoir and class balance techniques. The reservoir component employs a normal distribution to randomly sample from a buffer of previous data points, which helps maintain a diverse and representative sample of the training data. Meanwhile, class balance ensures that the training data is balanced across all classes, in this case, the camera poses. It prevents the model from focusing excessively on one class at the expense of others.

Algorithm 3 details the training of the network procedure with the reservoir buffer with the class balance approach. This algorithm requires training and testing datasets in the time domain. It also requires three parameters that are c_1, c_2 and c_3. c_1 is the number of sample sizes for previous camera poses, while c_2 is the number of camera poses used for sample collection. Finally, c_3 is the number of training samples for the current time. The dataset partition is detailed in section 5.5.1, and 14 camera poses are used for training in each time step, which means 700 data (RGB and depth images and the pose of the reference point).

Algorithm 3 The reservoir buffer with class balance algorithm

```
Require: $n$ the number of training subsets
Require: Training dataset ($Tr_1$ to $Tr_n$), Test datasets ($Te_1$ to $Te_n$)
Require: $c_1, c_2, c_3 \leftarrow$ the number of sample size, camera configurations, training size
  $S_{n\times n} \leftarrow \emptyset$    ▷ The matrix of the success of the model for each test set in every stage
  for $i \leftarrow 1$ to $n$ do
      if $i \leftarrow 0$ then
          $trainset \leftarrow$ select $c_3$ samples from $Tr_i$ via uniform distribution
          train dl-based regression architecture (chapter 3) by using $trainset$
          $S_i^{1,..,n} \leftarrow$ test network success on each test set $Te_1$ to $Te_n$
      else
          $trainset \leftarrow$ select $c_3$ samples from $Tr_i$ via uniform distribution
          $bufferset \leftarrow$ select $c_1$ samples from $c_2$ camera configuration in the previous time steps
                     (1 to i-1) via uniform distribution
          $trainset \leftarrow trainset + bufferset$
          train the dl-based regression architecture (Chapter 3) by using $trainset$
          $S_i^{1,..,n} \leftarrow$ test network success on each test set $Te_1$ to $Te_n$
      end if
  end for
  return $S_{n\times n}$
```

This approach considers a set of camera poses as a time step presented in the third column of Table 5.1, which represents a 3D region of the calibration space. In the first time step, the deep learning-based regression architecture was trained on the current dataset, and the learned weights were passed to the next time step. The learned weights were updated for the following time steps by processing the current training data and buffer data from previous observations. The performance of the approach was evaluated on the current and all test sets for each time step.

5.3.5 The reservoir buffer with class balance and camera pose selection

It is a hybrid method to streamline the processing of camera poses while maintaining the integrity of previously acquired data through the use of a reservoir buffer and the class balance technique. To determine whether a new camera pose contains novel information, a random sample is drawn and compared to a threshold value, which is determined experimentally and described in the experimental design section. The reservoir buffer plays a crucial role in this process, enabling the retention of relevant data while eliminating redundant information.

Algorithm 4 details the reservoir buffer with class balance and camera pose selection approach. This algorithm requires a threshold for judging whether the current camera pose contains novelty or not. When a new subset is introduced, a sample consisting of 10% of each camera pose is randomly selected using a normal distribution. Next, the mean errors for each camera pose are computed based on the last trained model. If the mean error for any camera pose exceeds a predefined threshold, that camera pose is considered to be novel and marked as such for further processing.

Algorithm 4 The reservoir buffer with class balance and camera pose selection algorithm

```
Require: $threshold$ the threshold for success evaluation
Require: Training dataset ($Tr_1$ to $Tr_n$)
  $S_{1\times n} \leftarrow False$        ▷ The novelty matrix for each camera pose in the current time step
  for $i \leftarrow 1$ to $n$ do
      select 10% of samples from the current camera pose via uniform distribution
      $error \leftarrow$ calculate the mean of the collected sample using the last model
      if $error \geq threshold$ then
          $S_{1\times i} \leftarrow Novelty == True$
      else
          $S_{1\times i} \leftarrow Novelty == False$
      end if
  end for
  return $S_{1\times n}$
```

The training stage started by processing the dataset consisting of a set of camera poses via DL-based regression architecture (developed in Chapter 3) in the first time step. Then this approach considers each camera pose as a new time step (as detailed in Table 5.1). Hence, when the new camera pose arrived, algorithm 4 was used to determine whether the current camera pose contained novelty. If it contains novelty, the learned weights were updated by processing this camera pose and buffer from past observations. The success of the trained model in each time step was evaluated on all test sets to show the progress of the trained model.

5.4 Simulation Experiments

5.4.1 Data generation and split

A Universal Robot 5 equipped with a parallel gripper was placed in the PyBullet simulation environment. To span the camera space from multiple viewpoints, a stereo pair of cameras was positioned in 108 different configurations divided into six subsets of 18 camera configurations each. For each subset, 14 camera configurations were used for training and four for testing. The data split is depicted in Figure 5.1, which shows a three-dimensional Cartesian space with each subset represented by a different colour. It should be noted that each subset is considered a time step for the naive CL and reservoir buffer with a class balance approach. On the other hand, for the reservoir buffer with a class balance and camera pose selection approach, the first subset (blue) is considered the first time step, and subsequently, each camera pose is treated as a separate time step. The overview of the adopted time step strategies is presented in Table 5.2.

The training and testing camera configurations for each subset are indicated by dots and stars in Figure 5.1. This visualisation allows for a better understanding of the distribution of the camera configurations and the data split across the different subsets.

For each camera pose, the end-effector was moved to 50 different configurations to capture the robot's movement in various configurations. For each end-effector and camera pose, RGB and depth images and the pose of the reference point on the robot's end-effector were collected. This process was repeated for all 50 end-effector configurations for each camera pose. The reference point on the robot's end-effector served as a marker for tracking the robot's movement and was located at a specific point on the end-effector for consistency.

Table 5.2: An overview of the time step for Continual Learning-based approaches in the simulated environment

	Time steps/Sim	
Approach	**Total Time step**	**Camera poses each time step**
Naive CL	6	14
The reservoir buffer with class balance	6	14
The reservoir buffer with class balance and camera pose selection	72	1

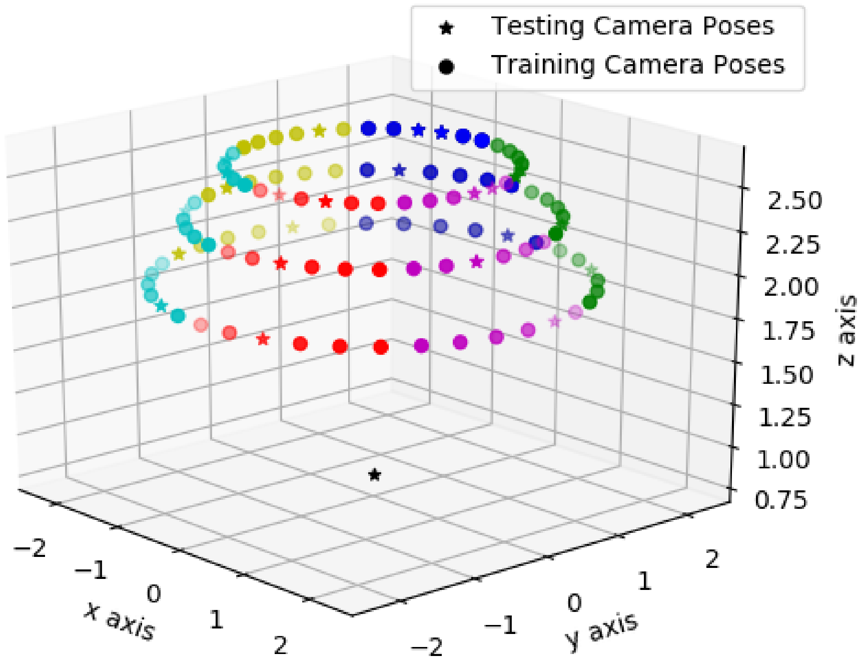


Figure 5.1: This figure visualises the camera configurations used in the PyBullet simulation environment. The 108 camera configurations are divided into six subsets, each represented by a different colour. The dots and stars indicate the camera configurations used for training and testing within each subset, respectively. The black star in the figure represents the robot's base, serving as a reference point for the camera configurations.

5.4.2 Experimental design

Naive CL

The naive Continual Learning approach starts with training the DL-based regression architecture with the first subset (coloured blue), which consists of 14 training camera poses, in the first time step. The learned weights are then updated using the new subsets (S_2 to S_5) in the subsequent time steps. This approach encompasses six-time steps, each containing 14 and 4 camera poses for training and testing, respectively (detailed in Table 5.2). The performance of the naive CL approach is assessed on each test set across all time steps, thereby providing insights into the model's progress over time in both the observed and non-observed calibration spaces.

The reservoir buffer with class balance

The reservoir buffer with the class balance approach follows the same training procedure as the naive approach for the first subset. However, for the second and subsequent subsets, the training set of the current subset is augmented with a small sample of the previous subsets using parameters c_1 and c_2. The c_3 parameter uniformly selects samples from the current training set. Then the learned weights in the first time step are updated by using the sampled training set and the buffer set. This process is repeated until all time steps are visited. The model's success is evaluated on all test sets for each time step.

Table 5.3: The parameters analysis of the reservoir buffer with a class balance CL approach

c1	c2	Buffer Size	c3	Training Data	Total (RGB and depth images)
50	14	700	100%	700	1400
2	14	28	100%	700	728
3	14	28	100%	700	728
3	7	14	100%	700	714
4	7	14	100%	700	714
4	14	28	100%	700	728
4	14	28	50%	350	378
4	14	28	28%	200	228
4	14	28	14%	100	128

Table 5.3 presents the selected parameters and corresponding buffer and training data size according to these parameters. The first row in this table shows if the dl-based regression architecture is trained without Continual Learning each time step. This means each time step considers the current dataset and all observed camera pose from past time steps. The following row shows the buffer size according to the c_1 and c_2 parameters and the sampling of the current training dataset using the c_3 parameter. It should be noted that each camera pose consists of 50 RGB and depth images, and each time step has 14 camera poses (detailed in Table 5.2).

The reservoir buffer with class balance and camera pose selection

The reservoir buffer with class balance and camera pose selection approach is a new method that incorporates the buffer system and camera pose elimination steps. Similar to the previous approach, the DL-based regression architecture developed in Chapter 3 is initially trained with the first subset (marked blue in Figure 5.2). However, each camera pose is considered a time step in this approach (detailed in Table 5.2). When a new camera pose arrives, the elimination step is triggered using Algorithm 4 to identify the camera poses marked as a novel. The novel camera poses are combined with the buffer dataset obtained using Algorithm 3 and used to update the model, while unmarked camera poses are excluded from the buffer system in the next time steps. This process is repeated until all time steps (camera poses) are covered.

As mentioned in Section 5.3.2 and Chapter 3, the DL-based regression architecture consists of two networks, one for estimating translation and the other for orientation components of the calibration parameters. For each network, the novelty thresholds for the translation and orientation are determined separately. The novelty thresholds for translation are 3 and 6 mm, while those for orientation are 4 and 8 degrees, based on the experimental results in Chapter 5.3.2. The small thresholds (3 mm and 4 degrees) are double the best result acquired by training networks without CL, so they eliminate only unnecessary camera poses. The other threshold values (6 mm and 8 degrees) allow networks to eliminate more camera poses, reducing computational complexity and forcing them to learn more distinctive features. Figure 5.2 shows the eliminated camera configurations for the translation network with a threshold of 6 mm.

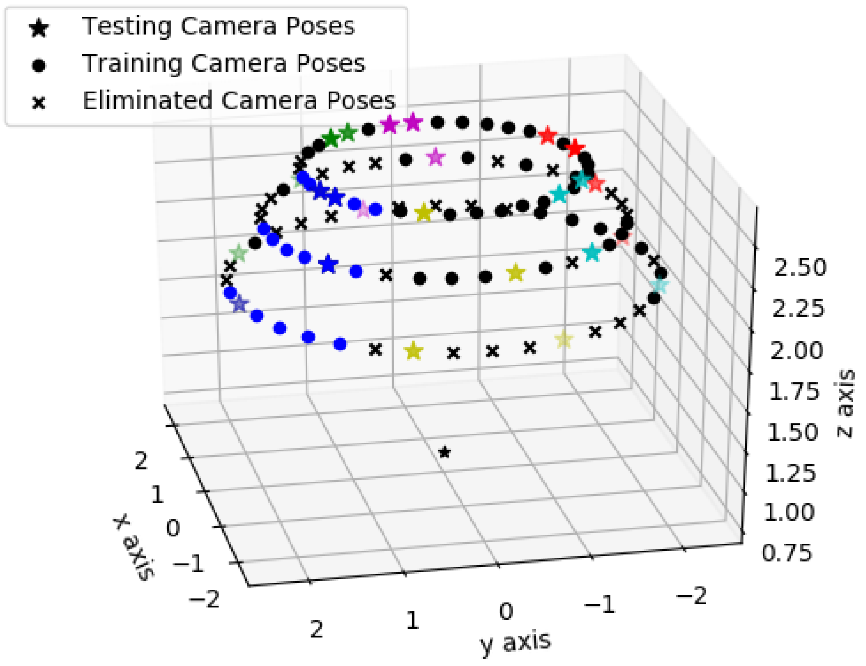


Figure 5.2: This figure illustrates the training stage of the reservoir buffer with class balance and camera pose selection approach in a simulation environment. The testing camera configurations for each subset are represented by colourful stars. The eliminated camera configurations, which are marked as not novel through algorithm 4, are indicated by black crosses.

5.4.3 Experimental results

This section presents the individual results for Naive, the reservoir buffer with class balance, and the reservoir buffer with class balance and camera pose selection approaches. Additionally, the comparison of the success of these approaches is presented in this section.

Naive CL experimental results

Tables 5.4 and 5.5 show the results of the experiments conducted to evaluate the performance of naive CL for each time step for translation and orientation components. To this end, the trained model in each time step was tested on all test set across all calibration space (**S1** to **S6**). Each row in the tables represents a specific time step, while each column displays the error (testing) for the sub-testing sets.

The results indicate that the naive CL approach suffers from catastrophic forgetting in translation and orientation components. Notably, the errors on the diagonal of both tables, where the training and test sets belong to the same subset, are relatively low (only 1.5 times worse than the baseline results), indicating that naive CL can only handle camera configurations within the current spanned space. Therefore, this approach is limited in adapting to new camera configurations outside this space.

Table 5.4: Naive CL experimental results for the translation

	Naive CL Mean / Std (mm)						
Time Step	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**Average**
T_1	6.41±3.66	36.95±5.25	57.29±2.96	69.02±3.88	52.76±3.27	40.13±2.39	43.76±3.57
T_2	24.72±3.74	3.21±0.72	28.4±3.98	54.18±2.64	62.85±3.51	54.54±4.15	37.98±3.12
T_3	55.04±3.48	19.9±4.39	2.19±0.75	18.54±2.28	59.39±4.14	65.14±4.11	36.7±3.19
T_4	63.12±1.32	54.9±5.96	24.93±4.48	1.0±0.01	26.54±0.24	47.31±1.13	36.3±2.19
T_5	48.93±2.37	55.47±1.14	41.53±2.7	23.62±0.76	1.42±0.16	18.58±3.27	31.59±1.73
T_6	24.22±5.6	46.39±3.35	48.08±5.74	46.58±7.05	17.97±5.79	2.2±1.05	30.91±4.76

Table 5.5: Naive CL experimental results for the orientation

	Naive CL Mean / Std (degree)						
Time Step	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**Average**
T_1	2.03±0.26	43.85±0.31	111.67±1.68	160.08±2.09	88.62±1.45	55.72±1.97	77±1.29
T_2	26.85±4.08	1.37±0.34	40.36±0.39	102.57±7.35	153.48±5.04	126.62±3.45	75.21±3.44
T_3	102.84±0.19	23.04±0.28	2.3±0.39	47.86±1.5	143.21±0.6	175.52±0.52	82.46±0.58
T_4	165.16±0.37	98.07±4.71	40.06±0.73	2.13±0.75	47.68±1.0	95.7±2.15	74.8±1.62
T_5	89.63±3.63	139.87±8.58	133.93±5.05	53.91±6.06	2.5±0.11	30.46±1.12	75.05±4.09
T_6	45.61±3.88	116.79±5.96	157.5±1.48	127.62±7.56	29.78±2.85	2.85±0.41	77±1.29

Experimental results of the reservoir buffer with class balance

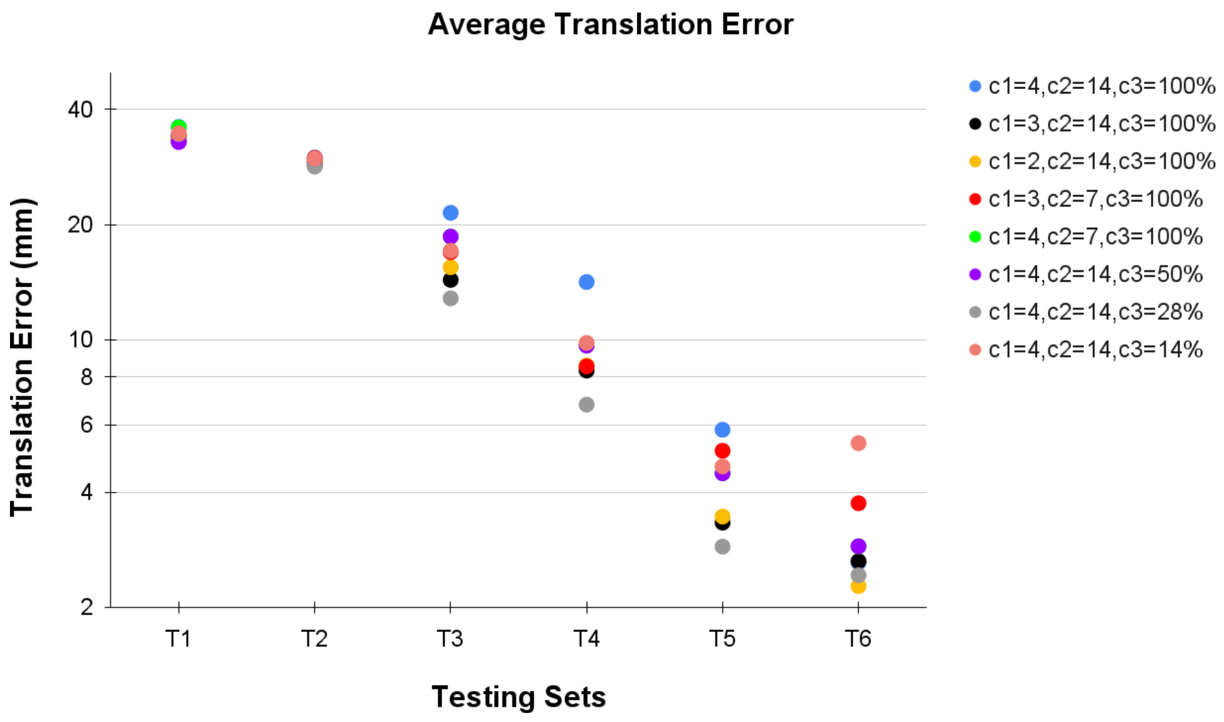


Figure 5.3: This figure shows the performance of the parameter for c_1, c_2, and c_3 for the translation with the reservoir buffer with the class balance approach.

The experiment evaluated the performance of the reservoir buffer with the class balance approach by computing the error for each subtest set (**S1** to **S6**) at every time step ($\mathbf{T_1}$ to $\mathbf{T_6}$). An ablation study was conducted to find the best values for the c_1, c_2, and c_3 parameters, as mentioned in Section 5.4.2 . Figures 5.3 and 5.4 show the results of the ablation study, which examined the influence of different parameterisations of c_1, c_2, and c_3 on translation and orientation estimation, respectively. The figures display the average error for each time step in the estimation process.

The study found that there was no significant difference in performance for translation estimation between the various parameterisations of c_1, c_2, and c_3. This suggests that the network can converge to good results regardless of the values used. Furthermore, the study suggests that estimating the translation component with a reduced amount of data may be possible.

In contrast, for orientation estimation, the parameterisation of c_1=4, c_2=14, and c_3=100% yielded the most accurate results. Decreasing the value of c_3 resulted in slower convergence and an increase in average error in the final step. Additionally, reducing the buffer sample size led to a corresponding increase in error. However, the error for a parameterisation of c_1=4, c_2=14, and c_3=50% was still relatively low, suggesting that some data can be omitted during the training stage.

Tables 5.6 and 5.7 show the experimental translation and orientation parameter estimation results for each time step and sub-testing set using the reservoir buffer with the class balance

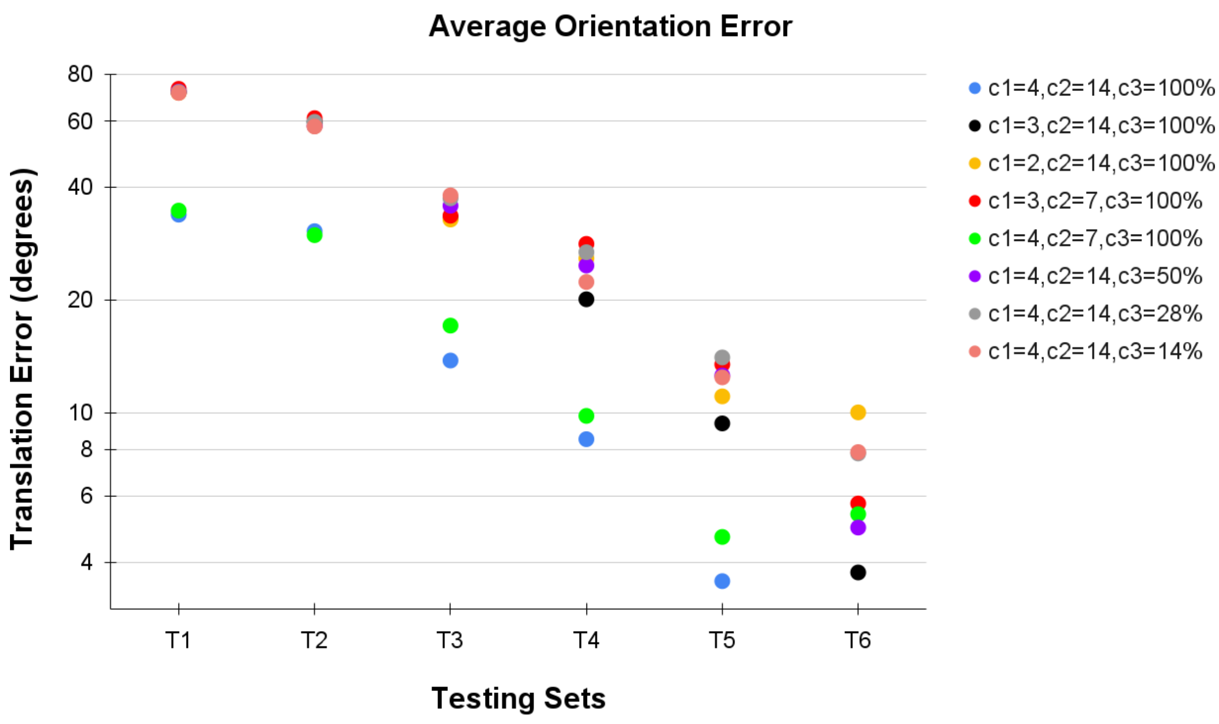


Figure 5.4: This figure shows the performance of the parameter for c_1, c_2, and c_3 for the orientation with the reservoir buffer with the class balance approach.

approach. The values of c_1, c_2, and c_3 are set to four, 14, and 100% of the training set in the current time step, respectively, and these parameter settings yielded the best performance based on the ablation study.

The model's error was computed for the unseen sub-test sets across the entire calibration space (**S1** to **S6**) at each time step. The error for a given time step indicated how well the model had performed on the sub-test sets that spanned the space covered by observed training sets. The final row ($\mathbf{T_6}$) showed the best performance for each sub-test set because the model had spanned all the calibration space in the class balance CL manner. This suggests that the model had learned to generalise well to new data by incorporating class balance during training.

Experimental results of the reservoir buffer with class balance and camera pose selection

Tables 5.8 and 5.9 show the experimental results of the final time step for each testing set (**S1** to **S6**) by employing the reservoir buffer with class balance and came pose estimation approach with two different thresholds. The results in Table 5.8 indicate that the translation error was lower for the 3 mm threshold for each sub-testing set when compared to the 6 mm threshold, implying that a smaller threshold led to increased accuracy. However, the difference in error between the two thresholds was found to be minimal, with a difference of less than 1 mm. Additionally, it was observed that the 6 mm threshold was able to eliminate 29 camera poses, whereas the 3 mm threshold could only eliminate 17 camera poses. These findings suggest that the 6 mm threshold processed fewer data points while performing with similar levels of error.

Table 5.6: Experimental results of the reservoir buffer with class balance approach for the translation, where c_1, c_2, and c_3 are four, 14 and 100%, respectively.

	Random Buffer CL with 4 samples Mean / Std (mm)						
Time Step	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**Average**
$\mathbf{T_1}$	2.21±4.47	36.04±1.0	53.24±3.34	65.0±1.99	51.79±4.17	40.36±0.24	41.44±2.54
$\mathbf{T_2}$	3.04±0.0	2.94±0.66	23.2±1.99	51.08±5.06	55.4±5.73	44.07±0.7	29.96±2.36
$\mathbf{T_3}$	2.4±0.21	1.8±0.13	3.1±0.06	28.82±6.44	52.63±6.24	40.06±1.9	21.47±2.5
$\mathbf{T_4}$	2.1±0.24	2.0±0.43	3.35±0.23	1.85±0.41	41.11±0.61	34.55±4.67	14.16±1.1
$\mathbf{T_5}$	3.76±0.82	1.68±0.05	4.48±0.21	2.16±0.15	2.12±0.09	20.73±3.62	5.82±0.82
$\mathbf{T_6}$	2.49±0.34	1.99±0.4	3.74±0.76	2.73±1.0	2.61±0.1	2.16±1.04	3.12±0.61

Table 5.7: Experimental results of the reservoir buffer with class balance approach for the orientation, where c_1, c_2, and c_3 are four, 14 and 100%, respectively.

	Random Buffer CL with 4 samples Mean / Std (degrees)						
Time Step	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**Average**
$\mathbf{T_1}$	1.77±0.2	41.84±0.04	110.46±0.33	163.78±3.04	89.2±2.02	57.74±2.72	77.47±1.39
$\mathbf{T_2}$	2.15±0.29	1.41±0.08	53.06±1.62	131.3±8.16	104.83±1.19	81.58±3.63	62.39±2.5
$\mathbf{T_3}$	3.6±1.52	1.94±0.75	3.86±0.7	84.62±24.93	114.77±2.53	92.49±4.84	50.21±5.88
$\mathbf{T_4}$	3.33±0.41	1.41±0.2	3.75±0.2	1.48±0.26	60.39±15.71	82.58±12.67	25.49±4.91
$\mathbf{T_5}$	4.71±1.45	1.67±0.32	3.4±0.38	1.81±0.36	1.5±0.27	21.7±0.99	5.8±0.63
$\mathbf{T_6}$	2.45±0.34	1.45±0.1	3.13±0.67	2.09±0.33	1.68±0.15	1.15±0.24	2.33±0.31

As indicated in Table 5.9, the final time step orientation error for each testing set was evaluated using both 8 degrees and 4 degrees thresholds. The results showed that a higher error threshold (8 degrees) led to better performance. However, the lower threshold results were still relatively strong, with an average of only 18% or 0.6 degrees worse than the higher threshold. It should be noted that the 8 degrees threshold eliminated 42 camera poses (60%). In comparison, the 4 degrees threshold eliminated 32 camera poses (45%).

Comparison of the CL approaches

Figures 5.5 and 5.6 illustrate the performance of the CL approaches for each testing set in terms of the translation and orientation components with the baseline approach, which is DL-based regression architecture (detailed in Chapter 3) trained with all training sets building from the ground up without CL.

The results demonstrate that the naive CL approach has the poorest performance for translation and orientation error, primarily due to forgetting previously learned knowledge over time. The naive CL approach has competitive results with the final testing subset (**S6**) because this sub-test set is in the final training subset, where the naive CL last updates the model.

In contrast, the buffer-based approaches, including the class balance approach, did not suffer from the catastrophic forgetting problem. Figures 5.5 and 5.6 show no significant perfor-

Table 5.8: Experimental results of the reservoir buffer with class balance and pose estimation approach for the translation error (mm) in the final time step.

	Camera Pose Selection and Random Buffer CL with 4 samples Mean / Std (mm)						
Treshold	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**Average**
6 mm	2.44±0.27	2.8±0.4	4.4±0.9	1.61±0.44	1.92±0.31	2.52±0.59	2.62±0.49
3 mm	1.91±0.22	2.08±0.33	3.11±0.91	1.29±0.15	1.47±0.14	2.0±0.5	1.98±0.38

Table 5.9: Experimental results of the reservoir buffer with class balance and pose estimation approach for the orientation error (degrees) in the final time step.

	Camera Pose Selection and Random Buffer CL with 4 samples Mean / Std (degrees)						
Treshold	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**Average**
8 degree	4.2±0.37	3.42±1.11	4.22±0.48	2.22±0.38	1.96±0.31	2.11±0.59	3.02±0.54
4 degree	4.95±0.69	2.01±0.26	3.05±0.35	1.54±0.2	1.81±0.08	1.55±0.14	2.49±0.29

mance differences among these approaches for both components. However, the buffer-based approaches with camera pose selection allowed for the elimination of unnecessary camera poses, resulting in reduced computational time over time. This makes them more adaptable for processing stream (online) data. It should be noted that the best parameters for the reservoir buffer with the class balance approach were only included in the plots for simplicity. Except for the naive approach, CL-based approaches have competitive results with the baseline for both translation and orientation.

Based on the comparison of CL approaches with the baseline approach, it can be concluded that CL approaches improve the model's translation and orientation estimation performance. The buffer-based approaches, including the class balance approach, performed better than the naive approach, which suffered from the catastrophic forgetting problem. Moreover, the buffer-based approaches with camera pose selection allowed for eliminating unnecessary camera poses, reducing computational time and making them more suitable for processing stream (online) data. The reservoir with the class balance approach has 1.5 mm and 0.5 degrees higher errors for the translation and orientation compared to the baseline (developed in Chapter 3). The reservoir with the class balance and camera pose selection approach has 0.3 mm and 1.1 degrees higher errors than the baseline. Overall, these results suggest that CL approaches, particularly buffer-based approaches, can improve the accuracy and efficiency of camera pose estimation models.

5.5 Real-world Experiments

5.5.1 Data collection and split

A Universal Robot 3 with a three-finger grasper (detailed in section 2.2.1) was placed on the table. To span camera space from different viewpoints, a stereo pair of cameras (The StereoLabs

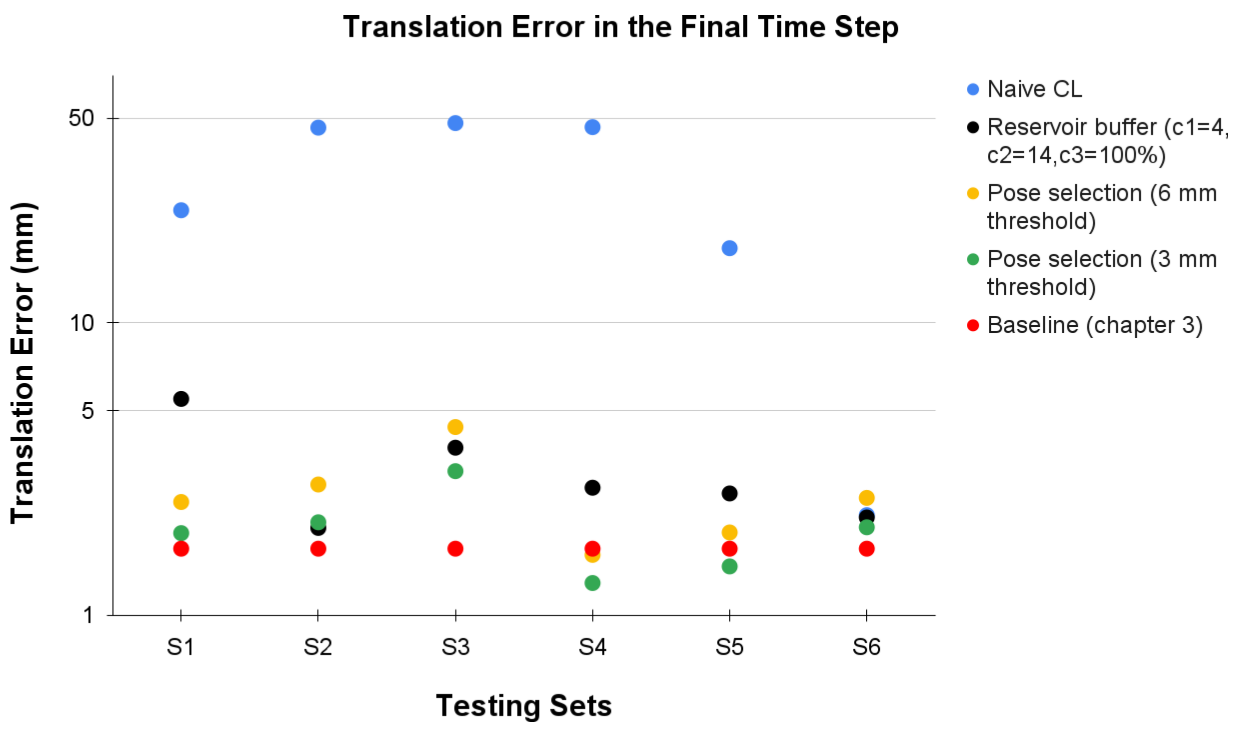


Figure 5.5: This figure compares each testing set's translation errors for adopted CL approaches with the baseline approach in the final time step. The baseline approach is dl-based regression architecture trained by all training sets building from the ground up without CL.

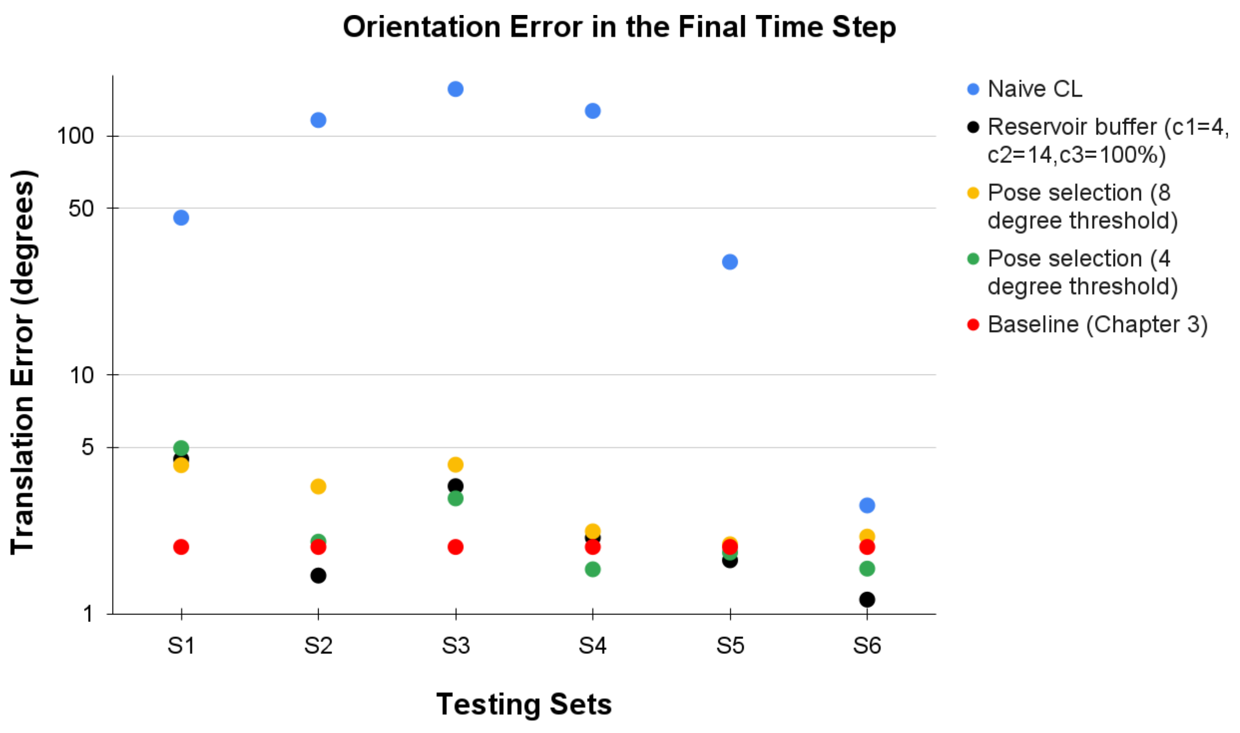


Figure 5.6: This figure compares each testing set's orientation errors for adopted CL approaches with the baseline approach in the final time step. The baseline approach is dl-based regression architecture trained by all training sets building from the ground up without CL.

camera (ZED) presented in section 2.2.3) was positioned in 24 locations, illustrated in figure 5.7. These camera configurations cover three sides of the table where the robot is mounted, which enables us to consider 90 and 180 degrees rotations which contain the challenging perpendicular and reflection configurations. These rotations cause a significant change in the appearance of the robot and the environment. To span the robot workspace space, the end-effector of the robot was moved to 100 configurations for each camera pose.

Figure 5.7 shows the camera configurations, where red and blue dots represent the train (19 camera poses) and test (5 camera poses) sets. For data partition, three subsets (**S1-S3**) were used, where each side was composed of the one-time step. Table 5.10 shows the time step selection strategy used in real-world experiments.

Table 5.10: An overview of the time step for Continual Learning-based approaches in the real-world environment

	Time steps/Real-world	
Approach	**Total Time step**	**Camera pose each time step**
Naive CL	3	6
The reservoir buffer with class balance	3	6
The reservoir buffer with class balance and camera pose selection	14	1

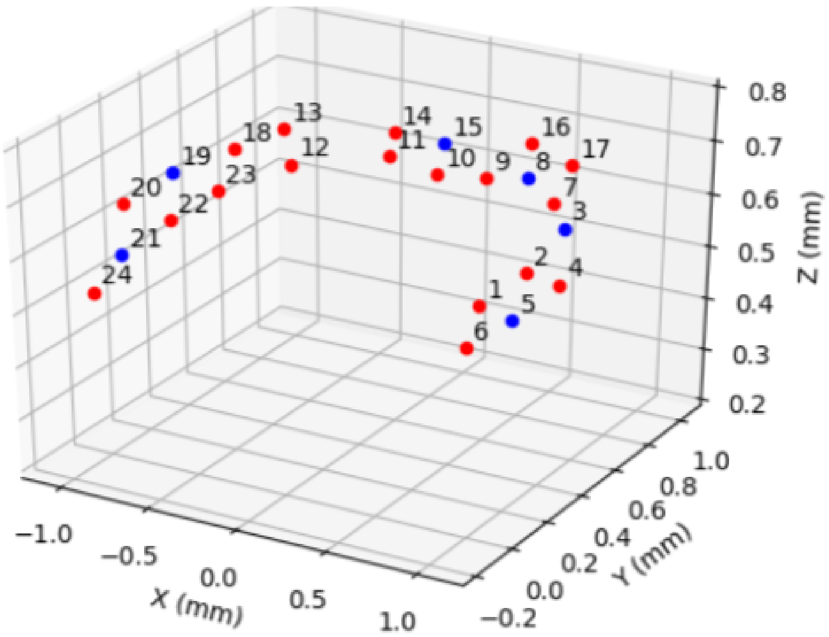


Figure 5.7: This figure shows the camera configurations used in the real-world environment. The 24 camera configurations are divided into three subsets. The red and blue colours represent the training and testing sets, respectively.

5.5.2 Experimental design

Naive CL

The naive CL approach processes the data separately for translation and orientation components by employing the network detailed in Chapter 3. It starts to train the network using the first subset and then update the trained model until all subsets are covered.

The reservoir buffer with class balance

This approach, similar to the naive CL approach, processes the data separately for the translation and orientation components using the network described in Chapter 3. The training process starts with the first subset, and the model is updated until all subsets are covered. However, unlike the naive approach, this approach utilises algorithm 3 to augment the training set with past observations. The parameters c_1, c_2, and c_3 are chosen for the real-world experiments as four, all observed camera poses in the previous time step, and 100% based on the simulation results discussed in section 5.5.3.

The reservoir buffer with class balance and camera pose selection

The reservoir buffer with class balance and camera pose selection approach follows a training procedure different from the buffer and naive CL approaches. Initially, the translation and orientation networks are trained using the first subset, which includes six camera poses. In contrast to the previous approaches, the other 13 camera poses are considered independent time steps, a more realistic representation of camera pose changes in real-world applications, which resembles a stream (online) hand-eye calibration.

At the end of the first time step, Algorithm 4 is used to test whether the new camera pose includes novelty. If so, the camera poses with novelty at any time step are used to update the last model using the reservoir buffer with the class balance approach. Specifically, four samples are taken from past novel camera observations to augment the current training set.

To determine the presence of novelty, a threshold is used for the translation and orientation components. Based on experimental results obtained from simulations, the thresholds for the translation and orientation components are set to 3 mm and 4 degrees, respectively.

5.5.3 Experimental results

This section presents individual and comparison results of each CL approach for the translation and orientation components in the real-world environment.

Naive CL experimental results in the real-world environment

Tables 5.11 and 5.12 present the experimental results of the Naive CL approach for the translation and orientation error at each time step and the unseen test set in the real-world environment. The results indicate that there is a good performance for the test set at the current time step, but catastrophic forgetting occurs for the previous time steps. Specifically, the average translation errors (as shown in Table 5.11) for the **S2** and **S3** test sets in the final step are 4.14 and 3.49 mm, respectively, while the error for the **S1** test set is 31.51 mm.

Table 5.11: Naive CL experimental results for the translation in the real-world

UR3	**Naive CL Mean / Std (mm)**			
Time Step	**S1**	**S2**	**S3**	**Average**
T_1	1.53±0.02	20.86±0.04	38.09±0.05	20.16±0.04
T_2	25.96±0.63	5.78±0.85	17.75±0.88	16.5±0.79
T_3	31.51±1.32	4.14±1.84	3.49±1.24	13.05±1.49

Table 5.12: Naive CL experimental results for the orientation in the real-world

UR3	**Naive CL Mean / Std (degree)**			
Time Step	**S1**	**S2**	**S3**	**Average**
T_1	4.62±0.05	77.13±0.03	167.53±0.06	83.09±0.05
T_2	111.27±0.19	10.13±0.37	60.6±1.69	60.67±0.75
T_3	159.89±4.54	9.51±0.5	9.59±0.88	59.66±1.97

Experimental results of the reservoir buffer with class balance in the real-world environment

Tables 5.13 and 5.14 present the translation and orientation error results for each time step and sub-testing set when using the reservoir buffer with the class balance approach, with parameters four, all observed camera poses in the previous time step, and 100% chosen based on the ablation study results in the simulation environment.

Table 5.13 indicates a decrease in translation error as camera poses are processed over time for both the current and previous test sets. Moreover, the average error for each test set in the final step is competitive with the deep learning-based HEC described in Chapter3, and the catastrophic forgetting problem observed in the Naive CL approach is resolved.

Regarding the orientation component (detailed in Table 5.14), a similar trend is seen in the translation regarding model success over time, and the impact of the catastrophic forgetting

Table 5.13: Experimental results of the reservoir buffer with class balance approach for the translation in the real-world environment, where c_1, c_2, and c_3 are four, all observed camera poses in the previous time step and 100%, respectively.

UR3	**Random Buffer CL with 4 samples Mean / Std (mm)**			
Time Step	**S1**	**S2**	**S3**	**Average**
$\mathbf{T_1}$	1.83±0.06	10.06±0.19	34.59±0.11	15.49±0.12
$\mathbf{T_2}$	3.83±0.4	1.59±0.17	24.32±1.66	9.91±0.74
$\mathbf{T_3}$	4.17±0.47	2.88±1.16	3.46±0.41	3.5±0.68

Table 5.14: Experimental results of the reservoir buffer with class balance approach for the orientation in the real-world environment, where c_1, c_2, and c_3 are four, all observed camera poses in the previous time step and 100%, respectively.

UR3	**Random Buffer CL with 4 samples Mean / Std (degree)**			
Time Step	**S1**	**S2**	**S3**	**Average**
$\mathbf{T_1}$	4.42±0.14	35.17±0.44	158.94±0.19	66.18±0.26
$\mathbf{T_2}$	6.32±1.52	2.89±0.29	110.92±3.58	40.04±1.8
$\mathbf{T_3}$	7.38±0.87	5.32±0.53	16.23±1.16	9.64±0.85

problem is reduced. However, the average results in the final time step are above the baseline detailed in Chapter 3, which are 1.4 mm and 2.8 degrees for translation and orientation, respectively.

Experimental results of the reservoir buffer with class balance and camera pose selection in the real-world environment

Tables 5.15 and 5.16 display experimental results for the translation and orientation components, respectively, in a real-world environment using the reservoir buffer with the class balance and camera pose selection approach. This approach considers each camera pose in the subsets as a time step, except for the first subset, resulting in 14 time steps. Parameters used include four, all observed novel camera poses in the previous time step, and 100% for the buffer approach, while 3mm and 4 degrees thresholds for the camera pose selection for translation and orientation, respectively. It should be noted that each experiment was repeated three times to mitigate the effects of stochasticity.

Table 5.15 demonstrates that the translation error for the first subset marginally increases after the first time step ($\mathbf{T_1}$) when new camera poses are considered. However, this increase is only 1mm and can be tolerated. For **S2** and **S3**, the errors decrease over time. Additionally, the threshold eliminates six camera poses, excluding novelty, which reduces computational complexity.

Table 5.15: Experimental results of the reservoir buffer with class balance and pose estimation approach for the translation in the real-world environment, where c_1, c_2, c_3, and threshold are four, all observed novel camera poses in the previous time step, 100% and 3 mm, respectively.

UR3	**Camera Pose Selection and Random Buffer CL with 4 samples Mean / Std (mm)**			
Time Step	**S1**	**S2**	**S3**	**Average**
T_1	1.83±0.01	10.14±0.42	34.59±0.06	15.52±0.16
T_2	2.08±0.13	3.49±0.15	30.33±0.71	11.97±0.33
T_3	2.04±0.14	2.86±0.35	25.1±4.29	10±1.59
T_4	2.16±0.27	3.26±0.44	20.5±0.25	8.64±0.32
T_5	2.16±0.27	3.26±0.44	20.5±0.25	8.64±0.32
T_6	2.16±0.27	3.26±0.44	20.5±0.25	8.64±0.32
T_7	2.16±0.27	3.26±0.44	20.5±0.25	8.64±0.32
T_8	2.45±0.49	2.75±0.08	15.25±2.66	6.82±1.08
T_9	2.94±1.07	3.2±0.7	10.77±0.56	5.64±0.78
T_{10}	2.94±1.07	3.2±0.7	10.77±0.56	5.64±0.78
T_{11}	2.54±0.53	2.99±0.24	4.16±0.77	3.23±0.51
T_{12}	2.54±0.53	2.99±0.24	4.16±0.77	3.23±0.51
T_{13}	2.6±0.29	3.02±0.33	3.78±0.92	3.13±0.51
T_{14}	**2.34±0.3**	**2.48±0.38**	**4.77±0.6**	**3.2±0.43**

Table 5.16 shows that the orientation trend is similar to the translation for **S1**, **S2**, and **S3**. Although the error in the **S3** test set decreases over time, it is still high compared to **S1** and **S2**. Increasing camera pose in that region may address this performance gap.

Comparison of the CL approaches in the real-world environment

Figures 5.8 and 5.9 present a comparison of different CL approaches with the baseline (detailed in Chapter 3) for the translation and orientation components, respectively. The naive CL approach exhibits the worst performance for both translation and orientation components. In comparison to the baseline, it is 14 and 31 times worse for translation and orientation components in the S1 test set in the final time step. Figure 5.8 demonstrates that the buffer-based CL approaches have results that are competitive with the baseline for the translation error. On the other hand, Figure 5.9 shows that the buffer-based approaches perform comparably to the baseline for the first two test sets, but there is a performance gap in the final test set. This difference may be due to the camera pose in this set representing the reflection (180-degree rotation), which is a challenging scenario because it causes a significant change in the appearance of the robot and the environment. The experimental result in the simulation (detailed in the section 5.4.3) shows that CL-based approaches have the potential to produce the same success. The gap in the final test set can be eliminated by increasing the number of camera poses in that region.

Table 5.16: Experimental results of the reservoir buffer with class balance and pose estimation approach for the orientation in the real-world environment, where c_1, c_2, c_3, and threshold are four, all observed novel camera poses in the previous time step, 100% and 4 degrees, respectively.

UR3	**Camera Pose Selection and Random Buffer CL with 4 samples Mean / Std (degrees)**			
Time Step	**S1**	**S2**	**S3**	**Average**
$\mathbf{T_1}$	4.4±0.18	34.89±0.19	158.74±0.42	66.01±0.26
$\mathbf{T_2}$	5.09±0.55	8.36±0.26	142.33±2.09	51.93±0.97
$\mathbf{T_3}$	6.72±1.1	5.6±0.28	105.84±8.1	39.39±3.16
$\mathbf{T_4}$	6.72±1.1	5.6±0.28	105.84±8.1	39.39±3.16
$\mathbf{T_5}$	6.74±1.08	5.04±1.06	103.53±4.91	38.44±2.35
$\mathbf{T_6}$	6.39±1.38	4.15±1.13	109.87±9.1	40.14±3.87
$\mathbf{T_7}$	6.39±1.38	4.15±1.13	109.87±9.1	40.14±3.87
$\mathbf{T_8}$	7.22±1.32	4.87±0.92	67.58±10.84	26.56±4.36
$\mathbf{T_9}$	5.57±0.7	5.7±0.49	56.12±2.55	22.46±1.25
$\mathbf{T_{10}}$	5.05±0.95	6.42±2.99	41.86±1.39	17.78±1.78
$\mathbf{T_{11}}$	5.95±0.8	4.35±0.54	20.09±3.29	10.13±1.54
$\mathbf{T_{12}}$	6.19±1.35	3.85±0.86	13.02±0.75	7.69±0.99
$\mathbf{T_{13}}$	6.17±1.37	4.6±0.49	14.48±2.0	8.42±1.29
$\mathbf{T_{14}}$	**7.07±2.2**	**3.87±0.43**	**12.0±0.88**	**7.65±1.17**

5.6 Conclusion

This chapter has verified the following hypothesis:

A Continual Learning-based hand-eye calibration system can extend the learned hand-eye calibration space through new observations over time.

This chapter presents a Continual Learning-based approach for hand-eye calibration, which allows for the extension of a previously learned calibration space with new observations over time. Three different Continual Learning-based methods are proposed: a naive approach, a reservoir buffer with class balance, and a camera pose selection approach with the buffer. Experimental results on both simulated and real-world environments show that the CL-based approaches, except for the naive one, achieve competitive performance with the batch learning-based approach. This suggests that the hand-eye calibration problem can be effectively treated as a time sequence problem, and the learned space can be extended without retraining the network building from the ground up with all the camera poses. The ability to extend the learned space also makes the hand-eye calibration system more adaptable to changes in camera pose over time.

The robotic system's calibration may be updated in real-time using CL-based hand-eye calibration approaches, essential for preserving accuracy over time. This is particularly crucial in production environments since the robotic system may encounter changes to the environment

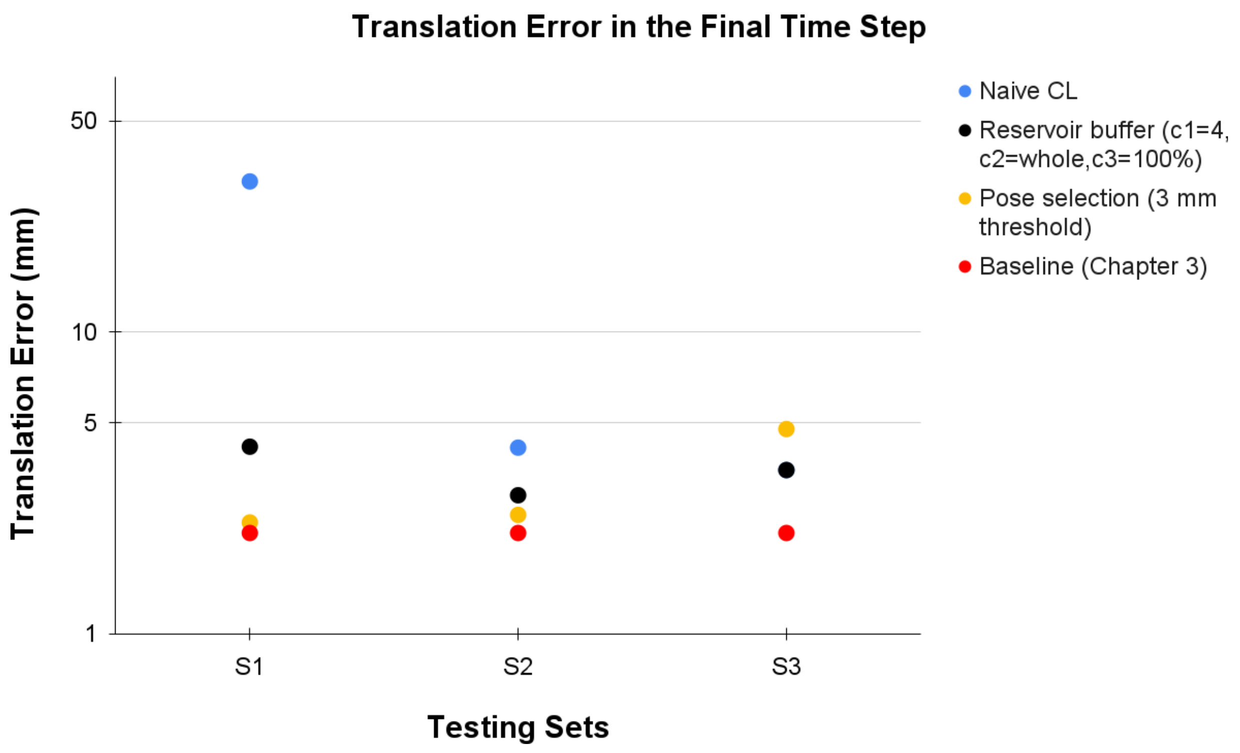


Figure 5.8: This figure compares each testing set's translation errors for adopted CL approaches with the baseline in the final time step.

or the products it is manipulating, which would cause adjustments to the calibration parameters. The system can retain accuracy and precision throughout the operation by updating the calibration parameters in real time, resulting in higher-quality output and greater efficiency.

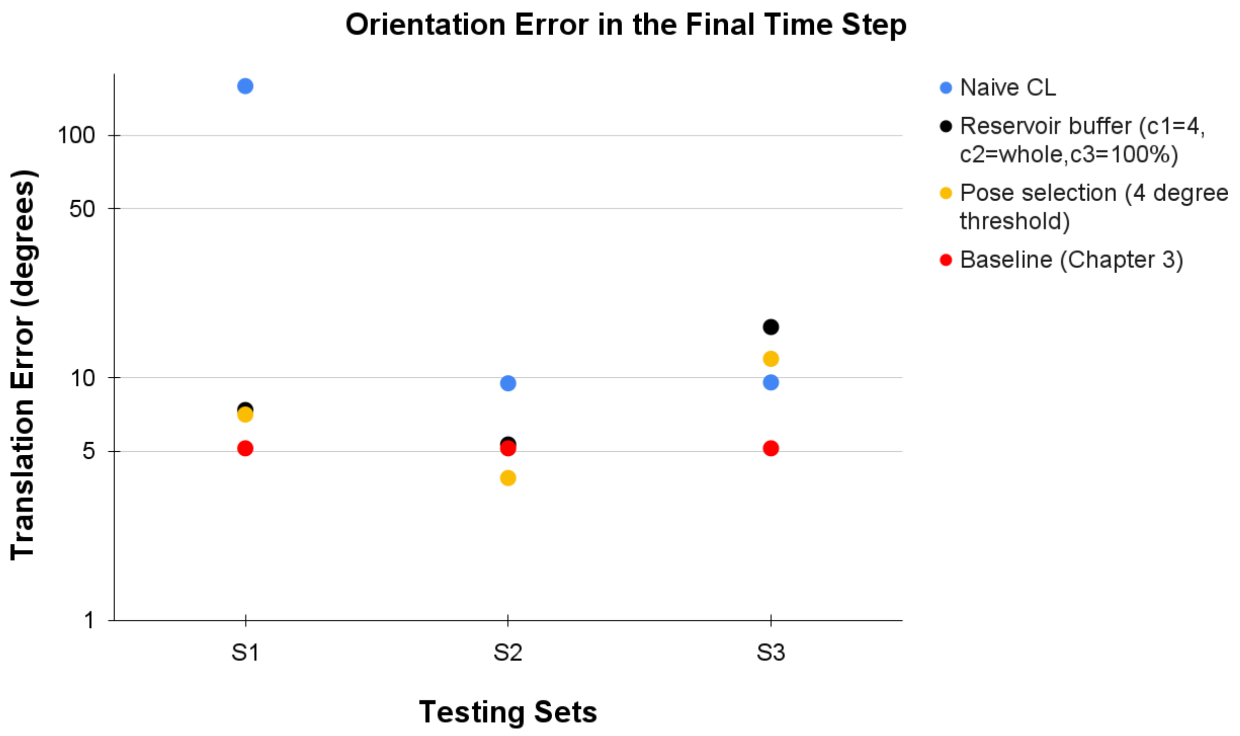


Figure 5.9: This figure compares each testing set's orientation errors for adopted CL approaches with the baseline in the final time step.

Chapter 6

Conclusion

The main goal of this thesis was to develop a flexible and autonomous hand-eye calibration approach based on deep learning, which has the following characteristics:

- It continuously updates the calibration parameters in the learned calibration space while eliminating data recollection and retraining as well as remaining competitive metric accuracy for the classic hand-eye calibration approaches, which could not address the camera pose changes over time. Furthermore, the developed deep learning-based hand-eye calibration approach has a better repeatability score (precision) than the classic and other deep learning-based approaches (Chapter 3).

- The developed approach has competitive results for performing a real-robotic task (a pick-and-place) with a classic hand-eye calibration approach while reducing the computational complexity, unlike classic metric error (Chapter 4).

- The developed approach extends its learned calibration space through new observations over time while preserving metric accuracy compared to training the network with all data building from the ground up (batch-learning) (Chapter 5).

6.1 Main Research Findings

In Chapter 3, a deep learning-based regression architecture was developed to handle the dynamic hand-eye calibration problem, which requires addressing camera pose changes over time without data recollection. To this end, a camera was positioned in *n* different configurations to span the camera space, and the robot end-effector was run *m* different configurations for each camera pose to span the robot space. Meanwhile, RGB and depth images and the poses of a single reference point selected on the robot's end-effector with respect to the robot bases through the robot kinematic chain were collected in a simulation environment and two real robotic environments. Using a single reference point simplifies data collection and reduces the dependence on external factors like specific calibration targets or image-based inference, leading to a more straightforward and reliable hand-eye calibration process. Additionally, extending a single reference point for hand-eye calibration to multiple robot models offers consistency, efficiency, and standardisation benefits, making the calibration process more streamlined and accessible across various industrial robotic applications. The deep learning-based regression architecture estimates calibration parameters continuously within the learned calibration space. The results showed that the developed approach has competitive results (1.69 mm and 1.91 degrees for the translation and orientation components, respectively) with the classic approaches in the simulation environment. A direct comparison of the developed approach with the classic approaches in real-world environments is difficult because the ground-truth calibration parameters could not be precisely estimated in the real world. However, the repeatability score, which shows the precision of the developed approaches, was used to compare with the classic and other deep learning-based hand-eye calibration approaches [18, 19]. These results showed that the developed hand-eye calibration approach has 96 times better than the Tsai approach [8]. Overall, the developed hand-eye calibration approach has the ability to update calibration parameters within the learned calibration space.

Chapter 4 presented a comprehensive analysis of a deep learning-based hand-eye calibration approach in a real-world robotic manipulation task, specifically pick-and-place. The primary objective was to demonstrate the method's effectiveness on task performance, unlike metric error. Furthermore, the results were compared with a classic hand-eye calibration approach [8] with respect to performing a real robotic manipulation task, computational complexity and precision. The results indicated that the developed hand-eye calibration approach had 76.25% and 62.5% success in picking and placing the object without data collection after training, while the classic approach had 74.33% and 65% for these operations requiring data recollection for each camera poses. Moreover, the average time consumption was 83 minutes for the classic approach, whereas the developed approach required 0.06 minutes for the calibration updating operation. Finally, 3.9 mm and 328 mm were the developed and classic hand-eye calibration approaches' precision errors, respectively. Overall, the results showed that the developed hand-eye calibration approach has competitive results for performing a real robotic manipulation task while decreasing the computational complexity and precision error.

Chapter 5 of the study focuses on Continual Learning, which aims to extend the learned calibration space of the developed hand-eye calibration approach over time. Three different Continual Learning-based approaches were developed and tested in simulated and real-world environments. The first approach updated the weights of the deep learning-based regression architecture developed in Chapter 3, considering only new observations outside of the previously learned calibration space over time. However, this approach encountered the problem of catastrophic forgetting, which involves losing previous knowledge. To overcome this issue, a reservoir buffer with the class balance approach was used for the dynamic hand-eye calibration problem. Moreover, a camera pose selection algorithm was included in the approach to eliminate unnecessary observations that had already been learned in the previous stages and reduce the computational complexity. The results demonstrated that the last two approaches effectively extended the learned calibration space of the developed hand-eye calibration approach by using new observations while maintaining metric accuracy in both simulated and real-world environments. The metric accuracy gaps for translation and orientation components in the simulation are 0.28 mm and 0.5 degrees, while they are 1.03 mm and 2.5 degrees in the real-world environment.

6.2 Implications of the Knowledge

The findings of Chapters 3 and 4 suggested that the developed approach in this thesis can reduce the computational complexity of hand-eye calibration in real-world environments, leading to more efficient and effective robot manipulation tasks. This approach can also handle the dynamic changes of the camera poses within the learned relative camera and the robot configurations without requiring data recollection, which is crucial for many real-world applications. Additionally, the continual learning-based approaches developed in Chapter 5 can extend the learned relative calibration space over time and preserve metric accuracy, an essential step towards evolving manufacturing processes and the long-term autonomy of robotic systems. Overall, this study contributes to the advancement of robotic systems and has the potential to significantly impact many industries that use automation and robotics technology.

6.3 Significance of Findings

The developed deep learning-based regression architecture for hand-eye calibration (HEC) in Chapter 3 represents a significant advancement in robotics. By utilising a 3D vision system to automatically detect a single reference point defined by the robot's kinematic chain, the architecture can estimate the camera pose and calibration parameters within seconds in the learned calibration space without data recollection. This approach outperforms classic HEC techniques that require data recollection for each camera pose, demonstrating competitive metric accuracy and superior precision. The ability to quickly and accurately calibrate a robotic system in real-time has implications for a wide range of applications, particularly in dynamic environments where frequent recalibration is necessary to maintain accuracy. Overall, the findings highlight the potential for deep learning-based HEC to enhance the performance of robotic systems and enable more efficient and effective automation in various fields, such as manufacturing processes, assembly lines, quality control, material handling, and inspection applications.

The experimental results in Chapter 4 suggest that the developed deep learning-based hand-eye calibration (HEC) approach performs comparably to the classic approach in a real-world robotic manipulation task. In addition, the developed approach exhibits a significant reduction in computational complexity, as shown by the decreased number of attempts required to obtain calibration parameters and the reduced calibration time compared to the classic HEC approach. These findings are significant because they present the potential of the developed approach to improve the efficiency and accuracy of robotic manipulation tasks. The reduced computational complexity and time needed for calibration can also lead to cost savings and improved productivity in various industries that rely on robotic automation.

The findings in Chapter 5 suggest that the hand-eye calibration problem can be solved effectively by treating it as a time-series problem and updating calibration parameters continuously with changing camera poses. The proposed continual learning-based HEC approach uses a buffer system to incorporate new observations and extend the learned calibration space without requiring complete retraining. This enhances the flexibility and autonomy of the calibration process, allowing the calibration parameters to adapt and improve over time. These results offer a promising solution to the hand-eye calibration problem with potential applications in various fields. For example, accurate hand-eye calibration is essential for improving the performance of robot manipulators, which are widely used in manufacturing and assembly tasks.

6.4 Limitations

These developed hand-eye calibration approaches consider external camera configuration where the camera is placed in an external location. They are not valid for the eye-in-hand configuration in which a camera is attached to one of the robot joints, generally end-effector.

Tsai's [8] hand-eye calibration method is a reference for evaluating the devised hand-eye calibration approaches without introducing absolute errors. This particular approach yields favourable outcomes following multiple data collection iterations. The utilisation of projection error enables assessing the accuracy of the calibration parameters estimated through Tsai's method Tsai's [8], a practical consideration since obtaining absolute calibration parameters in real-world contexts entails the utilisation of costly calibration equipment.

The repeatability score is employed as a comparison method developed approaches with the state-of-the-art deep learning-based and the classic HEC approaches in the real world without absolute error.

Obtaining accurate camera poses with respect to the robot base can be particularly challenging in real-world scenarios. The proposed approach relies on labelled data in the training phase, which can be a challenging and time-consuming task in some application areas. Furthermore, the success of the approach is highly dependent on the accuracy and reliability of the data labelling operation.

Another important assumption of the current approach is that the 3D vision system is accurately calibrated. However, in practice, this system may be prone to errors or malfunctions, which can affect the performance of the proposed approach. The proposed approach has been tested on a limited set of simulated and real-world environments. Therefore, its generalisability and performance in other settings remain to be evaluated.

6.5 Future Works

This section presents potential future works to address the developed approaches' limitations, which are detailed in section 6.4.

6.5.1 Integration with manufacturing robotic systems

The developed deep learning-based hand-eye calibration approach provides a flexible and autonomous calibration paradigm for dynamic robotic manufacturing environments. It enables calibration updates in the learned space and extension of the learned space over time for camera pose changes. Moreover, the developed approach can be used for calibration transfer between the robots. For example, the calibration model of the Universal Robot 3 can be transferred to the Universal Robot 5 by just using a few samples, eliminating the data collection from the beginning. Furthermore, *the developed approach can extend the robot's kinematic chain by adding a tool without specifying the model.* For example, a robot's arm can be extended by holding a hammer, which would enable the robot to manipulate the environment with an extended workspace.

6.5.2 Integration with other robotic systems

The suggested method may be deployed with other robotic systems, such as self-driving cars, drones, or mobile robots, to allow them to carry out tasks in dynamic situations with precise and adaptable hand-eye calibration.

The developed hand-eye calibration approaches can be applied to self-driving, which uses cameras to perceive the environment for making decisions. The position and orientation of the cameras with respect to the car's base can change over time due to factors such as vibrations, wear and tear, or accidental impacts. Thus, the flexible and autonomous hand-eye calibration must be frequently updated to maintain accurate perception and prevent accidents.

The developed deep learning-based hand-eye calibration approach can also be modified for drones, in which the calibration of the camera with respect to its base can be broken because of the temperature or wind when they are operating.

6.5.3 Investigation of different learning paradigms

The developed approach was currently trained in a supervised manner. However, the hand-eye calibration problem can be reformulated using other learning paradigms, such as reinforcement learning and self-supervised learning. The hand-eye calibration problem can be approached using Reinforcement Learning, where the robot acts as an agent, and the environment is defined as the camera space (images) and object space. The agent's goal is to maximise a reward function by taking actions to manipulate an object detected in the camera space using the current hand-eye calibration parameters. The agent controls the robot arm and adjusts the calibration parameters over time to achieve this objective, using feedback from the environment to guide its actions. By formulating the hand-eye calibration problem as a Reinforcement Learning problem, we can develop more flexible and adaptive calibration approaches that can be applied to a wider range of robotic systems and environments.

The variational autoencoder [102] can also be used to learn the distribution of the robot and camera parameter spaces in the encoder part. Then, the decoder part can be used to construct images using encoded features, and the difference between the constructed and real images can be used as a learning signal. This approach could lead to more efficient and flexible calibration methods that adapt to changing environments and hardware configurations.

6.5.4 Exploration of other deep learning architectures

The developed approach employs deep learning-based regression architecture, but other deeply learning architectures can be used to formulate the problem, such as recurrent neural networks. NARX (Nonlinear Autoregressive with exogenous inputs) [103, 104] networks can address the hand-eye calibration problem by learning the relationship between the camera and robot poses and adjusting the calibration parameters in a closed-loop manner. NARX networks are similar to the Multi-Layer Perceptron (MLP) in training, but they are the type of recurrent networks whose outputs are used as inputs through direct connections. It has a prominent performance on nonlinear systems [103]. A feedback mechanism can enable the network to control its outputs, allowing the calibration parameters to be continuously updated as the camera and robot positions change. Furthermore, the ability of NARX networks to remember past observations over time can make them compatible with the continual learning-based approach, eliminating the need for a buffer system. This can lead to a more efficient and adaptable calibration approach in dynamic environments.

Bibliography

[1] X. Fan, X. Wang, and Y. Xiao, “A combined 2d-3d vision system for automatic robot picking,” in *Proceedings of the 2014 International Conference on Advanced Mechatronic Systems*. IEEE, 2014, pp. 513–516.

[2] M. Pena, R. Osorio *et al.*, “Robot vision methodology for assembly manufacturing tasks,” in *Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007)*. IEEE, 2007, pp. 289–294.

[3] N. Herakovic, *Robot vision in industrial assembly and quality control processes*. IN-TECH Open Access Publisher London, UK, 2010.

[4] R. Tsai, “A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[5] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[6] H. Zhang, C. Zhang, W. Yang, and C.-Y. Chen, “Localization and navigation using qr code for mobile robot in indoor environment,” in *2015 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2015, pp. 2501–2506.

[7] J. Abdullah and K. Martinez, “Camera self-calibration for the artoolkit,” in *The First IEEE International Workshop Agumented Reality Toolkit,*. IEEE, 2002, pp. 5–pp.

[8] R. Y. Tsai, R. K. Lenz *et al.*, “A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration,” *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 345–358, 1989.

[9] J. C. Chou and M. Kamel, “Finding the position and orientation of a sensor on a robot manipulator using quaternions,” *The international journal of robotics research*, vol. 10, no. 3, pp. 240–254, 1991.

[10] N. Andreff, R. Horaud, and B. Espiau, “Robot hand-eye calibration using structure-from-motion,” *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 228–248, 2001.

[11] R. Horaud and F. Dornaika, “Hand-eye calibration,” *The international journal of robotics research*, vol. 14, no. 3, pp. 195–210, 1995.

[12] K. Daniilidis, “Hand-eye calibration using dual quaternions,” *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[13] H. Zhuang, Z. S. Roth, and R. Sudhakar, “Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form ax= yb,” *IEEE Transactions on Robotics and Automation*, vol. 10, no. 4, pp. 549–554, 1994.

[14] F. Dornaika and R. Horaud, “Simultaneous robot-world and hand-eye calibration,” *IEEE transactions on Robotics and Automation*, vol. 14, no. 4, pp. 617–622, 1998.

[15] M. Shah, “Solving the robot-world/hand-eye calibration problem using the kronecker product,” *Journal of Mechanisms and Robotics*, vol. 5, no. 3, p. 031007, 2013.

[16] A. Tabb and K. M. Ahmad Yousef, “Solving the robot-world hand-eye (s) calibration problem with iterative methods,” *Machine Vision and Applications*, vol. 28, no. 5-6, pp. 569–590, 2017.

[17] Z. Zhao, “Simultaneous robot-world and hand-eye calibration by the alternative linear programming,” *Pattern Recognition Letters*, vol. 127, pp. 174–180, 2019.

[18] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, “Camera-to-robot pose estimation from a single image,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9426–9432.

[19] E. Valassakis, K. Dreczkowski, and E. Johns, “Learning eye-in-hand camera calibration from a single image,” in *Conference on Robot Learning*. PMLR, 2022, pp. 1336–1346.

[20] J. Jiang, X. Luo, Q. Luo, L. Qiao, and M. Li, “An overview of hand-eye calibration,” *The International Journal of Advanced Manufacturing Technology*, pp. 1–21, 2021.

[21] C. Fitzgerald, “Developing baxter,” in *2013 IEEE conference on technologies for practical robot applications (TePRA)*. IEEE, 2013, pp. 1–6.

[22] R. Bloss, “Collaborative robots are rapidly providing major improvements in productivity, safety, programing ease, portability and cost while addressing many new applications,” *Industrial Robot: An International Journal*, vol. 43, no. 5, pp. 463–468, 2016.

[23] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[24] UR Polyscope. [Online]. Available: https://www.universal-robots.com/

[25] J. W. Eaton, D. Bateman, and S. Hauberg, “Gnu octave,” *GNU Octave*, 2013.

[26] P. H. Winston and B. K. Horn, “Lisp,” 1986.

[27] A.-M. Hellmund, S. Wirges, Ö. Ş. Taş, C. Bandera, and N. O. Salscheider, “Robot operating system: A modular software framework for automated driving,” in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1564–1570.

[28] S. Inc., “Zed,” https://www.stereolabs.com/zed, 2022, last accessed 10 February 2022.

[29] J. Diebel, “Representing attitude: Euler angles, unit quaternions, and rotation vectors,” *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[30] P. Flores, *Euler Angles, Bryant Angles and Euler Parameters*. Cham: Springer International Publishing, 2015, pp. 15–22. [Online]. Available: https://doi.org/10.1007/978-3-319-16190-7_4

[31] Y.-B. Jia, “Quaternions and rotations,” *Com S*, vol. 477, no. 577, p. 15, 2008.

[32] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, “On the continuity of rotation representations in neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5745–5753.

[33] V. Peretroukhin, M. Giamou, D. M. Rosen, W. N. Greene, N. Roy, and J. Kelly, “A smooth representation of belief over so (3) for deep rotation learning with uncertainty,” *arXiv preprint arXiv:2006.01031*, 2020.

[34] T. Luhmann, C. Fraser, and H.-G. Maas, “Sensor modelling and camera calibration for close-range photogrammetry,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 115, pp. 37–46, 2016.

[35] I. Sobel, “On calibrating computer controlled cameras for perceiving 3-d scenes,” *Artificial intelligence*, vol. 5, no. 2, pp. 185–198, 1974.

[36] K. M. Dawson-Howe and D. Vernon, “Simple pinhole camera calibration,” *International Journal of Imaging Systems and Technology*, vol. 5, no. 1, pp. 1–6, 1994.

[37] L. Huang, Q. Zhang, and A. Asundi, "Flexible camera calibration using not-measured imperfect target," *Applied optics*, vol. 52, no. 25, pp. 6278–6286, 2013.

[38] W. Qi, F. Li, and L. Zhenzhong, "Review on camera calibration," in *2010 Chinese Control and Decision Conference*. IEEE, 2010, pp. 3354–3358.

[39] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 3400–3407.

[40] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," *International journal of computer vision*, vol. 8, no. 2, pp. 123–151, 1992.

[41] F. Li, H. Sekkati, J. Deglint, C. Scharfenberger, M. Lamm, D. Clausi, J. Zelek, and A. Wong, "Simultaneous projector-camera self-calibration for three-dimensional reconstruction and projection mapping," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 74–83, 2017.

[42] L. S. Ginani and J. M. S. Motta, "Theoretical and practical aspects of robot calibration with experimental verification," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 33, no. 1, pp. 15–21, 2011.

[43] K. Okamura and F. C. Park, "Kinematic calibration using the product of exponentials formula," *Robotica*, vol. 14, no. 4, pp. 415–421, 1996.

[44] G. Li, F. Zhang, Y. Fu, and S. Wang, "Kinematic calibration of serial robot using dual quaternions," *Industrial Robot: the international journal of robotics research and application*, 2019.

[45] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," 1955.

[46] S. Hayati, K. Tso, and G. Roston, "Robot geometry calibration," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. IEEE, 1988, pp. 947–951.

[47] A. Nubiola and I. A. Bonev, "Absolute calibration of an abb irb 1600 robot using a laser tracker," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 1, pp. 236–245, 2013.

[48] M. Ikits and J. M. Hollerbach, "Kinematic calibration using a plane constraint," in *Proceedings of International Conference on Robotics and Automation*, vol. 4. IEEE, 1997, pp. 3191–3196.

[49] H. Zhuang, S. H. Motaghedi, and Z. S. Roth, "Robot calibration with planar constraints," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 1. IEEE, 1999, pp. 805–810.

[50] M. Švaco, B. Šekoranja, F. Šuligoj, and B. Jerbić, “Calibration of an industrial robot using a stereo vision system,” *Procedia Engineering*, vol. 69, pp. 459–463, 2014.

[51] A. Roncone, M. Hoffmann, U. Pattacini, and G. Metta, “Automatic kinematic chain calibration using artificial skin: self-touch in the icub humanoid robot,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2305–2312.

[52] M. Hersch, E. Sauser, and A. Billard, “Online learning of the body schema,” *International Journal of Humanoid Robotics*, vol. 5, no. 02, pp. 161–181, 2008.

[53] K. Stepanova, T. Pajdla, and M. Hoffmann, “Robot self-calibration using multiple kinematic chains—a simulation study on the icub humanoid robot,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1900–1907, 2019.

[54] K. Stepanova, J. Rozlivek, F. Puciow, P. Krsek, T. Pajdla, and M. Hoffmann, “Automatic self-contained calibration of an industrial dual-arm robot with cameras using self-contact, planar constraints, and self-observation,” *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102250, 2022.

[55] Z. Zhao, “Hand-eye calibration using convex optimization,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2947–2952.

[56] A. Li, L. Wang, and D. Wu, “Simultaneous robot-world and hand-eye calibration using dual-quaternions and kronecker product,” *International Journal of Physical Sciences*, vol. 5, no. 10, pp. 1530–1536, 2010.

[57] X. Zhi and S. Schwertfeger, “Simultaneous hand-eye calibration and reconstruction,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1470–1477.

[58] I. Ali, O. Suominen, A. Gotchev, and E. R. Morales, “Methods for simultaneous robot-world-hand–eye calibration: A comparative study,” *Sensors*, vol. 19, no. 12, p. 2837, 2019.

[59] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[60] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, “Bundle adjustment—a modern synthesis,” in *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*. Springer, 2000, pp. 298–372.

[61] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[62] J. Lambrecht, "Robust few-shot pose estimation of articulated robots using monocular cameras and deep-learning-based keypoint detection," in *2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA).* IEEE, 2019, pp. 136–141.

[63] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[64] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[65] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision*, vol. 13, no. 2, pp. 119–152, 1994.

[66] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[68] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[69] S. T. Grossberg, *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control.* Springer Science & Business Media, 2012, vol. 70.

[70] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, pp. 1–13, 2022.

[71] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[72] R. Aljundi, P. Chakravarty, and T. Tuytelaars, “Expert gate: Lifelong learning with a network of experts,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3366–3375.

[73] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, “Pathnet: Evolution channels gradient descent in super neural networks,” *arXiv preprint arXiv:1701.08734*, 2017.

[74] A. Mallya and S. Lazebnik, “Packnet: Adding multiple tasks to a single network by iterative pruning,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[75] T. L. Hayes, N. D. Cahill, and C. Kanan, “Memory efficient experience replay for streaming learning,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9769–9776.

[76] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: Incremental classifier and representation learning,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[77] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, “Experience replay for continual learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[78] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[79] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov, “Rotate your networks: Better weight consolidation and less catastrophic forgetting,” in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 2262–2268.

[80] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 139–154.

[81] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, “Class-incremental learning via deep model consolidation,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1131–1140.

[82] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[83] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[84] S. Wang, Z. Laskar, I. Melekhov, X. Li, and J. Kannala, "Continual learning for image-based camera localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3252–3262.

[85] A. Chrysakis and M.-F. Moens, "Online continual learning from imbalanced data," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1952–1961. [Online]. Available: https://proceedings.mlr.press/v119/chrysakis20a.html

[86] K. Pauwels and D. Kragic, "Integrated on-line robot-camera calibration and object pose estimation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2332–2339.

[87] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[88] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[89] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[90] S. R. Company, "Modular grasper," https://modular-grasper.readthedocs.io/en/latest/user_guide/1_introduction/, 2022, last accessed 10 February 2022.

[91] T. Foote, "tf: The transform library," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2013, pp. 1–6.

[92] K. S. Chen, "Application of the iso 9283 standard to test repeatability of the baxter robot," 2015.

[93] O. Saha and P. Dasgupta, "A comprehensive survey of recent trends in cloud robotics architectures and applications," *Robotics*, vol. 7, no. 3, p. 47, 2018.

[94] O. Bahadir, J. P. Siebert, and G. Aragon-Camarasa, "A deep learning-based hand-eye calibration approach using a single reference point on a robot manipulator," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2022, pp. 1109–1114.

[95] I. A. Sucan and S. Chitta, “Moveit!” 2013.

[96] A. Mordvintsev and K. Abid, “Opencv-python tutorials documentation,” *Obtenido de https://media. readthedocs. org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals. pdf*, 2014.

[97] K. Koide and E. Menegatti, “General hand–eye calibration based on reprojection error minimization,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1021–1028, 2019.

[98] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O’Reilly Media, Inc.", 2008.

[99] “ZED Stereo Camera.” [Online]. Available: https://www.stereolabs.com/zed/

[100] Stanford Artificial Intelligence Laboratory et al., “Robotic operating system.” [Online]. Available: https://www.ros.org

[101] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural networks*, vol. 113, pp. 54–71, 2019.

[102] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[103] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, “Learning long-term dependencies in narx recurrent neural networks,” *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1329–1338, 1996.

[104] H. Xie, H. Tang, and Y.-H. Liao, “Time series prediction based on narx neural networks: An advanced approach,” in *2009 International conference on machine learning and cybernetics*, vol. 3. IEEE, 2009, pp. 1275–1279.