A Novel Self-recoverable Hand-eye Calibration Technique for Vision-guided Robot

Enebuse, Ikenna Stanley

Award date:
2024

Awarding institution:
Coventry University

A Novel Self-recoverable Hand-eye Calibration Technique for Vision-guided Robot

By

Ikenna Stanley Enebuse

March 2023

A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy

Certificate of Ethical Approval

Applicant:	Ikenna Enebuse
Project Title:	Vision Based Robotic Guidance and Monitoring Using 3D Sensor and Industrial Internet of Things

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:	01 Feb 2021
Project Reference Number:	P116330

Dedication

I dedicate this thesis to my wife and incoming baby.

Acknowledgment

I am grateful to my supervisors, Babul KSM Kader Ibrahim, Ranveer Matharu, Mathias Foo, and Hafiz Ahmed. This work would not have been possible without their guidance, encouragement, recommendations, and revisions.

I am deeply grateful to the administrators of the Doctoral College, in particular, Dr. Heather Sears and Carolyn Wynne for their unwavering support during the trying times of my PhD journey.

I would also like to express my gratitude to Professor Trevor Toman, Alex Gkantsios, and Rizwan Tai of The Advanced Institute of Manufacturing and Engineering for their technical support and resources provided in the design and setup of my experiments.

To my lovely wife Chileziem. You were my greatest cornerstone and my best cheerleader during the 3 years of this journey. You were the reason I needed to keep pushing on, even at my lowest point. I am eternally grateful for your support which never ceases to cheer me up.

To my parents, Chief Mr. and Mrs. Enebuse. I thank you for your moral support during this journey. Their words of encouragement and prayers kept me going. I also would like to extend my gratitude to my siblings, Collins, Chioma, and Ugochukwu for their continued support, and my second parents, Mr. and Mrs. Greg Enyimiri for their constant words of encouragement. You all gave me all the love and prayers that made this journey successful. You will be forever cherished.

Finally, I wish to gratefully acknowledge the support of the Doctoral College for the full PhD studentship awarded to me that enabled the successful completion of the program without any financial difficulty.

Abstract

Long-term operability is essential to reap the benefits of cost-effective industrial automation involving vision-guided robots (VGR). However, it is essential that the accuracy of the robot during the period of operation is uncompromised. The hand-eye calibration accuracy of a VGR plays a huge role in the accuracy of the robot during operation as it enables the proper perception of the environment in which a VGR operates. Hand-eye calibration of VGRs is typically done offline. However, the calibrated parameters may change during the operation, which requires recalibration to maintain operational accuracy, therefore taking the robot offline resulting in operational downtime and lost revenue. As such, it becomes imperative to integrate a system that enables the robot to maintain its accuracy by adapting to changes in its environment. To tackle this challenge, different hand-eye calibration techniques for vision-guided robots were reviewed to assess and evaluate their strengths and weaknesses in an operational environment, as well as the practicality of different types of calibration targets. Furthermore, experimental validation of the different factors that affect the accuracy of hand-eye calibration was carried out using six commonly used algorithms in the industry. To minimise the problems common with these algorithms, a self-recoverable hand-eye calibration scheme based on a hybrid filter is proposed. This algorithm runs simultaneously with the robot's operation while ensuring the robot recovers from any changes in the calibration accuracy. The hybrid filter is based on a combination of Kalman and particle filters with optimisations on particle transition and genetic algorithm-based resampling as a proposed modification. To reduce the dimension of the particle filter for improved performance, the particle filter, which estimates the rotation parameter,

is coupled with a Kalman filter with iterated state update to estimate the translation parameter. While the proposed particle transition scheme handles the problem of sample impoverishment common with the standard particle filter, the genetic algorithm resampling method handles the problem of degeneracy without inducing sample impoverishment. A gradient descent estimator, which has a much simpler implementation than the Kalman filter was also explored for the translation parameter estimate. Experimentation with the algorithm shows that it is able to actively detect and recover from errors due to changes in the calibration parameter. Comprehensive experimental results with a UR5e robot arm show that even in the offline scenario, the proposed method outperforms the offline-specific calibration method, highlighting the suitability of the developed method for online and offline calibration.

Table of Contents

List of Figures

List of Tables

Chapter 1

General Introduction

1.1 Background and motivation

Robots, a term first used by Czech writer Karel Čapek in 1920 are machines designed to enable the autonomous or semi-autonomous execution of tasks. They are increasingly being used in the manufacturing industries for tasks such as assembly [1], welding [2], and painting [3] where they can operate continuously, and thus their suitability for repetitive tasks. They can also be found in the health industries for tasks such as surgery, as in Robot-Assisted Surgery (RAS), rehabilitation, and other medical procedures to increase accuracy and efficiency [4]. Other applications include crop planting and harvesting in the agricultural industries [5] as well as the use in theme parks and special effects in TV shows in the entertainment industries [6].

Robots can be grouped into several types as shown in Figure 1.1. These include industrial robots and service robots. While industrial robots are designed for manufacturing and other industrial-related and high-power tasks which are usually deployed in dangerous environments, service robots, on the other hand, are designed for routines

FIGURE 1.1: Different types of robots. (A) Industrial assembly robot[7]. (B) Foodservice robot[8]. (C) Home cleaning robot[9]

such as cleaning, security, and delivery. Service robots can further be grouped into personal and professional service robots based on their intended use. Personal service robots are usually employed in non-commercial settings by unskilled persons to complete personal tasks such as mobile chairs for the elderly for commuting, cleaning robots for the home, robotic toys, health and fitness robots, etc. Professional service robots on the other hand are used in commercial settings to provide professional services. These may include public road cleaning, security, and delivery services.

Robots by themselves have high repeatability but are generally poor when it comes to absolute accuracy. This can be attributed to differences between the ideal and the actual kinematic model used in the design of the robot parameters [10]. Other sources of error can be attributed to the mechanical characteristics of the robot components such as stiffness, backlash, and elasticity. To improve the accuracy of the robot, it is usually necessary to do a kinematic calibration of the robot [11]. This adjusts the designed parameters of the robot to be within a close range of the ideal parameters such that the resulting accuracy of the robot is increased. With the continued need for improved efficiency and productivity in the operations of businesses and industries, the need to improve upon the accuracy and efficiency of robots continues to grow. One way this has been

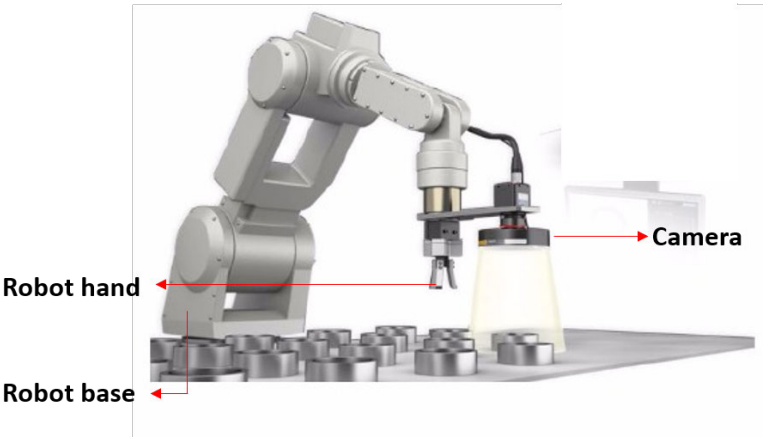


FIGURE 1.2: Vision-guided robot for pick-and-place application [12].

achieved is to increase the autonomy of robots in their operation. The increased autonomy is based on allowing the robot to make informed decisions on its actions based on feedback from its immediate environment. This feedback can be obtained from a vision sensor or camera mounted on the robot to allow the robot accurately perceive the environment. These are called vision-guided robots (VGR) as shown in Figure 1.2. With their ability to perceive and understand their environment, VGRs play a major role in shaping the future of automation and beyond with the end result of improved efficiency, increased productivity, reduced costs, and improvement in the quality of products and services.

VGRs are applied in a growing number of industries today. In the manufacturing industry, they are increasingly being used in pick and place applications, assembly operations, and other material handling tasks, where with the aid of the camera, the robot is able to track the work object and/or navigate its environment such that appropriate real-time decisions based on visual information can be taken to speed up and improve the production process.

Similarly, in the healthcare industry, VGRs are increasingly being used in applications such as RAS or robotic scrub nurses (RSN) for

operations such as surgery or physical therapy [4]. In these applications, feedback from the camera is used to monitor the movements and position of the patient and/or surgical tools to provide real-time feedback to guide the healthcare providers in a way that improves the accuracy and efficiency of the procedure while minimizing the risk of medical errors.

In the service industry, applications of VGR can be found in autonomous security systems that monitor and analyzes complex environments such as airports, shopping centers, and public spaces to improve security [13]. These systems can detect and respond to or report anomalies in the environment. Other applications can be found in self-driving cars where the implementation of VGR enables the perception of conditions in the environment to safely and adequately guide the movement of a vehicle.

In addition to their use in various industries, VGRs are also being used in consumer applications like drones [14], home, and logistic robots [15], etc. These systems use feedback from the integrated cameras integrated with algorithms and/or machine learning techniques to enable them to undertake a wide range of tasks from monitoring homes to package delivery.

Advances in VGR are primarily driven by the growing need for automation, improvements in computer vision algorithms, as well as the improved reliability and accuracy of hardware components that can foster the automation of complex tasks. Furthermore, the advancement in the Industrial Internet of Things (IIoT) has provided a new level of connectivity and data-sharing platforms that enables the development of highly sophisticated and interconnected systems that can interact and cooperate with one another.

The importance of VGR can be summarised as

1. **Increased Accuracy and Efficiency:** VGRs enable accurate perception of the immediate environment through the use of integrated cameras and image processing algorithms, enabling them to carry out complex and highly demanding tasks with high levels of accuracy and efficiency.

2. **Improved Quality:** VGRs can inspect and sort products with a high level of accuracy, such as in pick and place applications, reducing the risk of defects and improving the overall quality of the finished product.

3. **Increased Safety:** The autonomous nature of VGRs allows for their deployment in dangerous or repetitive tasks, which would otherwise risk injury to human workers. VGRs can also be used for the monitoring of hazardous environments which reduces the risk of accidents and other incidents.

4. **Flexibility:** Due to the fact that VGRs can accurately perceive and understand their immediate environment, they can easily adapt to unforeseen changes in the environment. Consequently, they can be programmed to perform a wide range of tasks depending on needs and requirements, making them highly flexible and adaptable.

5. **Cost Savings:** The use of VGRs to maximise the automation of highly complex and intensive tasks reduces the need for human elements in the production process. This results in a significant reduction in the cost of defective products as well as labor.

6. **Human augmentation:** VGRs can assist human workers by providing real-time feedback and guidance, improving the quality and accuracy of their work, and helping to reduce the risk of medical errors. They have also been employed successfully in the development of exoskeletal systems that provide additional support to the wearer's muscles and joints, enhancing their strength and endurance, as well as in prosthetics to replace or augment missing or impaired limbs, restoring mobility.

7. **Advancement of Automation:** With the integration of visual feedback, robots acquire an increased level of intelligence and capability which greatly improves the level of operational autonomy with increased precision and reliability.

During the operation of a robot, it may need to interact with or manipulate objects within its workspace. The coordinates of these objects are typically fed into the robot's program which allows the robot to accurately locate them. The coordinates fed to the robot are described relative to a reference frame. A convenient reference frame is one that remains fixed with the robot and is usually set as the robot's base frame. Hence, the position of every object that the robot has to interact with is known relative to the robot-based frame and preprogrammed into the robot's program. With the integration of a camera in a VGR, it becomes possible for the robot to obtain the position of the objects directly from the camera's view during operation, thus increasing the flexibility of the robot. This can be achieved through 2D to 3D mapping techniques such as Perspective-n-point [16], structure from motion [17], deep learning [18], etc. that can help in the localization of an object in 3D space from a 2D image. The position of the object retrieved using any of these techniques is

referenced from the optical center of the camera. Therefore, for the robot to obtain the position of the object viewed by the camera based on its own reference, it must obtain the position of the camera based on its own reference. The process of localising the camera of a vision-guided robot based on the robot's reference is called hand-eye calibration. This task cannot be accomplished directly for several reasons. Firstly, the frames from which the measurements are to be taken are unreachable as they are within the camera or the robot's links. Secondly, the measurement path may be obstructed by the geometry of the sensor, the robot, or other parts of the assembly. For these reasons, a lot of research has gone into the accurate estimation of these measurements.

With hand-eye calibration, the robot is able to obtain accurate visual feedback of its workspace from the integrated camera for adequate servo control or manipulation of objects. In a typical setup, hand-eye calibration is performed offline before the robot is put into operation. The calibration parameters are then fed into the robot's program for proper reference transformation during operation. This allows the robot to make sense of the visual information from the camera. However, this comes with some drawbacks. Because these calibration parameters are preprogrammed into the robot's program, it is expected to be constant irrespective of the condition in which the robot operates, as any change in their values would affect the accuracy of the robot's operation. In the industry today, this is an ever-present challenge as the prolonged usage of the robot, abrupt motion of the robot, wear in the camera fixtures, etc. could cause the pose of the camera to change and hence the calibration parameters.

This is usually solved by periodic hand-eye recalibration which requires taking the robot out of service, resulting in operational downtime. The work in this thesis aims to ensure a long-term guarantee of the accuracy of the robot during operation without the need for recalibration, this would enable the VGR to recover from changes in the calibration parameters to restore the accuracy of the robot. Furthermore, this calibration technique should be capable of online operation so that it can be operated in real-time during the robot's operation.

1.2 Aims and objectives

The aim of this research is the development of a highly accurate self-recoverable hand-eye calibration algorithm for vision-guided robots. This algorithm is required to allow a vision-guided robot to maintain its accuracy for a sustained period of time during operation without the needed recalibration procedure. To achieve this aim, the following objectives of this research have been laid out.

1. To review the current hand-eye calibration techniques for vision-guided robots.
2. To assess the accuracy of current hand-eye calibration techniques through simulation and experimental studies.
3. To develop an online calibration technique capable of maintaining its accuracy.
4. To implement and validate a newly proposed technique via experimental study.

1.3 Contributions

The successful operation of a VGR requires that the accuracy of the robot-camera setup is not compromised. This thesis aims to develop a self-recoverable hand-eye calibration technique for a VGR to ensure the long-term accuracy of the robot's operation in an uncertain environment. This is achieved using a guided particle hybrid filter with a genetic algorithm-based resampling technique. The performance of the algorithm will be validated against currently existing algorithms using a UR5e Universal robot and Azure Kinect camera.

The contributions of the thesis are listed as follows.

1. **A Comparative Review of Hand-Eye Calibration Techniques for Vision Guided Robots:** This paper gives a general insight into the strengths and weaknesses of different hand-eye calibration algorithms available to academics and industrial practitioners to make an informed design decision, as well as incite possible areas of research based on the identified challenges. *Enebuse, I., Foo, M., Ibrahim, B. K. K., Ahmed, H., Supmak, F., & Eyobu, O. S. (2021). A comparative review of hand-eye calibration techniques for vision-guided robots. IEEE Access, doi: 10.1109/ACCESS.2021.3104514 (**Published**)*

2. **Accuracy evaluation of hand-eye calibration techniques for vision-guided robots:** This paper provides new experimentally validated insights on the factors that affect the accuracy of hand-eye calibration for vision-guided robots. It explored and evaluated scenarios that have previously been overlooked especially for benchmarking the performance of these algorithms

and showed the role these scenarios play in assessing the accuracy of the algorithms.
*Enebuse, I., Ibrahim, B. K. K., Foo, Ranveer S. M, M., Ahmed. (2022). Accuracy evaluation of hand-eye calibration techniques for vision-guided robots. PLOS ONE, doi:10.1371/journal.pone.0273261 (**Published**)*

3. **An Accuracy Assessment of Hand-Eye Calibration Techniques in Uncertain Environments for Vision-Guided Robots:** This paper examines how the accuracy of vision-guided robots are affected by uncertain environmental condition. As the majority of hand-eye calibration techniques employ passive hand-eye calibration that requires frequent recalibration operations, this paper goes further to propose the use of active hand-eye calibration as a solution for these issues and the expected challenges to be resolved.
*Enebuse, I., Ibrahim, B. K. K., Foo, Ranveer S. M, M., Ahmed. (2023). An Accuracy Assessment of Hand-Eye Calibration Techniques in Uncertain Environments for Vision Guided Robots. Proceedings of International Conference on Mechatronics (**Accepted**)*

4. **A Self-recoverable Hand-eye Calibration for Vision Guided Robots using a Guided Particle Hybrid Filter and Genetic Algorithm-based Resampling:** This paper proposes a novel hand-eye calibration technique based on a Guided Particle Hybrid Filter and Genetic Algorithm-based Resampling. This technique enables the self-calibration of vision-guided robots which allows them to be operated without prior calibration. The algorithm also allows the VGR to automatically detect and recover from faults sustained in an uncertain environment while in operation thus eliminating the need for recalibration which

would otherwise result in operational downtime and lost revenue.

*Enebuse, I., Ibrahim, B. K. K., Foo, Ranveer S. M, M., Ahmed. (2023). A Self-recoverable Hand-eye Calibration for Vision Guided Robots using a Guided Particle Hybrid Filter and Genetic Algorithm-based Resampling. IEEE Transaction on Instrumentation and Measurements (**Submitted**)*

1.4 Outline of the thesis

The rest of this thesis is organised as follows: Chapter 2, presents a general overview of hand-eye calibration techniques and compared the strengths and weaknesses of different techniques for hand-eye calibration which was followed by an overview of the factors that affect hand-eye calibration. Furthermore, different types of calibration targets were examined as well as their practicality for hand-eye calibration. Chapter 3 focuses on the analyses of the factors that affect the accuracy of hand-eye calibration through simulation and experimental validation using a UR5e Universal robot and Azure Kinect camera, while Chapter 4 deals with the evaluation of the performance of hand-eye calibration techniques in uncertain environments with an overview of considerations for active hand-eye calibration to improve the robustness of hand-eye calibration. Chapter 5 describes the development of a new algorithm for hand-eye calibration based on a guided particle hybrid filter, as well as a comparative assessment of the algorithm with other algorithms. Finally, in Chapter 6, a general conclusion of the work in this thesis is given as well as a possible direction for future work in other to advance the algorithm developed.

Chapter 2

Overview of hand-eye calibration

2.1 Introduction

Industrial robots have been around for decades, first gaining popularity in the automotive industry [19]. Automotive plants were suitable for early industrial robots because the tasks in these plants involve a high level of repeatability, large payloads, and moderate speeds. Robots are also being used in a growing number of sectors, such as chicken deboning in the food industry [20–22], drug manufacturing in the pharmaceutical industry [23, 24], and aircraft engine construction in the aerospace industry [25–27]. According to the International Federation of Robotics (IFR) estimations, over 690 thousand new industrial robots will be deployed globally in 2025 [28], and vision systems are now becoming a major component of many industrial robots as they improve the capabilities of robots in operation. For example, vision-guided robots can allow for variability in the positioning of work objects or deviations in the programmed pathway without breaking the production flow [29–31].

Emerging applications demand that industrial robots not only be faster, but also be able to accurately identify and find parts that are

randomly placed on moving conveyors, containers, or on pallets [32–34]. Machine vision systems, which have been around for decades, are now being used in conjunction with robotics to aid automation systems in the processing of such components [35, 36].

VGR is rapidly becoming a key enabler for the automation of a broad range of processes in a wide range of industries. A typical vision-guided robot has a camera attached close to the robot hand or gripper with which it can perceive the work environment (Figure 1.2). The two major areas in the field of computer vision are 2D and 3D technologies. In a flat plane relative to the robot, a 2D VGR device processes the captured images of randomly placed pieces. These images are 2D projections of the 3D spatial pieces, which results in the loss of depth information. A 3D VGR device, on the other hand, can process parts that are randomly positioned in three dimensions (i.e., X-Y-Z) and can also accurately determine the 3D orientation of each part. In practice, 2D machine vision is typically accomplished using a digital camera and software that analyses a digital image of the part's 2D position and orientation in preparation for robotic handling or processing [37, 38]. The 3D vision system on the other hand uses sensors like laser displacement, structured light, and stereo camera capable of generating a point cloud representation of a surface in the 3D space [39–41]. The point cloud enables the spatial reconstruction of a 3D scene, which facilitates the handling of a wide variety of complex objects in a challenging environment, thereby enhancing the capabilities of robots for vision-guided applications. One particular advantage is being able to pick up objects placed on a surface with irregular height, which would be difficult for the 2D vision system.

Applications of vision-guided robots include part assembly [1], bin

picking [42], inspection [43] etc. These robots can either have the camera mounted in a fixed position with a fixed field of view (eye-to-hand configuration) or have the camera mounted on the hand of the robot (eye-in-hand configuration) so that new images can be acquired by changing the point of view of the camera. However, the robot can only perceive the 3D world based on its own base frame. In order for the robot to obtain an accurate estimate of the 3D position and orientation of a part relative to its own base within the work volume, it is necessary to know the relative position and orientation between the hand and the robot base, between the camera and the hand, and between the object and the camera. These three tasks require the calibration of robot [10, 11, 44], camera [45, 46], and robot hand-to-camera (hand-eye) [47, 48]. Robot calibration is needed because, despite the fact that robots have very good repeatability, they are poor when it comes to absolute accuracy, due to inherent differences between the ideal and actual kinematic parameters. Camera intrinsic calibration is required to ensure that the images captured are of accurate dimensions and free of lens distortion, which would otherwise introduce errors in the measurement estimates being fed back to the robot during operation. Hand-eye calibration ensures that the measurements made by the camera are converted to the reference used by the robot for measurement. The focus of this review is on hand-eye calibration and its associated challenges to robotic vision system.

The rest of the chapter is organised as follows. Section 2.2 introduces the problem of hand-eye calibration, and Section 2.3 discusses the different hand-eye calibration algorithms. A comparative analysis of calibration target is given in Section 2.4, while common challenges

associated with hand-eye calibration is presented in Section 2.5. Finally, a conclusion is given in Section 2.6.

2.2 Hand-eye calibration

The perception of the environment by the robot can be accomplished using a camera. This enables the navigation and manipulation of objects in an unknown and dynamic environment. This vision system involves the perspective projection and mapping of a 3D world coordinate point onto a 2D image plane, which can be achieved using a pinhole camera model [49] as shown in Figure 2.1.

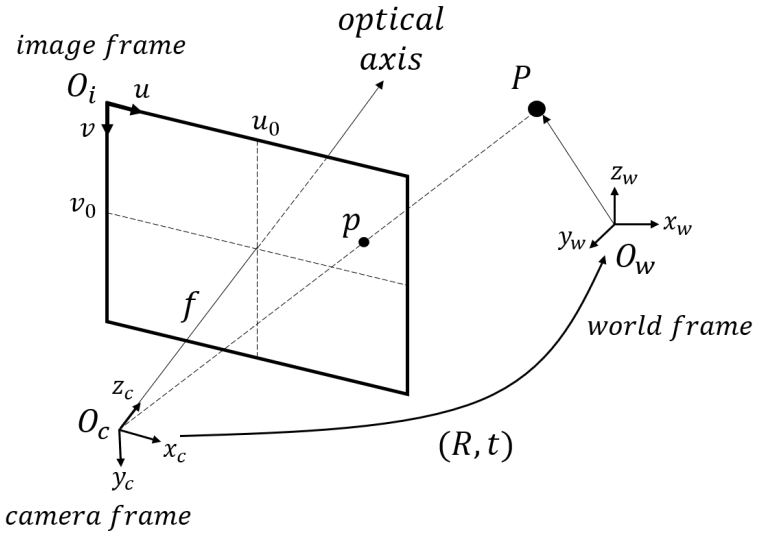


FIGURE 2.1: Pinhole camera model

From Figure 2.1, an optical ray passing through a 3D world point P through the optical center O_c intersects the image plane at a point p located at a distance of f (focal length) from the optical center. To obtain the point p in the image plane $O_i(u, v)$, the world coordinate points O_w first have to be transformed to the camera coordinate at O_c. This is achieved using the transformation Equation (2.1). From Equation (2.1), the camera coordinate points $P^c = (x_c, y_c, z_c)$ are

realised from world coordinate points $P^w = (x_w, y_w, z_w)$ using the rigid body homogeneous transformation matrix H_w^c

$$\begin{pmatrix} P^c \\ 1 \end{pmatrix} = H_w^c \begin{pmatrix} P^w \\ 1 \end{pmatrix}, \tag{2.1}$$

Equation (2.1) can also be expressed as

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \begin{pmatrix} R_w^c & t_w^c \\ 0_{1\times 3}^T & 1 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}, \tag{2.2}$$

where R_w^c and t_w^c denote rotation and translation, respectively, from the world to camera coordinate frames. These parameters are regarded as the extrinsic parameters of the camera. The projection of the points in the camera coordinate onto the image plane based on the pinhole camera model is given by Equation (2.3).

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z_c} \begin{pmatrix} x_c \\ y_c \end{pmatrix}, \tag{2.3}$$

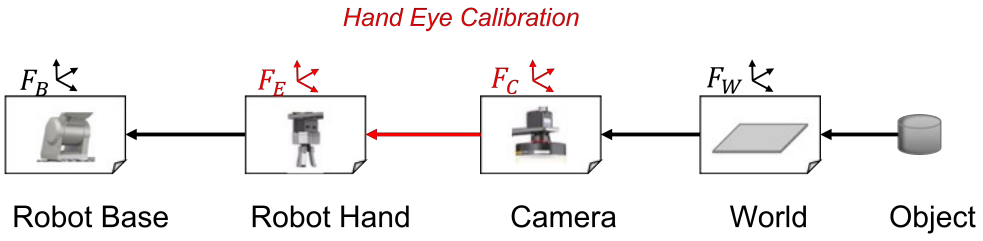


FIGURE 2.2: Relationships between component frames for vision-guided robot.

The task of computing the relative 3D position and orientation between the camera and the robot hand in an eye-on-hand configuration, where the camera is rigidly attached to the robot hand, is known as hand-eye calibration. More specifically, this is the task

of computing the relative rotation and translation (homogeneous transformation) between two coordinate frames, one centered at the camera lens center, and the other at the robot hand. Figure 2.2 shows the relationships between the different component's frames of a vision-guided robot operation. To ensure easy operation of the robot, all commands to the robot are referenced to the robot base frame. Hence, for a complete identification of the object based on the robot base frame, all the relationships must be obtained. While the relationship between the robot base and the robot hand can be realised from the robot kinematic model, the relationship between the camera and the world (see Equation (2.1)) can be obtained from camera calibration. This results in the relationship between the camera and the robot hand needing to be computed. This relationship cannot be measured directly because [47]

1. the measurement path may be obstructed by the geometry of the sensor, the robot, or other parts of the system
2. the hand and camera frames are unreachable. The camera frame is unreachable because it is the intersection of various link axes while the camera frame is unreachable because its origin is at the focus point inside the camera.

Since direct measurements are difficult, other approaches have been investigated to solve the problem. Earlier approaches used non-linear optimisation of a model that coupled the robot forward kinematics with the hand-eye system [50]. These techniques are quite expensive computationally and require the estimation of a large number of variables. In view of that, the most common technique used [47, 51–53] is based on solving the homogeneous transform equation according to Equation (2.4), where $A_{c_2}^{c_1}$ and $B_{h_2}^{h_1}$ are the homogeneous transform

matrices for the motion of the camera and robot hand, respectively, between two positions 1 and 2, and X_h^c is the required robot hand to camera homogeneous transform. If the position and orientation of the hand are known, the position and orientation of the camera can be simply computed, and vice-versa. The object can then be located with respect to the robot base and locating information from different views can be fused. The first challenge encountered during hand-eye calibration is usually the estimation of the pose of the camera relative to the world as the hand pose can easily be acquired from the robot's forward kinematic chain. Depending on how the camera pose is estimated, hand-eye calibration can be regarded as either target-based or targetless. Target-based hand-eye calibration takes advantage of specially made visual features of known dimensions called calibration objects or target - whose origin is set as the origin of the world frame - to estimate the pose of the camera using special algorithms like the Perspective-n-Point [16]. Targetless hand-eye calibration without a calibration target uses techniques such as in structure from motion [54, 55], tool motion tracking [4] etc, to estimate the pose of the camera with respect to the world. These methods can prove useful when taking the size and weight of the calibration object into consideration as well as the size of the workspace for the robot motion. These considerations for a calibration object usually come into play when there is a strict limitation of the payload of a mobile robot such as in space application or sterility of the setup in medical applications. In this review, only techniques based on target-based hand-eye calibration are considered.

Furthermore, it is important to note that the methods presented in this review focused primarily on the deterministic formulation.

Therefore, this review is by no means an exhaustive list of the approaches to hand-eye calibration for visual guided robots. It is noted that there are other key methods available, which include (but are not limited to) model based [56, 57] and probabilistic [58–62] formulation of the hand-eye calibration problem. The intent of this review is to act as a guide to academics and industrial practitioners from which further research in this topic area can be incited.

2.3 Hand-eye calibration algorithms

2.3.1 Homogeneous transform equation

Based on the work of Shiu and Ahmad [47], the hand-eye transform can be obtained by solving the homogeneous transform equation given by

$$A_{c_2}^{c_1} X_h^c = X_h^c B_{h_2}^{h_1}, \tag{2.4}$$

where, $A_{c_2}^{c_1}$ and $B_{h_2}^{h_1}$ are the homogeneous transform matrices representation of the relative motions of the attached camera and the robot hand between two points, respectively, while X_h^c is the required transform between the robot hand and the camera as shown in Figure 2.3. $A_{c_2}^{c_1}$ and $B_{h_2}^{h_1}$ can be expressed as the product of two rigid body transform given by

$$A_{c_2}^{c_1} = A_w^{c_1}(A_w^{c_2})^{-1}, \tag{2.5a}$$

$$B_{h_2}^{h_1} = B_b^{h_1}(B_b^{h_2})^{-1}, \tag{2.5b}$$

where $A_w^{c_1}$, $A_w^{c_2}$ and $B_b^{h_1}$, $B_b^{h_2}$ are the poses of the camera with respect to the world frame or calibration object, and the poses of the robot

hand and with respect to the robot base, respectively, for different robot positions. Equation (2.4) can be represented in a matrix form as

$$\begin{pmatrix} R_A & \vec{t}_A \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} R_X & \vec{t}_X \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} R_X & \vec{t}_X \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} R_B & \vec{t}_B \\ 0^T & 1 \end{pmatrix}, \tag{2.6}$$

where R is a 3×3 rotation matrix and $\vec{t}$ is a 3×1 translation vector. Hence, the calibration operation involves obtaining sets of robot hand and camera poses as shown in Figure 2.4. While the hand poses can easily be obtained from the robot forward kinematics using the joint encoder readings, the camera pose is usually estimated by observing a set of 3D points provided by a calibration object and their corresponding 2D images using Perspective-n-point algorithm [16, 63]. While this formulation shows a more intuitive way to represent and solve the hand-eye problem, estimating the hand-eye transform based on Equation (2.4) is not trivial. This is because the Special Euclidean SE(3) group structure of the homogeneous matrices must be preserved in the solution. Hence, the solution to this form of matrix equation using general matrix algebra [64] would not work.

Finding methods of solving the homogeneous transform equation that meets this requirement has been the focus of majority of the research in hand-eye calibration. Several solutions have been proffered over the years, each with its strengths and weaknesses. They can be grouped based on how the rotation and translation parameters are estimated as *separated* or *simultaneous* solutions. In the separated solutions, the rotation parameter is first estimated based on representing Equation (2.6) as

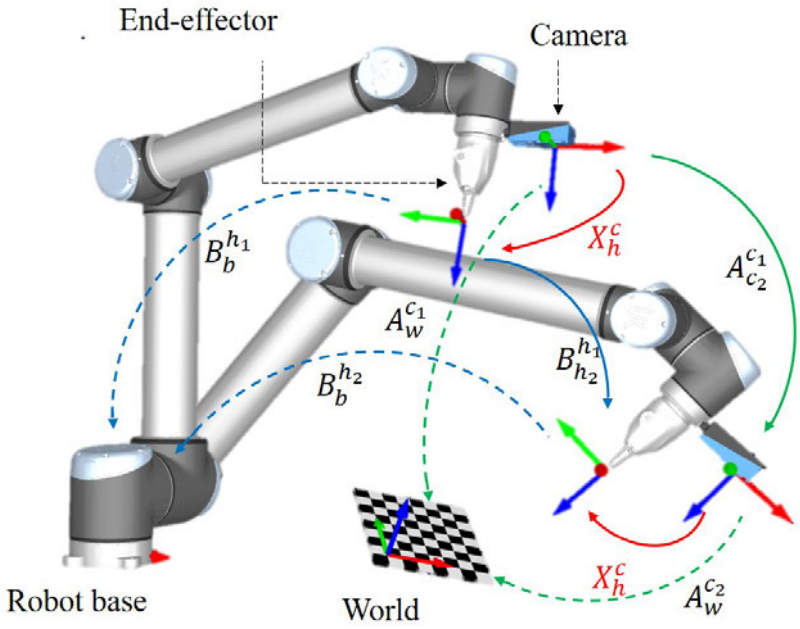


FIGURE 2.3: Hand-eye calibration setup.

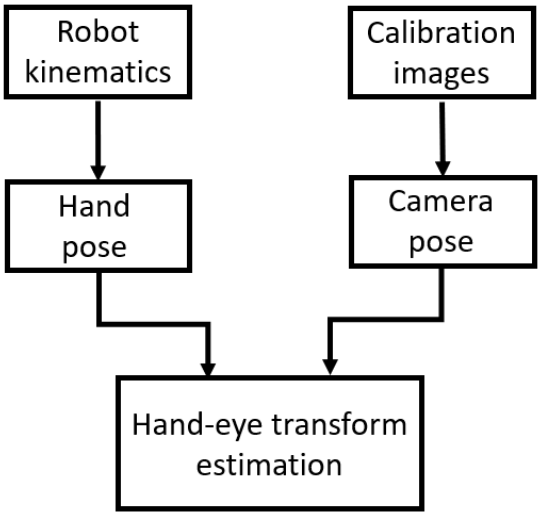


FIGURE 2.4: Calibration process flow using homogeneous transform equation.

$$\begin{pmatrix} R_A R_X & R_A \vec{t}_X + \vec{t}_A \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} R_X R_B & R_X \vec{t}_B + \vec{t}_X \\ 0^T & 1 \end{pmatrix}. \tag{2.7}$$

hence,

$$R_A R_X = R_X R_B, \tag{2.8a}$$

$$R_A \vec{t}_X + \vec{t}_A = R_X \vec{t}_B + \vec{t}_X. \tag{2.8b}$$

If R_X is known, then Equation (2.8) becomes linear and $\vec{t}_X$ can then be estimated. The different techniques that focus on the parametrisation of R_X include, Angle-axis [47, 51], Lie algebra [52], Quaternions [53] and Kronecker product [65]. For the details of the implementations of these algorithms, see the listed references above. Based on the practical considerations, generally, this group of solutions is computationally fast but suffers in terms of accuracy, especially in the translation estimates. This is due to the assumption that no relationship exists between the rotation and translation parameters, hence, their separate estimation. However, these two parameters are tightly coupled with high level of non-linearity [66], and estimating them separately would lead to the propagation of errors from the rotation estimates onto the translation estimates.

The simultaneous solutions provide a way of solving for the rotation and translation parameters simultaneously, either analytically or by way of numerical optimisation. Representative implementations based on analytical approach include Quaternions [67], Screw motion [68], Dual Quaternions [69], Kronecker product [70], Dual Tensor [71], and Dual Lie algebra [72], while implementations based on numerical optimisation include Gradient/Newton optimisation method [73], Linear-matrix-inequality [74], Alternative linear programming [75], and Pseudo-inverse [76]. These methods can generate highly accurate results and generally avoid the problem stated earlier for the separated solutions. However, their implementations are usually complex, which may affect their computational speed. Furthermore,

the optimisation methods may suffer from the problem of not guaranteeing convergence, being trapped in a local minima of the cost function, or being dependent on a good starting estimate. A comparison of these approaches based on the accuracy and the computational speed is shown in Table 2.1. The accuracy criteria are based on the Euclidean norm of the combined rotation and translation error (unitless) for N robot movements derived from Equation (2.4), as given by Equation (2.9). The computation time is in seconds, based on execution on a MacBook Pro 2017 with i7-3.5Ghz CPU along with the MATLAB r2018a software [77].

$$\text{Error} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\|A_iX - XB_i\|^2}. \tag{2.9}$$

It is important to note that the values in Table 2.1 can only be considered as an overview of what can be expected, especially due to the fact that it is devoid of any measurement uncertainty. This is a largely ignored area of research when considering hand-eye calibration, as only a hand-full of works [58, 66, 77, 78] has taken measurement uncertainty into consideration given the large number of research outputs in this area. However, it is important to note that the accuracy of the calibration methods can be improved by increasing the number of robot movements used during the calibration process, maximising the angular spread between the different robot movements, minimising the distance between the camera and the calibration target and minimising the distance moved by the robot arm between two positions [51].

TABLE 2.1: Speed and error comparison of different approaches to solving the homogeneous transform equation. Speed is in milliseconds while the accuracy is unitless, based on the Euclidean norm of the combined rotation and translation given by Equation (2.9)

Methods	**Type**	**Computation Speed (milliseconds)**	**Error** ($\times 10^{-3}$)
Angle-axis [51]	Separated	40.50	12.046
Lie algebra [52]	Separated	35.95	6.825
Quaternions [53]	Separated	10.19	6.826
Dual quaternions [69]	Simultaneous, Analytical	38.30	6.708
Kronecker product [70]	Simultaneous, Analytical	57.29	7.265
Linear-matrix-inequality [74]	Simultaneous, Optimisation	14.58	7.980
Alternative linear programming [75]	Simultaneous, Optimisation	80.86	7.149
Pseudo-inverse [76]	Simultaneous, Optimisation	8.82	7.613

2.3.2 Reprojection error minimisation

The homogeneous transform equation relies on the hand and camera pose information to estimate the hand-eye transformation. As such, errors in these pose estimates will affect the end result of the calibration. While the hand pose errors can be minimised by calibrating the robot [79], through reprojecting the image of the calibration pattern at each hand position and minimising the error between the real image and the reprojected image, the required hand-eye transform can then be estimated as shown in the process flow in Figure 2.5.

Reprojection error minimisation is a well-known technique used in computer vision for pose estimation [80, 81], 3D measurements [82] and shape reconstruction [83] [84], with high level of accuracy and robustness. It shows how precise an estimated 3D world point $\hat{X}$ recreates the true projection x on the image (see Figure 2.6). If P is the projection matrix of the camera, then the image projection $\hat{x}$ can

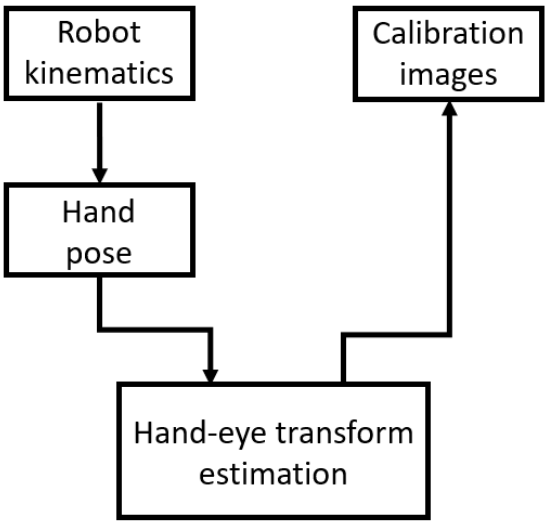


FIGURE 2.5: Calibration process flow using reprojection error minimisation.

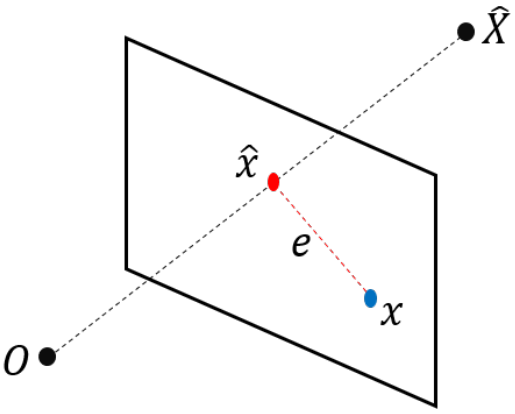


FIGURE 2.6: Reprojection error.

be expressed as $\hat{x} = P\hat{X}$, where $e(x, \hat{x})$ represents the reprojection error is the Euclidean distance between x and $\hat{x}$. By minimising e the true projection matrix can be obtained, and if the camera calibration is known, then the pose of the camera can be realised implicitly.

The main advantage of this technique over the homogeneous transform equation is that it directly takes images of the calibration object without requiring an explicit pose estimate of the camera, which may otherwise contribute to errors. The Perspective-n-point algorithm is usually used in the estimation of the camera pose information from the pattern images [85, 86]. However, this can be

problematic when using cameras with a narrow field of view such as in thermographic cameras [87]. Furthermore, the formulation of the homogeneous transform equation is perfectly suited to normal cameras, whose optics are modeled using the pinhole camera projection model. When considering vision sensors with different optics, such as in X-rays with source-detector projection model (see Figure 2.7), it becomes difficult to use the homogeneous transform formulation as the typical pinhole projection model does not provide a proper representation of its optics. One way of achieving this is by using pose graph optimisation [88], which estimates the relative pose of an object based on a network of observed pose sequences. With pose graph optimisation, it becomes possible to extend the calibration to vision sensors with a different optical projection model like in the source-detector model, where the source pose and the detector pose can be reliably represented in the pose graph.

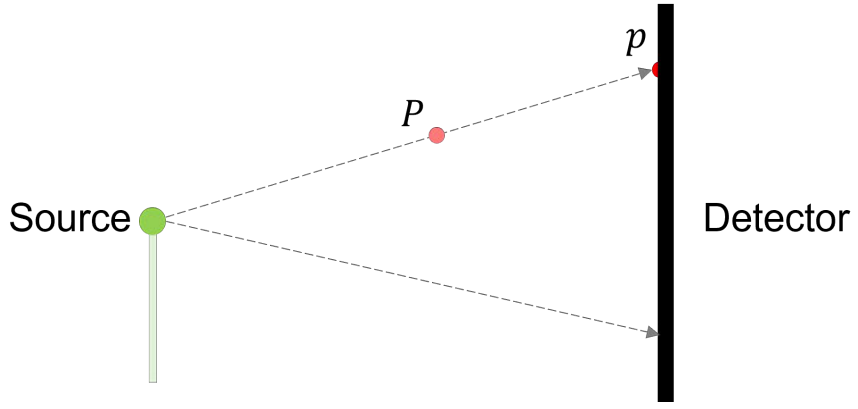


FIGURE 2.7: Source-detector projection model. Light ray from the source passes through a world object P and is projected on the detector at point p

2.3.3 Artificial Neural network

Artificial neural network (ANN) is motivated by the neural system in the brain and is one of the most commonly used tools in machine learning [89]. In its basic form, it consists of layers of interconnected

nodes, each representing a mathematical function. The strength of ANN comes from its ability to model highly non-linear functions that map an input to an output (see Figure 2.8). Hence, its application in pattern recognition [90], robotics [91], signal and image processing [92] and non-linear system state estimation [93–95] have been very successful. An ANN model is obtained by training the network with a set of input and corresponding output data to obtain a set of optimised network parameters. The trained model with its optimised parameters can then be applied to an appropriate input to get the expected output.

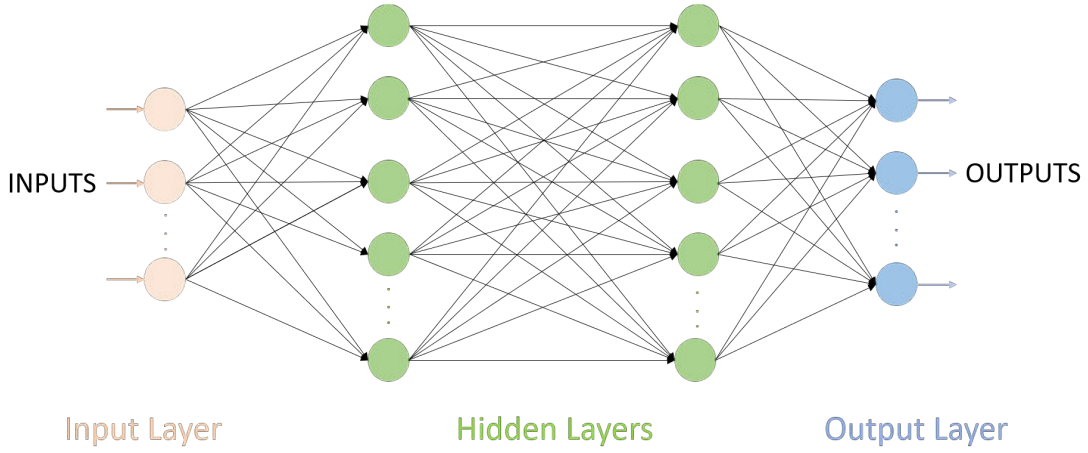


FIGURE 2.8: A simple artificial neural network with two hidden layers.

Employing ANN in hand-eye calibration can be thought of as finding a mapping between the hand coordinate with respect to the robot base and the respective image coordinate of the calibration object. This problem can be posed as $A = f_n(B)$, where A and B are the robot's hand coordinate and calibration image coordinate respectively and f_n is the function depicting the non-linear ANN model. With a trained model, the required hand coordinate for a corresponding object position as observed from the camera can be obtained. An advantage of this formulation is that it can be used without the knowledge of the camera parameters or pose estimation

[96]. This comes from the strong ability of ANN to generalise non-linear relationships between variables, which also makes it suitable for handling noise [96].

While ANN has some comparative advantages over the methods of homogeneous transform equation and reprojection error minimisation, it is important to note that the solutions provided by ANN are usually unexplainable [97]. This can lead to mistrust of the system and difficulty in troubleshooting problems. Furthermore, the performance of an ANN model is highly dependent on the network structure used [98] for which there is no definitive rule for appropriate specification. As such, it is common to select a network structure based on trial and error and user experience.

Parameter over-fitting is another limitation of ANN [99]. This is usually attributed to the failure to properly generalise the model on the available data set, where the model is too simple that could not learn enough, or the model is too complex that it learn too much and over-fits the data. Techniques for preventing over-fitting includes simplifying a complex model, stopping the training early when error starts to increase, data augmentation and regularisation [100].

Table 2.2 shows a comparison of the methods of homogeneous transform equation, reprojection error minimisation and artificial neural network for hand-eye calibration.

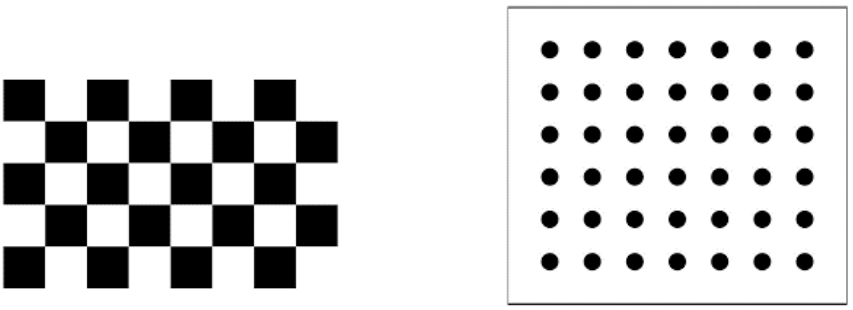
2.4 Calibration target

The calibration target (object) is a very important piece of a calibration process, be it for camera calibration or hand-eye calibration. This subject is rarely discussed when dealing with calibration and

TABLE 2.2: Homogeneous transform equation vs reprojection error minimisation vs artificial neural network for hand-eye calibration.

Homogeneous transform equation	**Reprojection error minimisation**	**Artificial neural network**
Requires explicit camera pose estimation	Camera pose estimation is implicit	Does not require camera calibration or pose estimation
Suited to camera that can be described with pinhole camera model	Because it uses direct images from the camera, it can accommodate other camera models	Model generalisation means it can accommodate other camera models
No issue with overfitting of solutions	Solutions can be prone to overfitting	Parameter over-fitting can be a limitation
Computation time of 0.142 s [88]	Computation time of 0.272 s [88]	Computation time of 2.355 s [96]
Error of 1.715 mm [88]	Error of 1.380 mm [88]	Error of 0.923 mm [96]

oftentimes, the decision to use a particular calibration target is not objective with more focus on lens distortion modeling and parameter optimisation [101]. Calibration patterns such as checkerboards and circles are the most used [102] (see Figure 2.9). This is due to the ease with which they can be created with sufficient accuracy, and their data points can be obtained easily using standard image processing algorithms [102]. During calibration, the calibration pattern captured by the camera can undergo perspective or non-linear distortion, or both. While the perspective distortion is due to the relative 3D position of the points, the non-linear distortion is due to camera lens distortion. How the resulting distortion affects the different calibration patterns determines their reliability.



(A) Checkerboard (B) Circle

FIGURE 2.9: Different types of calibration patterns [103]

2.4.1 Checkerboard target

Checkerboard target is the most common calibration pattern [104–107]. The interest points are the corners of the squares, which can be detected as the intersection of the lines that make up the square edges. Mathematically, these intersection points are the saddle points, which can easily be detected as points, where the first derivative goes to zero. The detection algorithm usually starts by binarising the image, followed by filtering to ensure that the size and organisation meet the dimension and structure specified by the user. The main disadvantage of checkerboard targets is that it is usually difficult to get the exact boundary of the corners [108]. However, the detection of the corners of the squares can usually be done with a sufficient level of accuracy because the corners, being infinitely small are mostly invariant to perspective and lens distortion [102].

Because of the alternating colours of adjacent squares, the checkerboard target can be made rotation-invariant by making the number of rows and columns even and odd, respectively, or vice-versa. Otherwise, with both rows and columns either even or odd, the pattern creates a 180-degree ambiguity that can be problematic for multi-camera calibration, where a similar point needs to be identified by multiple cameras like in the calibration with stereo cameras.

2.4.2 Circular grid target

Circular grid targets are based on circles with the feature point being at the center of the circle. Appropriate circles in the target can be detected using characteristics like circularity and convexity, and bad-featured circles can be eliminated. While the circles themselves

are easy to detect and to be filtered, unlike the checkerboard target, they are not invariant under perspective and lens distortion as shown in Figure 2.10. Under perspective projection, the circles are imaged as ellipses. Ideally, this can be solved using image rectification, however, the additional lens distortion on the ellipses adds some bias to the detected points, which in general would require a more complex algorithm to correct.

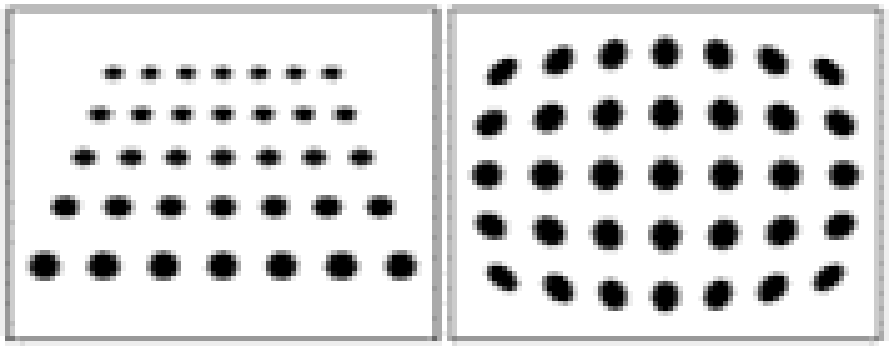


FIGURE 2.10: Circular grid under perspective (left) and lens (right) distortion [103].

Just as in checkerboard targets, the circular grid target can be made rotation invariant for multi-camera view. This is done by using an asymmetric grid pattern as shown in Figure 2.11.

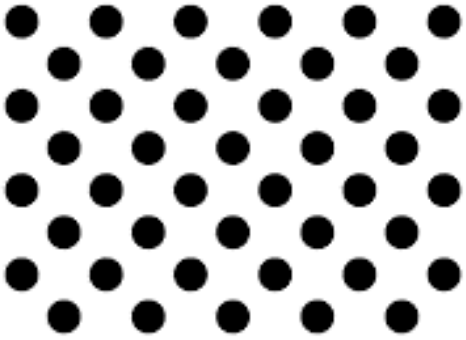


FIGURE 2.11: Asymmetric circular grid pattern [109].

2.4.3 Discussions

Checkerboard and circular grid are the most widely used patterns for vision system calibration. The choice of pattern used depends on the application constraints such as accuracy, the complexity of detection algorithm, distortion, etc. The feature points for the checkerboard pattern are the intersection of the lines that can easily be obtained using standard corner detection strategies [110]. For circular grid patterns, the feature point commonly used is the center of mass of the circle pixels. Often times the estimated position of the checkerboard corner, or circle center does not fall at the exact point and further computation would be required for sub-pixel accuracy[102]. Generally, the mathematics involved in realising sub-pixel accuracy for circular grids is much more complex than for checkerboard patterns [102]. This complexity is compounded by the fact that the feature point for circular grid is affected by both radial and perspective bias. Hence the accuracy of circular grid depends on how well the true centre of the circle can be determined. Figure 2.12 illustrates the effect of radial distortion on the accurate detection of the features for checkerboard and circular targets. In this illustration the radial distortion coefficient k_1 as given in Equation (2.10) is increased from -2 to 2, where $k_i, i = 2, 3, ...$ are the distortion coefficients, r_d is the distortion radius, (x_u, y_u) and (x_d, y_d) are the undistorted and distorted image points respectively. Table 2.3 shows a summary of the comparison between the properties of checkerboard and circular grid patterns.

$$x_u = x_d(1 + k_1 r_d^2 + k_2 r_d^4 + \cdots) \tag{2.10a}$$

$$y_u = y_d(1 + k_1 r_d^2 + k_2 r_d^4 + \cdots). \tag{2.10b}$$

TABLE 2.3: Comparison of checkerboard and circular grid calibration patterns.

	Checkerboard	**Circle**
Feature points	Corners and edges	Centroid and conics
Detection Complexity	Features can easily be resolved even in the presence of distortion	Feature detection is usually complicated by distortion
Perspective distortion	No	Yes
Effect of lens distortion (See Figure 2.12)	low	High
Occlusion	It may be possible to interpolate lines to infer intersection	Connecting distorted points to infer intersection is not usually possible
Sub-pixel accuracy	Simple interpolations from already acquired points can be used to achieve sub-pixel accuracy for other points	Special techniques are required to achieve sub-pixel accuracy

2.4.4 Other Calibration Targets

Other calibration targets exist with the aim of overcoming the limitations of the checkerboard and circular calibration targets. Most of these come with some form of encoding marker. An example of such is the CharuCo target shown in Figure 2.13.

CharuCo target is a board made up of a grid of aruco markers. Aruco markers are square fudicial markers made up of an inner binary matrix which determines its identifier (id) and a wide black border. The black border facilitates its fast detection in the image and the binary

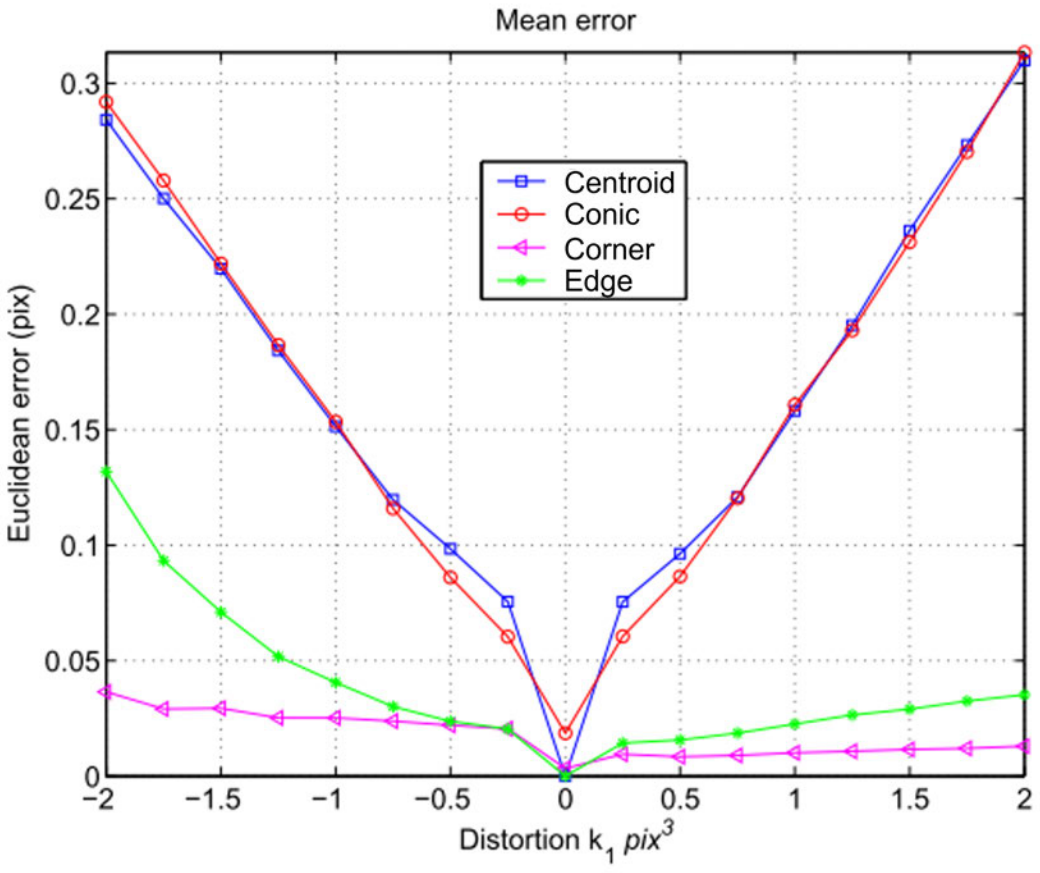


FIGURE 2.12: Accuracy evaluation for feature points detection for checkerboard (corners and edges) and circular grid (centroid and conics) targets for increasing levels of radial distortion k_1 [102].

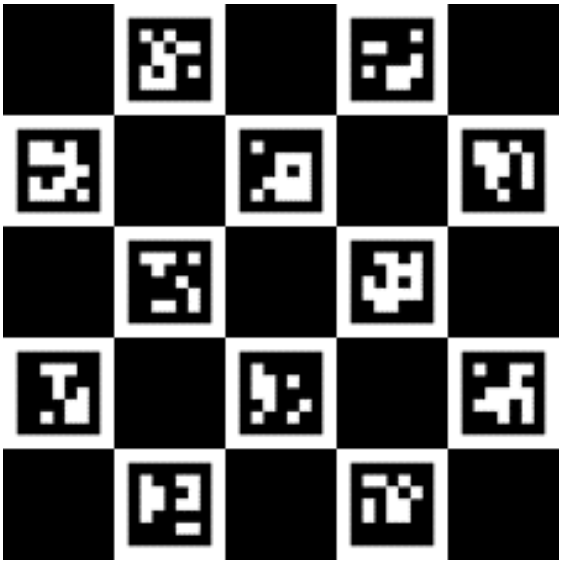


FIGURE 2.13: CharuCo target [111].

codification allows its identification and the application of error detection and correction techniques. In the CharuCo target, the light

squares are uniquely encoded. Thus, this makes CharuCo target possible to carry out calibration even with part occlusion or poor image conditions such as in inhomogeneous lighting, while maintaining the advantage that the intersection of the square edges or interest points can easily be recovered, when an ordinary checkerboard or circular grid target under these conditions would normally fail. The main drawback of this calibration target is the complex algorithm required for the detection and decoding of the patterns.

2.5 Common Challenges of Hand-eye Calibration

Hand-eye calibration is an active field of research in robotics and computer vision mainly due to the importance of precision and accuracy in these industries. For example, while an accuracy level of 1 mm may be required for spot welding operation in the automotive industry, an accuracy measure of at least ten to twenty-fold would be required in the aerospace industry [112]. Similar accuracy levels can also be found in robotic applications in the health industry, where safety is of utmost importance [113]. Achieving this level of accuracy is a major challenge in hand-eye calibration for robots due to a number of factors such as data asynchronicity, noise, and limited motion range.

2.5.1 Data asynchronicity

The hand-eye calibration problem is constrained on data from two sources: the eye (camera) and the hand (robot). This constraint requires correspondence in the data stream from both sources, which may not be practically possible, resulting in temporal misalignment

in the data [114]. This temporal misalignment may be due to the differences in the operating frequency of the sensors, difficulty in synchronising the trigger for the data capture on both sources or missed data in either stream. Many solutions to hand-eye calibration are offline in nature [52, 53, 65, 76], where the calibration setup is made, complete pose data set for both the hand and the eye with respect to the robot base and world, respectively, are acquired, and computation of the required hand-eye transform made. Regardless of the fact that the acquisition of both sets of data is made in discrete steps, data asynchronicity still forms a major problem that affects the correspondence of the data. Offline calibration nonetheless is not suitable for certain applications. An example is in critical operations like RAS, where frequent changes in setup and recalibration is an expensive operations that must be dealt with on the fly [115]. This type of application requires online calibration [116], where data is continually being captured and used to update the calibration algorithm, rendering the need for data synchronisation on both sources very apparent.

One solution to the problem of temporal misalignment is the use of timestamp [117]. By timestamping the data from both sources, users could manually or programmatically synchronise the data streams and also avoid missing data. Cross-correlation techniques can also be used to achieve data synchronisation for hand-eye calibration as in [118, 119]. Normalising and resampling the data before the cross-correlation operation can be used to ensure that differential data length caused by time delay or different sampling rates do not affect the result. A more elegant solution can be found in the use of a real-time embedded operating system for the control of the data capture and synchronisation operation [120]. This, however, would

require compatibility with different sensors and robot systems and can quickly make the setup less attractive in terms of cost and complexity.

2.5.2 Noise

Noise is a major problem in hand-eye calibration, which arises as a result of perturbations in the robot-camera assembly. This causes some degree of uncertainty in the calibration results. A direct impact of the noise in hand-eye calibration is the need to use measurements from multiple coordinate frames (greater than the theoretical minimum) for the estimation of the hand-eye transform [51]. The required hand-eye transform is estimated from a system of equations based on the rigid body transform of the robot-camera assembly, which normally results in an overdetermined system [47]. In an ideal scenario, with no noise in the system, because the measurements are physically constrained to be consistent with the robot-camera assembly, the set of equations could be solved by a simple least square method. Since there are more equations than unknowns in the presence of measurement noise, the equations become inconsistent and multiple frames or robot motions would be required to accurately estimate the system variables.

Noise in robot hand-eye calibration can be categorised into two forms. These are noise as a result of the robot's motion and the camera's motions. Noise from the robot motion directly affects the kinematic model of the robot as they are caused by measurements from the joint encoders or optical trackers in the robot. The error in measurement can be due to various factors such as kinematic errors, non-kinematic errors, and joint errors.

- *Kinematic errors*: Kinematic errors are related to and have a direct impact on the kinematic model of the robot [121, 122]. These may be due to manufacturing and assembly tolerances, the geometry of the robot components such as orthogonality or parallelism, or the position of the reference frame.
- *Non-kinematic errors*: Unlike kinematic errors, non-kinematic errors are due to the mechanical characteristics of the robot components such as stiffness, backlash elasticity, and impact of temperature [123, 124].
- *Joint errors*: Joint errors are directly related to the error in motion measured at individual joints of the robots by the joint encoders and are caused by the sensors themselves [125, 126].

Noise from the camera motions is a direct consequence of camera calibration, which can result from low camera quality, poor calibration parameter estimates, low-quality calibration patterns, etc. These errors, while they can be small from a camera calibration perspective [127], can be propagated to the estimates of the hand-eye calibration.

2.5.3 Limited motion range

The range of allowable motion of the robot hand during calibration has a direct impact on the results of the calibration. Large motions in the robot hand have the effect of suppressing noise in the setup that can arise due to perturbations [51]. Despite this advantage, not every application is able to permit a wide motion range. In RAS, only a small motion range of the surgical tool is permitted. This is usually constrained to within the vicinity of the trocar entry ports [123]. This is done to minimise the damage that can be done to surrounding

tissues at the entry ports [128]. In pick-and-place applications like sorting and assembly facilities, the constraint is the field of view of the camera outside in which the operation of the robot is not feasible [129]. In other instances, the robot motion is limited to a particular area to provide a safe environment in which human operators can operate [130, 131]. In these applications, the robot is controlled by an embedded control system that specifies and limits the motion of the hand to a given workspace. While the allowable range of motion of the robot hand cannot always be controlled, a lot of gains can be achieved by implementing proper path planning algorithm and pose selection methods to obtain well-conditioned robot hand-eye constraints [51, 132, 133].

2.6 Summary

In this chapter, different solutions to hand-eye calibration were discussed with the aim of presenting their strengths and weaknesses. The purpose was to provide necessary information that would be required for implementation by academics and industrial practitioners, as well as encourage further research. The most common formulation of the problem requires finding a solution to the homogeneous transform equation $AX = XB$. A lot of research have be done in this area, with solutions found using the angle-axis representation of the rotation parameter, Lie algebra, Quaternions, Dual Quaternions, Screw motion, Kronecker product as well as using optimisation techniques. Each of the resulting algorithms differs in their level of accuracy and computational requirement, which needs to be taken into account by academics or industrial practitioners depending on their design constraints. Alternate methods that solve the problem

using reprojection error minimisation and artificial neural networks are also presented. The main advantages of the method of reprojection error minimisation are error avoidance due to camera pose estimation and the ability to work with camera models (e.g. source-detection model used for X-rays) other than the pinhole projection model. While the method of artificial neural network also simplifies the problem by using only images taken by the camera, network parametrisation and over-fitting may limit its usage.

Different considerations for the choice of calibration pattern are also discussed, with checkerboard and circular grid patterns being the most common calibration targets. While sub-pixel accuracy can be achieved using either of the target choices, circular grid targets usually require more complex algorithms. How the patterns respond to perspective and lens distortion plays a huge role in their reliability, with circular grid targets being more susceptible to distortions that must be corrected. The CharuCo target on the other hand embeds encodings on its pattern to avoid the limitations of the most commonly used checkered board pattern.

Finally, some common challenges that are expected in the calibration of robots' eye-hand systems was discussed. While proper planning and appropriate calibration setup can improve the calibration estimate, it is sometimes difficult to meet all the conditions for improved accuracy and a compromise has to be made. Data asynchronicity, noise, and limited motion range are identified as some of the challenges of hand-eye calibration that can also gain improvement from proper path planning, calibration setup, and robot calibration prior to hand-eye calibration. In general, for accurate vision-guided robotic operation, there has to be proper calibration of the robot to correct the joint variables and robot parameters, calibration of

the camera to determine its accurate pose relative to world measurements, and then calibration of the hand-eye system to obtain the transformation of the camera relative to the robot's hand.

Chapter 3

Accuracy evaluation of hand-eye calibration techniques

3.1 Introduction

Several techniques have been proposed to solve the calibration problem, and they can broadly be classified as either the separated methods or the simultaneous methods. In the separated methods, the rotation parameter is first estimated from Equation (2.8a), then the translation parameter is estimated based on the estimated rotation using the linear equation in Equation (2.8b). Shiu and Ahmed [47] and Tsai and Lenz [51] employed axis-angle parameterisation for the estimation of the rotation parameter while Chou and Kamel [53] and Park and Martin [52] used unit quaternions and Lie-algebra respectively to represent the rotation parameter. Because the separated methods required estimating the translation parameter from the rotation parameter, errors from the rotation estimates are directly propagated to the translation estimates. Moreover, the separation of the rotation and translation parameters loses the inherent coupling between both parameters [68]. These arguments necessitated the need for simultaneous solutions to the hand-eye calibration

problem. Chen [68] provided the first simultaneous solution to the hand-eye calibration problem. His method based on screw theory described the calibration problem as the rigid transformation between the screw axis of the hand and the camera. Zhao and Liu [134] extended the screw motion approach by representing the rotation with unit quaternions and formed a system of linear equations that were solved using Singular Value Decomposition (SVD). Daniilidis and Bayro-Corrochano [69] introduced the method of dual quaternion for solving the rotation and translation components simultaneously. Li et al. [135] applied Kronecker product in their approach but required additional orthogonalisation steps to ensure a rotation matrix is realised.

In this study, we provide a systematic evaluation of how different factors (rotation noise, translation noise, rotational motion, translational motion) can impact the hand-eye calibration algorithms, through simulation with synthetic data and real experimental data. We also evaluate the computation time of each algorithm as a way to assess their relative complexities. Using six algorithms (Tsai and Lenz [51], Chou and Kamel [53], Park and Martin [52], Daniilidis and Bayro-Corrochano [69], Lu and Chou [67], and Li et. Al. [136]) as references, we comparatively show that the impact of those aforementioned factors does not follow a similar trend. The choice of these six algorithms is based on the performance evaluation from recent studies in comparison to other algorithms. From [137], the method from Park and Martin [52] showed the second-best accuracy based on Reconstruction Accuracy Error (RAE) while providing the best computation time based on the comparison of 4 hand-eye calibration algorithms. From [77] the methods form Daniilidis and Bayro-Corrochano [69], and Chou and Kamel [53] provided the best and

second-best accuracies respectively in terms of relative pose error based on comparison of 10 hand-eye calibration algorithms. Based on the experimental results from [138], with increase in the size of the dataset, the method from Tsai and Lenz [51] provided the second-best rotation accuracy, based on comparison with 5 other hand-eye calibration methods. The choice of the method from Li et.al. [136] was based on the fact that it provided the best accuracy when evaluated against two other methods that employed Kronecker product technique based on the relative rotation and translation errors. This was thus chosen as a candidate method for the evaluation of algorithms that utilise Kronecker product implementation. Furthermore, they also form the base idea on the development of most of the proposed hand-eye calibration algorithms, making them widely used for benchmarking [77, 139] which gives them their popularity. These chosen algorithms also cover the different methods (separated and simultaneous) of generating solutions to the hand-eye calibration problem. In particular, the simultaneous methods are more resistant to rotation noise, whereas the separated methods are better at dealing with translation noise. Furthermore, while increasing the robot rotation motion span during calibration enhances the accuracy of the separated methods, it has a negative effect on the simultaneous methods. On the other hand, increasing the translation motion range improves the accuracy of simultaneous methods but degrades the accuracy of the separated methods. These findings suggest that those conditions should be considered when benchmarking algorithms or performing a calibration process for enhanced accuracy.

3.2 Nomenclature

R: 3×3 rotation matrix

$\vec{t}$: 3×1 translation vector

$[\vec{v}]_\times$: Skew of vector $\vec{v}$ such that

$$[\vec{v}]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ -v_3 & 0 & -v_1 \\ v_2 & v_1 & 0 \end{pmatrix}$$

$q(q_0, \vec{q})$: Unit rotation quaternion made up of a scalar part q_0 and a vector part $\vec{q}$

$q^\pm$: Representation of a quaternion to aid matrix multiplication such that

$$q^\pm = \begin{pmatrix} q_0 & -\vec{q}^T \\ \vec{q} & q_0 I \mp [\vec{q}]_\times \end{pmatrix}$$

where I is the identity matrix.

q' : The dual of a quaternion

$sign(x)$: Defines the sign of a value such that

$$sign(x) = \begin{cases} -1, x < 0 \\ 1, x \geq 0 \end{cases}$$

$vec(A)$: Vectorization of matrix A

$\otimes$: Kronecker product

3.2.1 Hand-eye calibration methods

The hand-eye calibration methods can be categorised based on the representation of the rotation parameter such as angle-axis, quaternions, dual-quaternions, Lie group, etc. They can also be described

based on the parameter estimation procedure as a separable solution or simultaneous solution. This section provides an overview of the six hand-eye calibration algorithms which will be evaluated in this study from the perspective of easy-of-implementation and reproducibility of the presented algorithms. These six algorithms are Methods of Tsai and Lenz [51], Chou and Kamel [53], Park and Martin [52], which are the separated methods and Methods of Daniilidis and Bayro-Corrochano [69], Lu and Chou [67], and Li et.al. [136], which are the simultaneous methods. These methods have been chosen for this study because they not only form the foundations of many hand-eye calibration algorithms but are frequently used for benchmarking newer hand-eye calibration methods.

3.2.2 Method of Tsai and Lenz (1989)

The Method of Tsai and Lenz [51] (hereinafter termed Method Tsai) provides a separable solution to the hand-eye calibration problem using the angle-axis representation of the rotation parameter R_X given by

$$R_X = \mathrm{Rot}(\vec{n}_X, \theta_X), \tag{3.1}$$

where $\vec{n}_X$ is the axis of rotation and θ_X is the angle of rotation. Using this method, the rotation axis and angle can be computed by

$$[\vec{n}_A + \vec{n}_B]_\times \vec{n}'_X = \vec{n}_A - \vec{n}_B \tag{3.2a}$$

$$\vec{n}_X = \frac{2\vec{n}'_X}{\sqrt{1 + \|\vec{n}'_X\|^2}} \tag{3.2b}$$

$$\theta_X = 2\tan^{-1}(\|\vec{n}'_X\|) \tag{3.2c}$$

where n_A and n_B are the axes of rotation in the camera and hand frames respectively, and $[\vec{v}]_\times$ is the skew of the vector $\vec{v}$ given by

$$[\vec{v}]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ -v_3 & 0 & -v_1 \\ v_2 & v_1 & 0 \end{pmatrix} \tag{3.3}$$

The rotation matrix R_X can be obtained by

$$R_X = \left(1 - \frac{\|\vec{n}_X\|^2}{2}\right) \cdot I + 0.5 \left[\vec{n}_X . \vec{n}_X^T + \left(\sqrt{4 - \|\vec{n}_X\|^2}\right) [\vec{n}_X]_\times\right] \tag{3.4}$$

where I is the identity matrix. The motivation of this approach is to provide a solution by solving a fixed linear system of equations, as the earlier approach [47] required an increasing number of equations for each additional robot motion used in the calibration.

3.2.3 Method of Park and Martin (1994)

Park and Martin [52] (hereinafter termed Method Park) formulated a computationally more efficient and linearised method by parameterising the rotation with Lie-group. This provides a logarithmic mapping from the SO(3) group to the corresponding so(3) Lie algebra, where SO(3) and so(3) represent the special orthogonal group matrices of size 3 x 3 and the corresponding Lie algebra matrix of size 3 x 1, respectively. Given that

$$\log(R) = \frac{\theta}{2\sin\theta}(R - R^T), \tag{3.5}$$

where θ satisfies $1 + 2\cos\theta = tr(R)$, where $tr(R)$ is the trace of R. The rotational part of the calibration Equation (2.8a) can be represented by its logarithmic mapping as

$$R_X \beta_i = \alpha_i \tag{3.6}$$

where α and β are $\log(R_A)$ and $\log(B)$, respectively. R_X can be obtained by least-square minimisation such that

$$R_X = (M^T M)^{-\frac{1}{2}} M^T \tag{3.7a}$$

$$M = \sum_{i=1}^{N} \beta_i \alpha_i \tag{3.7b}$$

where N is the number of data points.

While this method is computationally efficient and does well in the presence of noise, the computation of $\log(R_A)$ and $\log(R_B)$ imposes a restriction that R_A and R_B must be rigid transforms, otherwise, it becomes impossible to compute their logarithms.

3.2.4 Method of Daniilidis and Bayro-Corrochano (1996)

Daniilidis and Bayro-Corrochano [69] (hereinafter termed Method Daniilidis) provided an algebraic interpretation of the screw motion approach to hand-eye calibration [68] using dual quaternion representation. For a unit quaternion $q(q_0, \vec{q})$ representing the rotation in a rigid body transformation, its dual $q'(q'_0, \vec{q}')$ is given as

$$q' = \frac{1}{2} t q \tag{3.8}$$

where t is the translation component of the transform. Using only the vector part of the dual quaternion representation of the camera and robot transform, i.e, $a_i(0, \vec{a}_i)$, $a'_i(0, \vec{a}'_i)$ and $b_i(0, \vec{b}_i)$, $b'_i(0, \vec{b}'_i)$, respectively, Equation (2.8) can be formulated as

$$\begin{pmatrix} \vec{a} - \vec{b} & [\vec{a} + \vec{b}]_\times & 0_{3\times1} & 0_{3\times3} & & \\ \vec{a}' - \vec{b}' & [\vec{a}' + \vec{b}']_\times & \vec{a} - \vec{b} & [\vec{a} + \vec{b}]_\times & 0_{3\times1} & 0_{3\times3} \end{pmatrix} \begin{pmatrix} q_x \\ q'_x \end{pmatrix} = 0 \quad (3.9)$$

This matrix has two singular vectors $\vec{u}_1^T = (\vec{v}_1^T, \vec{w}_1^T)$ and $\vec{u}_2^T = (\vec{v}_2^T, \vec{w}_2^T)$ that span the null-space and hence satisfy the equation

$$\begin{pmatrix} q_x \\ q'_x \end{pmatrix} = \lambda_1 \begin{pmatrix} \vec{v}_1 \\ \vec{w}_1 \end{pmatrix} + \lambda_2 \begin{pmatrix} \vec{v}_2 \\ \vec{w}_2 \end{pmatrix} \quad (3.10)$$

To ensure the result is a unit dual quaternion, Equation (3.10) must be solved together with the constraints given by

$$q_x^T q_x = 1 \quad (3.11a)$$

$$q_x^T q'_x = 0 \quad (3.11b)$$

This leads to the formation of two quadratic equations in λ_1 and λ_2 from which (q_x, q'_x) can be determined.

3.2.5 Method of Lu and Chou (1995)

Lu and Chou [67] (hereinafter termed Method Lu) proposed a simultaneous solution by formulating a linear system of equations using quaternion given by

$$\begin{pmatrix} P & Q \\ Q & 0 \end{pmatrix} \begin{pmatrix} q_x \\ t'_x \end{pmatrix} \equiv Cx = 0 \tag{3.12}$$

where $q_x(q_{0_x}, \vec{q}_x)$ is the unit quaternion representation of the rotation and the translation component t_x is given by

$$t'_x = E^T t \tag{3.13}$$

where $E = \begin{pmatrix} -\vec{q}_x & q_{0_x} I + [\vec{q}_x]_\times \end{pmatrix}$, while P and Q are given by

$$P = q_b^-(t_b^+ - t_a^-) \tag{3.14a}$$

$$Q = q_b^+ - q_a^- \tag{3.14b}$$

where q_a and q_b are the quaternion representation of the rotation of the camera frame and robot hand frames respectively, and t_a and t_b are the quaternion representation of the translation of the camera frame and robot hand frames respectively. This system must be solved with the constraint given by

$$q_x^T q_x = 1 \tag{3.15a}$$

$$q_x^T t'_x = 0 \tag{3.15b}$$

3.2.6 Method of Li et. Al. (2018)

To simultaneously solve for the rotation and translation components Li et. al., [136] (hereinafter termed Method Li) described the calibration equation (2.8) using the relationship between matrix vectorisation and Kronecker product. The vectorisation of the product of matrices A, B and C can be written as

$$vec(ABC) = (C^T \otimes A)vec(B) \tag{3.16}$$

Equation (2.8) can thus be written as

$$\begin{pmatrix} I_9 - (R_B \otimes R_A) & 0_{9\times 3} \\ t_B^T \otimes [t_A]_\times & [t_A]_\times (I_3 - R_A) \end{pmatrix} \begin{pmatrix} vec(\hat{R}_X) \\ \hat{t}_X \end{pmatrix} = \begin{pmatrix} 0_{9\times 1} \\ t_A \end{pmatrix} \tag{3.17}$$

Equation (3.17) is thus a linear system that can be solved by least-square. To ensure that the recovered rotation meets the constraint that its determinant is 1, a proportionality constant ω can be calculated as

$$\omega = sign(det(\hat{R}_X))det(\hat{R}_X)^{-\frac{1}{3}} \tag{3.18}$$

The recovered rotation and translation can thus be given as

$$R_X = \omega \hat{R}_X \tag{3.19a}$$

$$t_X = \omega \hat{t}_X \tag{3.19b}$$

3.3 Performance evaluation metrics

The following evaluation metrics were used to comparatively evaluate the performance of the different algorithms, each of which has its usefulness.

3.3.1 Relative transformation error

The relative transformation error E_{rn} is unitless and is derived from Equation (2.8). It evaluates how close the rigid transform on the left side of the equation is to the right side of the equation based on the estimated hand-eye transform X. The relative transformation error is given by

$$E_{rn} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\|A_iX - XB_i\|^2}. \tag{3.20}$$

3.3.2 Rotation error

Two forms of rotation errors are utilised for this evaluation: the relative rotation error E_R, and the mean absolute rotation error E_{R_x}. These are given by

$$E_R = \frac{1}{N}\sum_{i=1}^{N} angle[(R_X R_{B_i})^T (R_{A_i} R_X)], \tag{3.21a}$$

$$E_{R_x} = \frac{1}{N}\sum_{i=1}^{N} angle(\hat{R}_{X_i}^{-1} R_X) \tag{3.21b}$$

where $\hat{R}_{X_i}$ is the rotation estimate value of the hand-eye transform during simulation. The relative rotation error, Equation (3.21a) is suitable for evaluation with real data where the ground-truth hand-eye transform is not available. For simulation study where the ground-truth data is available, then, it becomes more useful to use the mean absolute rotation error E_{R_x} given in Equation (3.21b).

3.3.3 Translation error

Following from the rotation errors, the relative translation error E_T and the mean absolute translation error E_{T_X} are used in this study for real data and simulation studies respectively. These are given by

$$E_T = \frac{1}{N}\sum_{i=1}^{N}\|(R_{A_i}t_X) - t_X - (R_X t_{B_i}) + t_{A_i}\|, \tag{3.22a}$$

$$E_{T_x} = \frac{1}{N}\sum_{i=1}^{N}\|t_X - \hat{t}_X\| \tag{3.22b}$$

3.4 Materials and methods

3.4.1 Real dataset collection

For this experiment, a UR5e robot arm rigidly mounted on the floor to provides the robot pose data and a Microsoft Azure Kinect camera secured to the last link of the robot for the image acquisition which is used to compute the camera poses. A 32mm, 11 by 8 checkerboard pattern was used as the calibration target. During the experiment, the robot arm was moved to a range of positions with the calibration pattern still in the view of the camera for image acquisition. The control of the robot's motion was achieved through an interface with RoboDK running a script written in Python. This ensured a high-level interaction with the robot which makes for easy implementation. During the experiments, software checks were implemented to detect and avoid configuration changes in the robot. Also, the motion of the robot was restricted such that the rotational angle was below 180 degrees. This ensures that the issue of singularity

was avoided during the computation of the hand-eye calibration parameters which occurs close to or at this threshold [47, 51]. The robot poses were obtained directly from the robot pendant while the camera poses were estimated using the P-n-P algorithm from the OpenCV library [140]. The setup is shown in Figure 3.1. A demo video of the calibration operation can be found in Supplementary Video 1. For the evaluation of hand-eye calibration algorithms, a total of 101 robot poses and images of size 1280×720 pixels were acquired.

During the camera pose computation, it was noted that the position of the origin as detected by the P-n-P algorithm from the OpenCV library was sensitive to the orientation of the calibration pattern in the image when both the rows and columns used are of even or odd number as shown in Figure 3.2. Figure 3.2 shows the origin of the target reprojected on the image after pose estimation with the OpenCV P-n-P library. In Figure 3.2A, the origin is located at position 1H on the chessboard. However, when the chessboard is rotated sufficiently as in Figure 3.2B, the origin location changes to position 7A. This leads to a loss in the actual computed rotation.

This change in the origin affected the actual camera pose estimate. Using odd and even numbers of rows and columns in the calibration pattern, however, forced the algorithm to be consistent in the position of the origin for every pose acquisition as shown in Figure 3.2C and 3.2D.

3.4.2 Simulation dataset generation

Using real dataset ensures that the overall dynamics and uncertainties in the system are captured. However, because it is impossible

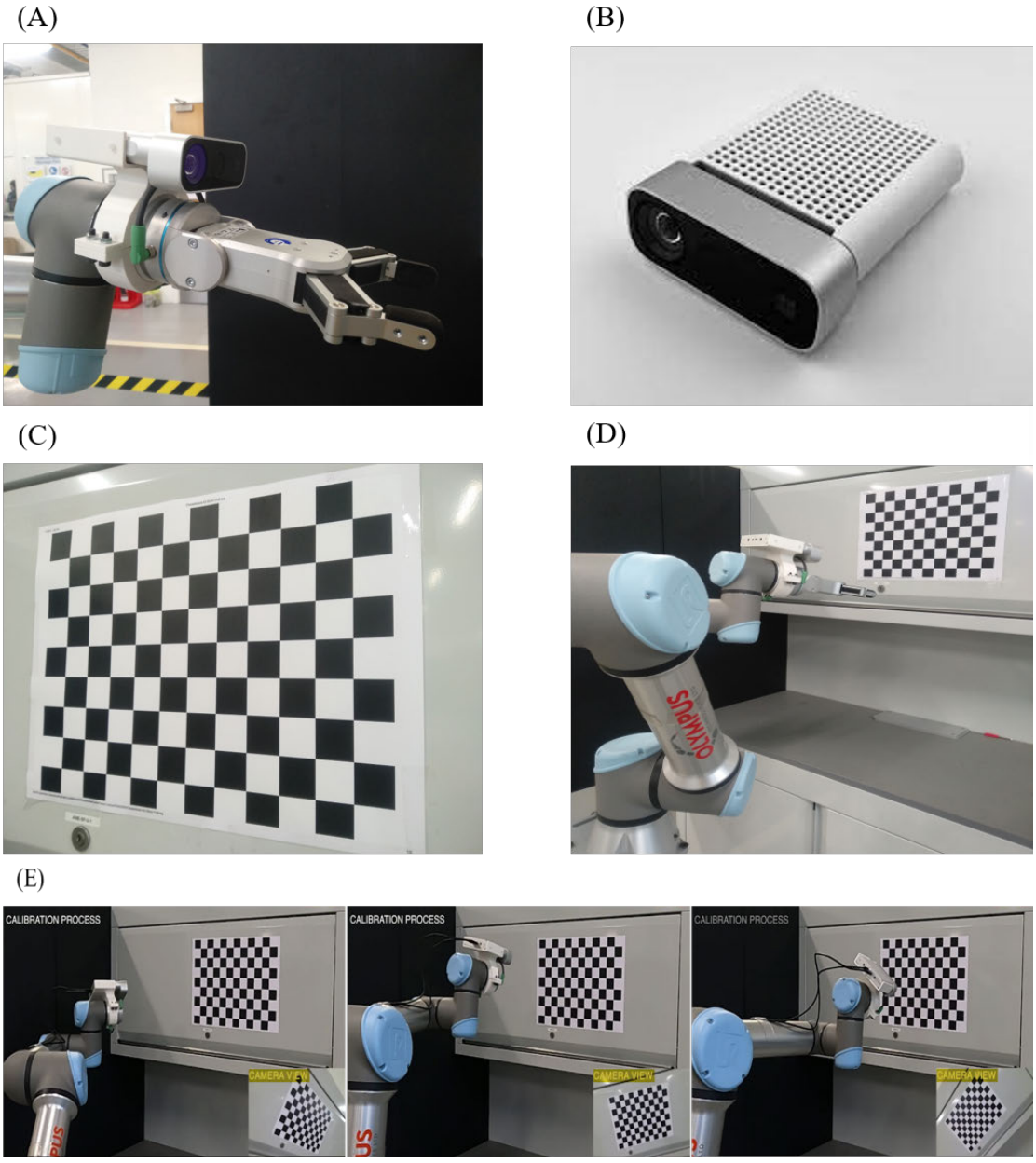


FIGURE 3.1: Hand-eye calibration setup. (A) Camera end-effector setup. (B) Camera. (C) Calibration pattern. (D) Experiment setup (E) Poses of robot and camera view representing camera pose during the calibration process.

to get the ground truth data for the hand-eye transformation, it becomes impossible to make an absolute evaluation of the performances of the different algorithms based on their true rotation and translation estimates. As such, synthetically generated data becomes useful for the study. This also allows for a quick, easy, and in-depth study of various scenarios for hand-eye calibration where the parameters

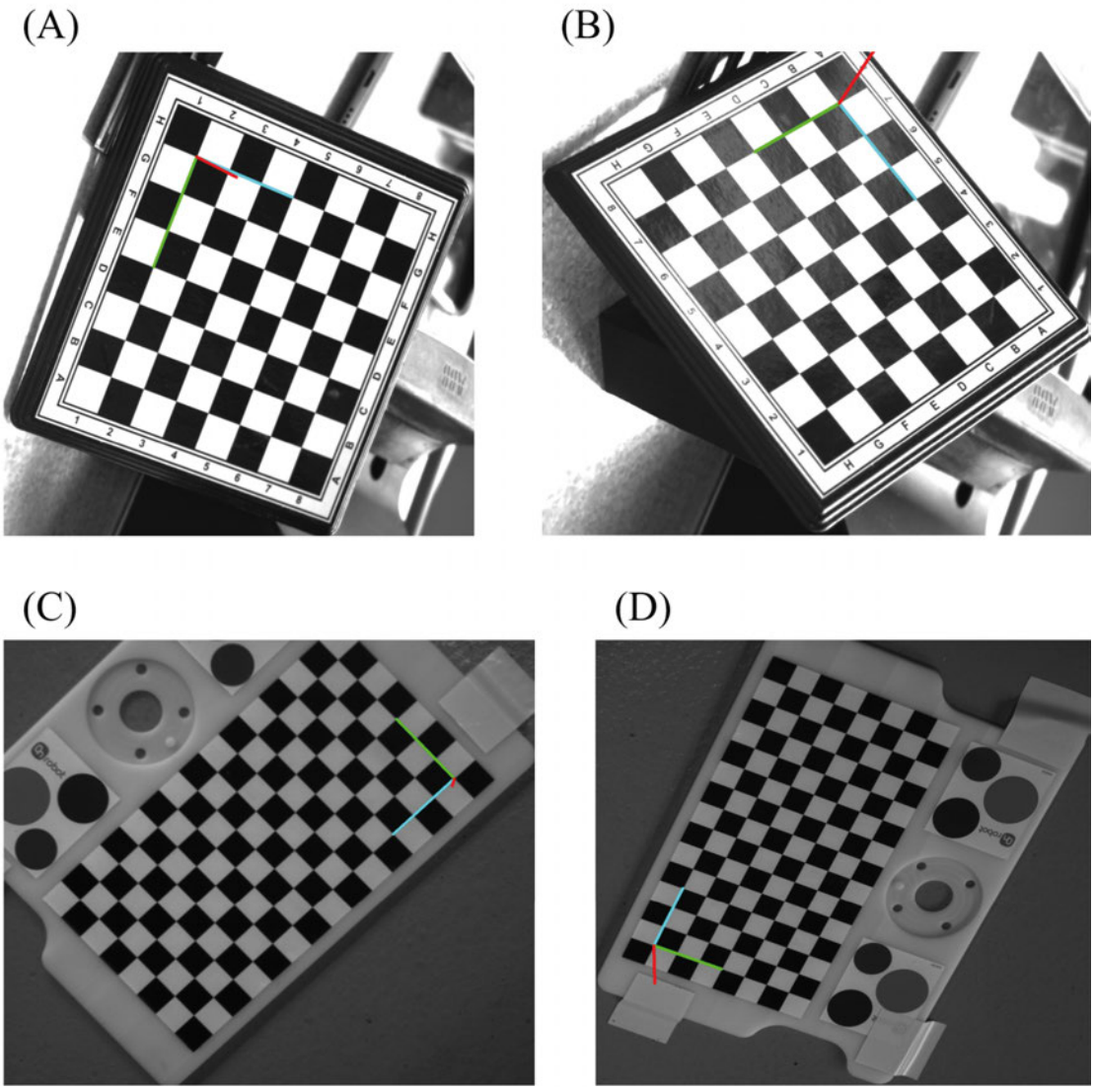


FIGURE 3.2: Change in position of origin (red, green, and blue axes) with change in orientation of target for an odd number of rows and columns. (A) Origin at position 1H. (B) Origin at position 7A. (C) Origin at the bottom right. (D) Origin at the bottom right after rotation.

can be controlled. For the simulation study, random ground truth data was chosen for X in terms of its translation vector t_X, t_Y, t_Z and rotation (Euler) angles R_X, R_Y, R_Z. These values were then converted to the required homogeneous transformation matrix. The same procedure was followed for generating the various robot pose data B and the position of the world coordinate W. The camera pose A_i for each robot pose B_i was then calculated using $A_i = WB_iX^{-1}$.

3.4.3 Pose error generation

During hand-eye calibration of an actual robot-camera system, the two sources of errors are from the robot and the camera. Because the data are in the form of poses, these errors must also pose errors, which can be interpreted as the transformation δ_B that moves the robot hand from their measured position $\hat{B}$ to their actual position B, such that $\delta_B = \hat{B}^{-1}B$. For the camera motion, the error δ_A is the transformation that moves the camera from its expected position A to its measured position $\hat{A}$ such that $\delta_A = A^{-1}\hat{A}$. During the calibration operation, the robot poses are obtained from the robot's forward kinematics, which is generally available from the robot control interface or pendant. As such only the measured pose of the robot is available. However, the robot pose error can be modeled by reflecting it in the camera pose measurements. This error in camera pose δ_{A_B} from the reflection of robot pose error δ_B can be expressed as $\delta_{A_B} = X\delta_B X^{-1}$. Hence, during the simulation study, defining total simulation pose error $\delta_e = \delta_{A_B}\delta_A$, then the following equation, $A\delta_{A_B}\delta_A X = XB$ can be used to estimate the hand-eye transformation X in the presence of pose error in the robot and camera, with A and B are the ideal relative camera and robot poses respectively.

3.5 Results and discussions

3.5.1 Simulation study

In the simulation study, the robot and camera pose data as described in the previous section was generated. In the simulation studies the level of noise in the calibration process is represented by

introducing additional random motion offsets with a specified standard deviation as this represents the variability in the motions of the robot and camera. Hence, throughout this study, the noise in the system will be referred to by the standard deviation. The robot poses were based on a uniform distribution, such that the Euler rotations $[\theta_X, \theta_Y, \theta_Z] \in \cup(-180, 180)(deg)$ and translation $[t_X, t_Y, t_Z] \in \cup(-1000, 1000)(mm)$. To study the sensitivity of various algorithms to noise in the robot and camera pose measurements, random noise poses with Gaussian distribution in the rotation based on the Euler angles (deg), and translation (mm) with zero mean μ and varied the standard deviation σ was generated. The converted homogeneous transformation noise δ_e was then added to the pose data. The simulation was conducted by executing 100 simulation runs at each estimation step, sampling the noise from its Gaussian distribution. The choice of 100 simulation runs follows from [75, 141] and provides a trade-off between total simulation time and statistical significance that arise from a large number of experiments. All simulations are based on a Python implementation of the algorithms and evaluation techniques running on a Windows PC with an Intel i7-2.7GHz CPU and 16GB of RAM.

3.5.2 Effect of number of robot motions

The performance of the various algorithms was evaluated based on the number of robot poses used for the calibration. For this study, a total of 100 simulation runs was performed while keeping the standard deviation of the rotation σ_r and translation σ_t noise fixed at 0.5 and 1, respectively. Given that the minimum number of robot poses for a valid computation of the hand-eye parameter is 3 [47], the number of robot poses was varied from 3 to 200. Figure 3.3 shows the

result of the simulation using the relative transform error evaluation E_{rn} plotted against the number of robot motions (or poses) used for the calibration operation.

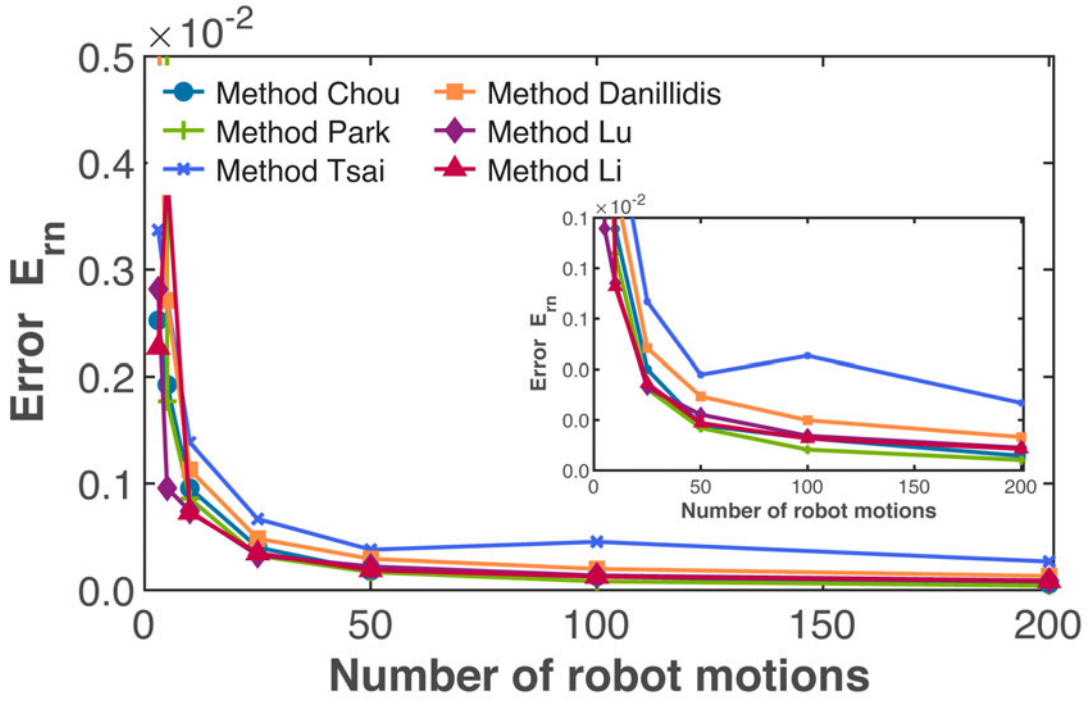


FIGURE 3.3: Effect of the number of robot motions on relative transformation error $E_{rn}(\sigma_r = \sigma_t = 0.5)$. The inset shows the zoomed-in accuracy level between E_{rn} of 0 and 0.001.

From Figure 3.3, it can be observed that for an increasing number of robot motions used in the calibration operation, all the evaluated algorithms show an increase in the accuracy given by the relative transformation error. Furthermore, the result makes it evident that as the number of robot motions used increases, the gain in estimation accuracy becomes minimal. For the number of 3 to 50 robot motions, the result shows a significant drop in the error. However, after 50 robot motions, only a minimal decrease in the error is observed. For other simulation studies, the number of 100 robot motions will be used as it is evident that all the algorithms perform better at a higher number of robot motions.

3.5.3 Effect of rotation noise

For this simulation study, the aim is to observe how the rotation and translation components of the estimated calibration parameter are affected by noise from the rotation component alone. The number of robot motions was set at 100 and varied the standard deviation of the rotation noise σ_r from 0 to 2 without any translation noise ($\sigma_t = 0$). For each noise sampling, a total of 100 simulations was performed and the mean absolute rotation and translation errors were computed. Figure 3.4A and 3.4B show the result of the simulation, where it can be observed that the accuracy of the rotation estimates based on absolute rotation error decrease with increasing rotation noise as expected. However, from Figure 3.4A, the rotation estimates based on Method Daniilidis showed the best performance with increasing rotation noise. While the performance of Method Chou and Method Park are not far off from Method Daniilidis, that of Methods Li and Lu which were similar became significantly worse as the rotation noise increases. The performance of Method Tsai on the other hand appeared to be very sensitive to rotation noise and provided large rotation error even at lower rotation noise.

Considering the effect of the rotation noise on the estimates of the translation component of the calibration parameter, Figure 3.4B suggests that the translation estimates provided by Method Daniilidis, Method Chou, and Method Park based on absolute translation error also remained more robust to translation noise than the other methods in the absence of rotation noise, with Method Daniilidis showing slightly better translation estimates. The translation estimates of Method Li, Method Lu, and Method Tsai progressively

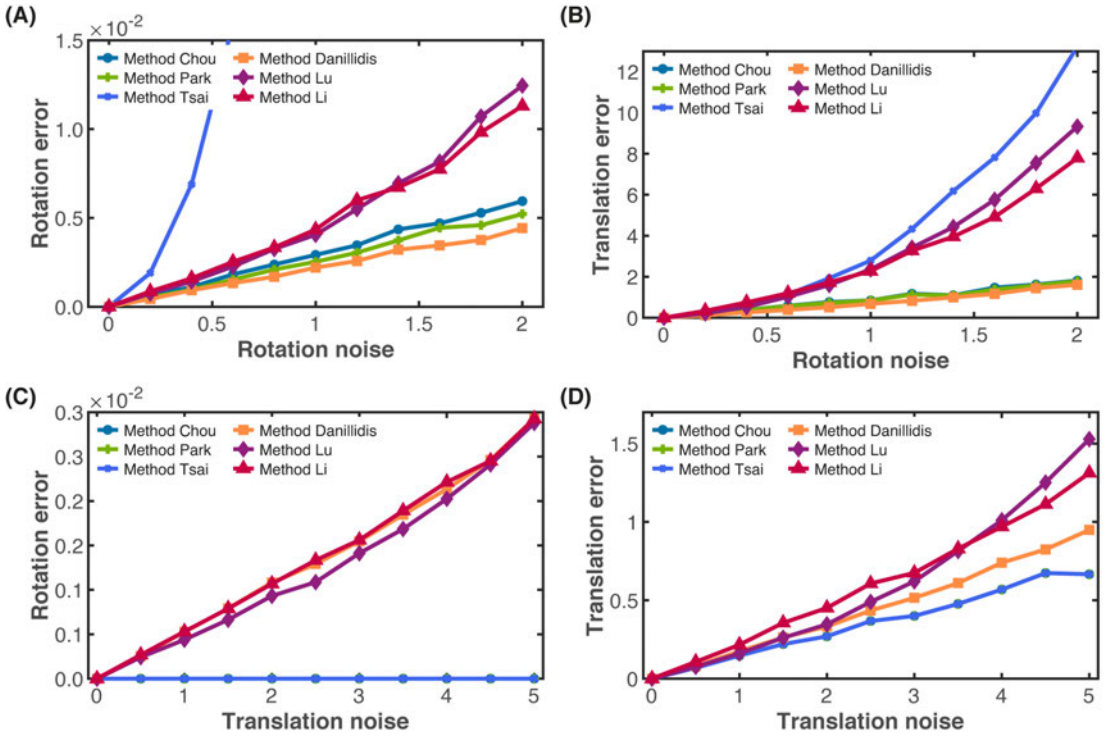


FIGURE 3.4: Effect of rotation and translation noise on rotation and translation calibration accuracy. (A) Effect of rotation noise on rotation accuracy. (B) Effect of rotation noise on translation accuracy. (C) Effect of translation noise on rotation accuracy. (D) Effect of translation noise on translation accuracy.

became worse as the rotation noise increased, with the latter providing the best translation estimate of the three at low rotation noise ($\sigma_r < 0.5$). At higher rotation noise, however, the translation estimates of Methods Tsai became the worst. Since the only noise present is from the rotation component, the errors in the translation components are propagated from the rotation components as argued in numerous literature on hand-eye calibration as a need for simultaneous solution [118, 134, 142]. However, it becomes apparent that for simultaneous methods, as seen in the performance of Method Daniilidis and Method Lu (Figure 3.4B), errors can be induced in the translational component as well in the presence of rotation error.

3.5.4 Effect of translation noise

To study the effect of translation noise on the calibration accuracy, 100 motions of the robot with no rotation noise ($\sigma_r = 0$) was used while varying the standard deviation of the translation noise σ_t from 0 to 5. One hundred simulation runs were performed and the mean absolute rotation and translation errors were calculated as shown in Figure 3.4C and 3.4D. Figure 3.4C shows the accuracy of the rotation estimate in the presence of translation noise. Based on the observed result, the separated methods (Method Chou, Method Park, and Method Tsai) show robustness against the translation noise from the robot motion. This is expected as the rotation parameter is computed without the translation component. For the methods with the simultaneous solutions (Method Danillidis, Method Lu, and Method Li), the result shows increasing error in the estimated rotation with an increase in the translation noise. Amongst the three simultaneous methods evaluated, Method Daniilidis and Method Li showed similar performance, however, Method Lu provided the best rotation estimates under translation noise as the only source of the noise. From the point of view of the translation estimate as seen in Figure 3.4D, while all the separated methods had similar translation accuracy, as the translation noise increases from a variance of 0, the accuracy level of the simultaneous solution methods became progressively worse compared to the separated methods, with Method Daniilidis showing the best performance of the three simultaneous methods. The superior performance of the separate methods compared to the simultaneous methods is attributed to the estimation of the translation parameter with least-square on a linear system rather than the non-linear system provided by the simultaneous methods.

3.5.5 Combined effect of noise on the rotation and translation estimates

In the previous section, the effect of the noise from the rotation and translation components on the estimated rotation and translation parameters was evaluated, where each noise source acted alone. The results suggest that the simultaneous method of Method Daniilidis and the separated methods of Method Chou and Method Park are more robust to noise in rotation and translation with Method Daniilidis slightly better. Method Tsai on the other hand appeared to be extremely sensitive to high rotation noise levels while Methods Li and Lu showed roughly similar performance. These results give an idea of the sensitivity of the different algorithms to noise from each of the components, however, in reality, the algorithms would have to handle the combined noise from both sources, which is not a linear function. To evaluate the sensitivity of the different algorithms to the noise from the rotation and translation components acting together, both the rotation and translation noise variance with $\sigma_r = (0, 2)$ and $\sigma_t = (0, 5)$ respectively were simultaneously increased .

The result of this evaluation based on the average of 100 simulation runs is shown in Figure 3.5A and 3.5B. From Figure 3.5A, with increasing rotation and translation noise, Method Daniilidis and Method Park showed roughly similar and better performance than the others with Method Chou only slightly worse. On the other hand, Method Li and Method Lu again showed similar performance, with Method Lu slightly better at lower joint rotation and translation noise levels ($\sigma_r < 1.4, \sigma_t < 3.5$), while Method Lu appear slightly better at higher noise levels. Method Tsai, as in the previous evaluations showed large rotation errors as the noise levels increased.

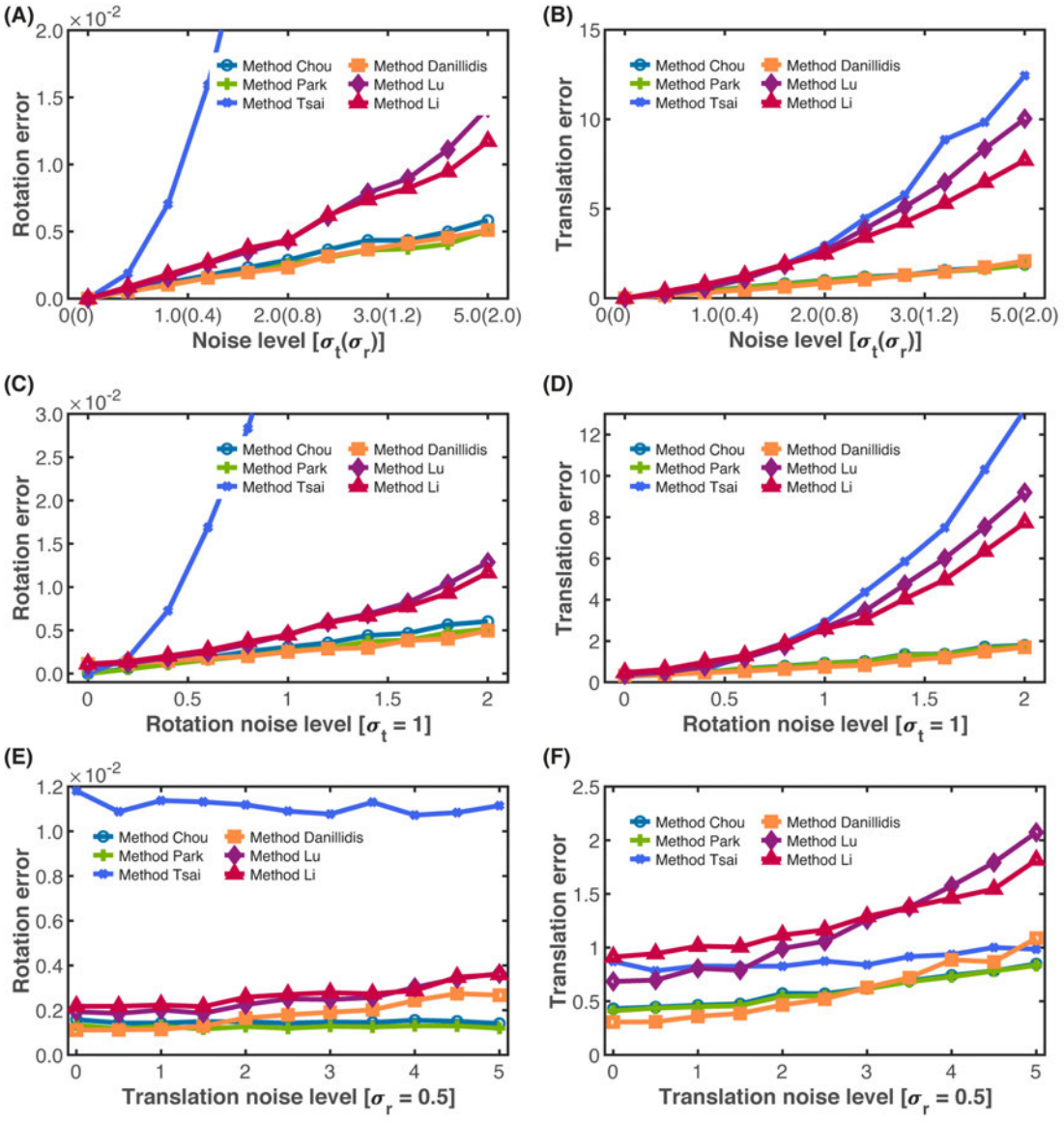


FIGURE 3.5: Effect of combined rotation and translation noise on calibration accuracy. (A) Effect of increasing rotation and translation noise on rotation accuracy. (B) Effect of increasing rotation and translation noise on translation accuracy. (C) Effect of rotation noise on rotation accuracy with fixed translation noise σ_t. (D) Effect of rotation noise on translation accuracy with fixed translation noise σ_t. (E) Effect of translation noise on rotation accuracy with fixed rotation noise σ_r (F) Effect of translation noise on translation accuracy with fixed rotation noise σ_r.

From Figure 3.5B, Methods Daniilidis, Park, and Chou again show the best performance for translation estimates. However, Method Daniilidis was slightly better at lower rotation and translation levels ($\sigma_r < 1, \sigma_t < 2.5$). With increasing rotation and translation noise levels, the estimated translation errors of Methods Tsai, Lu, and Li increases progressively. Of these three methods, Method Tsai

proved the best at lower noise levels ($\sigma_r < 0.8, \sigma_t < 2$) but the worst at higher noise levels. Method Li on the other hand showed the worst performance at lower noise levels ($\sigma_r < 0.8, \sigma_t < 2$), but the best at higher noise levels.

To get more dynamic insights into the performance, the noise variance of the rotation, or the translation components was set to a fixed value while varying the other. The results based on the average of 100 simulation runs are shown in Figure 3.5C to 3.5F. In Figure 3.5C and 3.5D, the translation noise variance was fixed at $\sigma_t = 1$ and varied the rotation variance from $\sigma_r = 0$ to $\sigma_r = 2$, while in Figure 3.5E and 3.5F, the rotation noise variance was fixed at $\sigma_r = 0.5$ and varied the translation variance from $\sigma_t = 0$ to $sigma_t = 5$. From Figure 3.5C, it can be observed that for the same translation noise, all the separated methods – Method Chou, Method Park, and Method Tsai - showed better performance at lower rotation noise ($\sigma_r < 0.6$) than the simultaneous methods. However, as the rotation noise increased, the performance of Method Daniilidis became better than all the separated methods. For the range of noise levels evaluated, Method Daniilidis, Method Chou, and Method Park consistently provided better rotation estimates than the other methods. Methods Lu and Li again showed similar performance but with Method Lu slightly performing better at lower rotation noise ($\sigma_r < 1$) than Method Li, while Method Tsai showed a good performance only at very low rotation noise levels ($\sigma_r < 0.125$). For the translation estimates (Figure 3.5D) all the separated methods again performed better only at rotation noise variance below 0.25. At higher rotation noise levels, the performance of Method Daniilidis becomes better than all the separated methods. While the translation errors of Methods Chou and Park which were similar were only marginally higher than Method

Daniilidis at higher rotation noise, the translation errors of Method Tsai rose above those of Method Lu and Method Li at rotation noise variance above 0.75. With increasing the translation noise at fixed rotation noise, it can be observed from Fig 6E that the rotation estimates of the separated methods remained relatively stable with Methods Chou and Park showing low rotation error while Method Tsai showed high rotation error. The stable rotation estimate with increasing translation noise at fixed rotation noise is expected as the rotation is estimated without the translation parameter. Hence the error in the separated methods is due only to the rotation error. However, Method Daniilidis showed better rotation estimates than the separated methods at low translation noise ($\sigma_r < 1$) after which its performance degraded further with increasing translation noise. From Fig 6F, it can be noticed that while all the methods showed an increase in translation error with increasing translation noise, the increase in translation error is more pronounced for the separated methods. At high translation noise levels ($\sigma_t > 4.75$) the performance of all the simultaneous methods became worse than the separated methods.

3.5.6 Effect of robot motion range

Here, the aim is to observe how the range of motion of the robot in rotation and translation affects the calibration accuracy. The range of motion refers to how much the robot arm is allowed to move in terms of rotation and translation during the calibration operation. For rotation, it is the average rotation about a random axis. For this simulation, first, the robot translation was restricted to a range $t_f = 50$ mm and varied the rotation around each of the axes from

a range of $R_r = 5$ deg to $R_r = 60$ deg. Here the range x_r of x is defined as

$$x_r = \cup(0.9x, x) \tag{3.23}$$

Secondly, the rotation of the robot motion was restricted to a range of $R_f = 20$ deg while varying the translation motion t_r between 40 mm to 450 mm. The translation range t_r is calculated based on the norm of the translation as

$$t_r = |t|, \tag{3.24}$$

while the rotation range $R_r$is calculated as

$$R_r = Rot(R) \tag{3.25}$$

where R is the rotation matrix. This simulation was done over a total of 100 robot motions with the standard deviation of the rotation and translation noise set to $\sigma_r = 0.1$ and $\sigma_t = 0.5$, respectively. The relative rotation and translation errors were calculated, and the results are shown in Figure 3.6.

As seen in Figure 3.6A, increasing the range of rotation of the robot during hand-eye calibration at a constant translation rate has a marginal effect on the rotation estimates for the separated methods. This increment appears to be more pronounced in Method Tsai from a very low rotation (below 10 deg) than in Methods Chou and Park where the accuracy improvement appears minimal. For the separated methods, however, the accuracy of the rotation estimates decreases when the rotation range is increased at a constant translation range. A similar trend is observed with the translation estimates

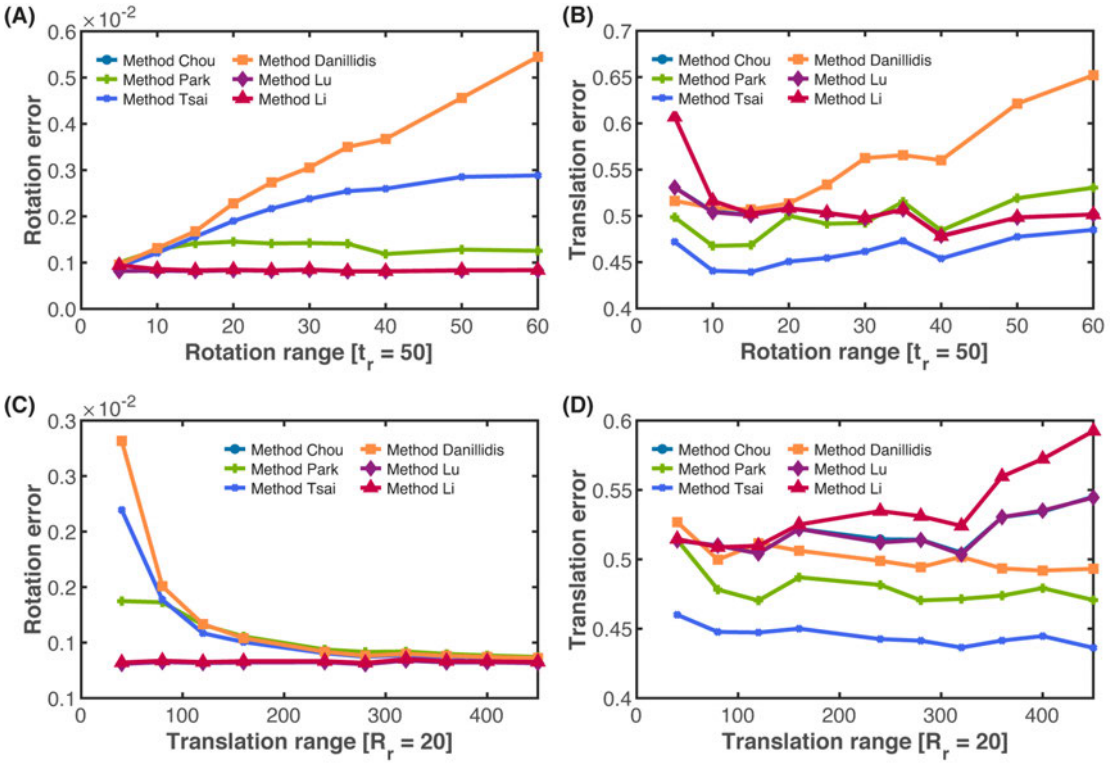


FIGURE 3.6: Effect of rotation and translation motion on estimation accuracy. (A) Effect of rotation motion on rotation accuracy with fixed translation range. (B) Effect of rotation motion on translation accuracy with fixed translation range. (C) Effect of translation motion on rotation accuracy with fixed rotation range. (D) Effect of translation motion on translation accuracy with fixed rotation range

based on the rotation span in Figure 3.6B, which shows a decreasing translation error with increasing rotation motion for the separated methods, while the translation error of the simultaneous methods increased with increasing rotation motion span.

In terms of the effect of the translation motion span on the estimation accuracy, Figure 3.6C shows a significant reduction in the rotation error of the simultaneous methods, while the rotation error of the separated methods increased marginally. Furthermore, from Figure 3.6D, increasing the translation motion span resulted in an increase in the accuracy of the simultaneous methods while exhibiting a decrease in the accuracy of the separated methods.

3.5.7 Simulation time

Here, interest is in the execution time of the algorithms in performing hand-eye calibration. This analysis considers 20 and 100 robot motions and executed 100 simulation runs for each algorithm. The execution time was averaged over the simulation runs. This evaluation is based on a Python implementation of the algorithms running on a PC with an Intel i7-2.7GHz CPU and 16GB of RAM. The result is shown in Table 3.1.

TABLE 3.1: Comparison of algorithm execution time for 20 and 100 robot motions.

	Checkerboard	**Circle**
Algorithms	**Time for 20 robot motions (secs)**	**Time for 100 robot motions (secs)**
Method Chou	0.084	0.444
Method Park	0.062	0.301
Method Tsai	0.079	0.399
Method Daniilidis	0.119	0.558
Method Lu	0.107	0.504
Method Li	0.096	0.887

The result from Table 3.1 suggests that Method Daniilidis is the most computationally expensive in comparison with the other methods for a lower number of robot motions. However, as the number of robot motions increases, the execution time of Method Li increases and is the most computationally expensive compared to the other methods. The large computational time for Method Daniilidis at a number of low robot motions can be attributed to the need to solve a dual variable polynomial. However, Method Li employs Kronecker product which has a quadratic complexity $O(n^2)$, as such its processing time increases progressively with the amount of data. Method Park appeared to be the most computationally efficient method in both scenarios. Interestingly, all the three separated methods are

shown to be more computationally efficient than the simultaneous methods.

3.6 Experimental evaluation with UR5e robot

For the experimental evaluation, a UR5e robot arm was used with a Microsoft Azure Kinect camera mounted on the last link for 2D image acquisition. During the experiment, the robot arm moved to random positions and orientations, and the image of a stationary calibration pattern was captured by the camera from which the poses of the camera with respect to the world were calculated, while the robot poses were obtained from the robot pendant. This procedure was done for 100 different motions of the robot. Figures 3.7A and 3.7B show the span of the rotation and translation motions, respectively with a mean rotation of 44.7 deg and a mean translation of 350.6 mm. The rotation and translation parameters of the hand-eye transformation were calculated from the acquired data using each of the algorithms. The comparison of the rotation and translation errors for the different algorithms under this condition is shown in Figures 3.7C and 3.7D, respectively. Because of the absence of ground truth data for the comparison, the relative rotation E_R and translation E_T errors were used for the evaluation instead.

From Figure 3.7C, the rotation estimated from Method Tsai showed the highest error, while the separated methods of Method Park and Method Chou provided the best estimate of the rotation based on the relative rotation error, with Method Park slightly outperforming Method Chou. All three simultaneous methods had similar rotation performance but were better than the Method Tsai, with the Method

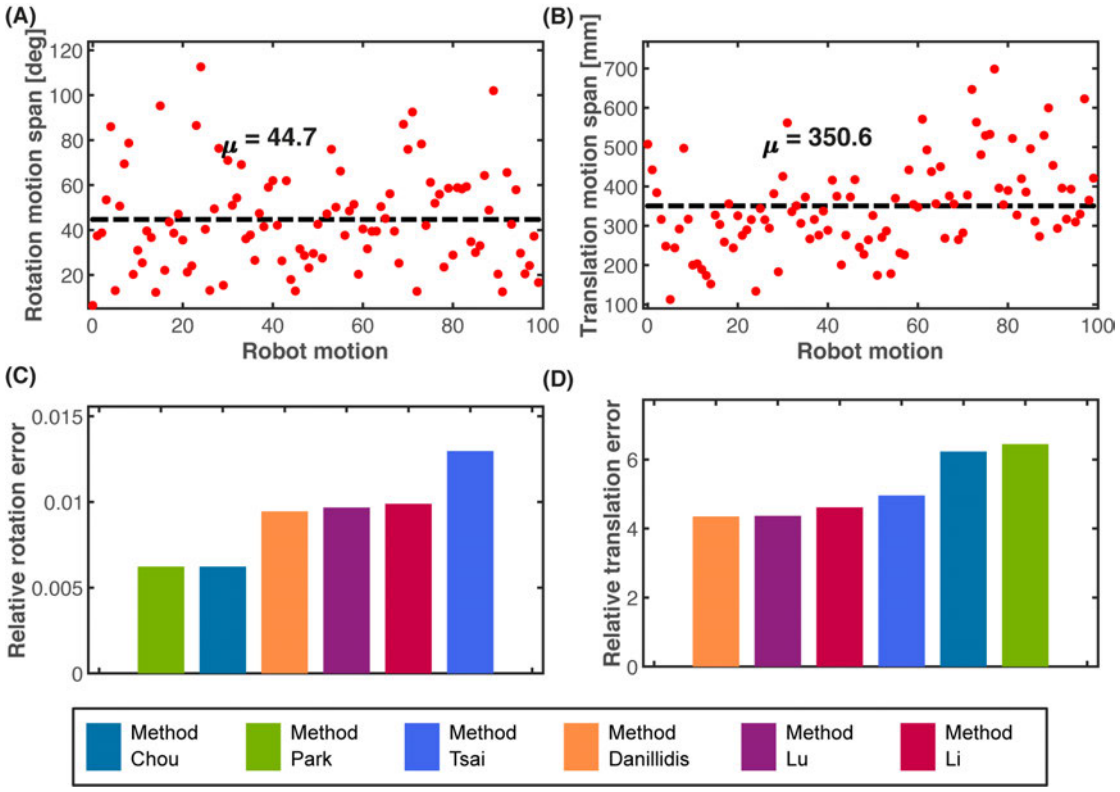


FIGURE 3.7: Random rotation and translation motion of the robot during data collection. (A) Rotation motion span, mean,$\mu = 44.7$ deg. (B) Translation motion span, mean $\mu = 350.6$mm. (C) Relative rotation error. (D) Relative translation error.

Daniilidis slightly outperforming the others. For the translation error (Figure 3.7D, all three simultaneous methods outperformed the separated methods with Method Daniilidis showing the best translation estimate. Method Tsai showed the best translation estimate among the separated methods, followed by Method Chou and then Method Park.

3.6.1 Effect of robot motion range

The aim of the experiment is to observe how the rotation and translation motions in isolation affect the calibration accuracy of the candidate algorithms. Three calibration operations were carried out with the real robot. For each calibration operation, the range of the translation and rotation motions were restricted to different values.

The actual motion range for each motion was allowed to vary a little from the chosen span value. Table 3.2 shows the mean translation and rotation motion ranges for each of the experiments. Experiments 1 and 2 describe a change in the rotation motion span with a fixed translation motion span, while Experiments 2 and 3 describe a change in the translation motion span with a fixed rotation motion span. The results of the experiments are shown in Figure 3.8.

TABLE 3.2: Motion range experiments

Experiments	**Mean rotation motion range (deg)**	**Mean translation motion range (mm)**
Experiment 1	10.6	301.0
Experiment 2	50.4	301.6
Experiment 3	51.0	52.0

Each calibration operation in the experiments is carried out with 20 individual translation motions with the span of each motion shown in Figures 3.8 A, E, and I. Similarly, each experiment also involved 20 individual rotation motions with the rotation span of each motion shown in Figures 3.8 B, F, and J. From the results of Experiments 1 and 2, as shown in Figures 3.8A and 3.8E, respectively, the rotation range increased from 10.6 deg to 50.4 deg while the translation range remained close to 301 mm from Figures 3.8B and 3.8F. During these conditions, the rotation errors for Method Chou and Method Park remained relatively the same, with marginal improvements as shown in Figures 3.8C and 3.8G. However, a significant improvement in the rotation accuracy was observed for Method Tsai as the rotation motion range increased with constant translation motion range. Conversely, an increase in the error of the rotation estimate was observed for all the simultaneous methods as the rotation range increased with a fixed translation range. This increase in the rotation error was more pronounced in Method Li. In terms of the translation estimates, from Figures 3.8D and 3.8H, increasing the rotation

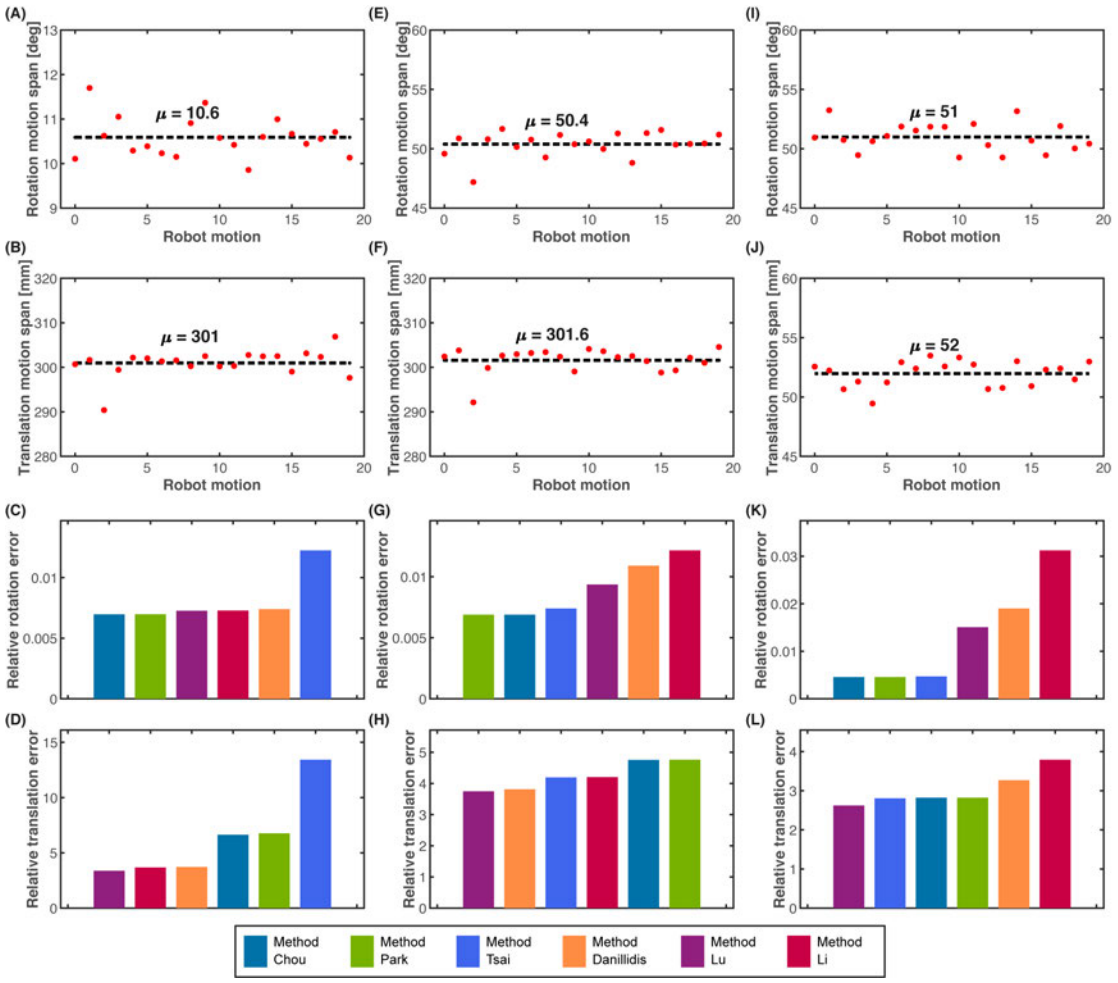


FIGURE 3.8: Motion range experiment: Experiment 1. Low rotation and high translation motion span. (A) Rotation motion span, mean $\mu = 10.6$ deg. (B) Translation motion span, mean $\mu = 301.0$ mm. (C) Rotation estimates. (D) Translation estimates. Experiment 2: High rotation and high translation motion span. (E) Rotation motion span, mean $\mu = 50.4$ deg. (F) Translation motion span, mean $\mu = 301.6$ mm. (G) Rotation estimates. (H) Translation estimates. Experiment 3: High rotation, low translation motion span. (I) Rotation motion span, mean $\mu = 51.0$ deg. (J) Translation motion span, mean $\mu = 52.0$ mm. (K) Rotation estimates. (L) Translation estimates.

range also improved the accuracy of the translation estimates for all the separated methods. The increment was also more noticeable in Method Tsai than in Method Chou and Method Park. However, just like the rotation estimates, Figures 3.8D and 3.8H show that the accuracy of the translation estimates for all the simultaneous methods decreased when the rotation range was increased at the fixed translation range, with Method Li performing the worst.

The results in Figures 3.8F and 3.8J, respectively, show a decrease in

the translation range from 301.6 mm in Experiment 2 to 52 mm in Experiment 3, while the rotation range from Figures 3.8E and 3.8I remains fixed at about 50 deg. From these results, it can be observed that as the translation motion range decreased from 301.6mm to 52 mm and at a constraint rotation motion range, the accuracy of the rotation estimates for the simultaneous methods decreased as seen in Figures 3.8G and 3.8K, with a more pronounced decrease in Method Li. The separated methods on the other hand experienced an increase in the accuracy of their rotation estimates when the translation range decreased with a fixed rotation range.

Observations from the translation estimates in Figures 3.8H and 3.8L show that the translation errors of all the calibration methods increased when the translation range was increased with a fixed rotation range. However, all the separated methods had a much higher increment in their translation errors than and even surpassed the translation errors of all the simultaneous methods. This suggests that the performance of the translation estimates of all the simultaneous methods improved much better than all the separated methods.

3.7 Simulation versus real experiment

The use of simulated data allows deeper insight into the evaluation of the behaviors of the different algorithms, which may not be possible with the use of real data from experimentation. For instance, the availability of ground truth data. However, with real data from experiments, there is the advantage of capturing the true dynamics of the system under test, which may not be completely possible via simulation. For the evaluations in this study, the availability of

ground truth data during simulation allows the comparison based on absolute errors in rotation and translation, which ideally should be the better evaluation metrics. On the other hand, because ground truth data is not available for the real experiment, relative rotation and translation errors were used for the evaluation. Hence there is the expectation of discrepancies in the evaluations, for example, the relative difference in observed simulation errors between the evaluated algorithms compared with the real experiment. The relative errors have also been used for the evaluation of the estimated errors based on the robot motion range to validate the simulation study with the experiment in the absence of ground truth data. Furthermore, as it was observed in the simulation tests, the rotation and translation errors depend on a number of factors. These factors have been evaluated at specific values and ranges during simulation. The total rotation and translation errors as seen from the experimental evaluation are a combination of the errors from each of these factors, which are largely unknown.

3.8 Summary

This chapter comparatively evaluates the accuracy of some of the common hand-eye calibration algorithms based on several factors: the use of simulated datasets and real datasets from experimentation with a physical robot. The result of the comparative study sheds light on how different factors affect the accuracy of estimates based on these methods.

Firstly, the number of robot motions used during the calibration is critical to the level of accuracy of the estimated hand-eye parameters in the presence of noise. Increasing the number of motions

increases the accuracy level. However, as the number of robot motions increases, the improvement in the accuracy achieved becomes minimal. Moreover, the number of robot motions used would impact the execution time, complexity, and computation cost of the different algorithms.

Furthermore, the noise in the rotation and translation motions affects the rotation and translation estimates in different ways in all the evaluated hand-eye calibration methods. While the quality of the estimated translation depends on the estimated rotation parameter for the separated methods, estimating the rotation and translation parameters together as in the simultaneous methods resulted in noise transfer between both parameters.

The results also clearly showed that the accuracy of a hand-eye calibration algorithm would vary substantially with different ranges of motions of the robot during calibration. As such, this factor should be taken into consideration when benchmarking a particular algorithm against other algorithms.

Chapter 4

Accuracy assessment of hand-eye calibration techniques in uncertain environment

4.1 Introduction

The growing need for high-efficiency automation in various industries has led to the rapid growth of utilising robots in the industry's day-to-day routines. Improvements in the flexibility of robotic applications have been achieved by the integration of vision systems. This flexibility enables the robot to sense and react to its environment dynamically based on feedback from its sensors, thereby improving automation and accuracy. However, with the integration of a vision system comes the requirement to find the rigid body transformation between the robot hand and the camera. This requirement is termed hand-eye calibration. As the accuracy of the robot's perception of the surrounding depends to a large extent on the accuracy of hand-eye calibration, extensive research has been done in this area to improve the accuracy of hand-eye calibration. In Chapter 3, the accuracy of six commonly used calibration methods was investigated

and showed good performance in terms of small rotation and translation errors. However, that investigation was carried out assuming no uncertainties in the robot operating environment. One common attribute that is also an inherent limitation of all these methods is that they require a set of robot pose data with the corresponding camera pose data for evaluation of the hand-eye calibration parameters, after which the robot is put into service. In other words, these methods are categorised as passive calibration.

In noisy or uncertain environmental conditions, where uncontrolled external conditions, e.g. vibrations, slips due to component wear from prolonged usage or shock from abrupt changes in robot motions [143], etc. are profound, the prolonged usage of the robot is bound to affect the calibration parameters. This can be manifested by a change in the pose of the camera for example, and a recalibration would have to be done to maintain the accuracy of the robot's operation. One critical downside to this is the cost of operational downtime as a result of obtaining a new set of calibration parameters due to recalibration. An interesting proposition would involve the elimination of operation downtime by allowing the robot to automatically recalibrate itself while in operation by way of active calibration. In this paper, the need for considering active calibration was experimentally justify by looking at the effect of the changes in the calibration parameters as could occur in uncertain environmental conditions on the accuracy. In addition, some expectations and possible challenges of active hand-eye calibration are also highlighted.

4.2 Assessment of the effect of changes in calibration parameters

The majority of the work on hand-eye calibration is focused on offline solutions based on the processing of batch data acquired from the robot and vision system as illustrated in Figure 4.1. This is referred to as passive calibration, as this method of calibration does not take into consideration any unforeseen or unintended changes (i.e., real-time information) in the setup, which would affect subsequent data acquired after the calibration is complete. This often limits the accuracy of the hand-eye calibration with time as the robot is put into operation.

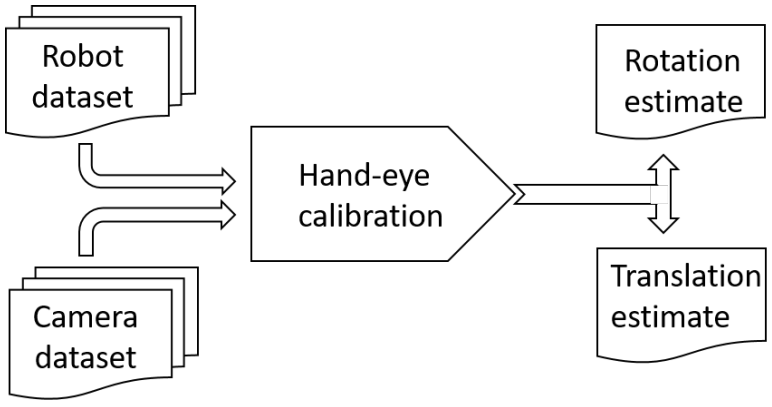


FIGURE 4.1: Passive calibration

In a previous study in Chapter 3, the accuracy of six commonly used passive hand-eye calibration methods in the field was evaluated. Specifically, it only considers the effect of rotation and translation noise on the calibration offline prior to putting the robot into operation. These six methods are termed Method Tsai [51], Method Chou [53], Method Lu [67], Method Park [52], Method Daniilidis [69], and Method Li [135]. The accuracy of those methods under changes in calibration parameters due to uncertain environmental

conditions was not evaluated. Given that these changes in the calibration parameters could have a significant impact on the overall accuracy, in this study these six methods were reevaluated under this new circumstance. In the following section, the experimental setup, the introduction of the changes, and the metric used for the evaluation are discussed.

4.3 Method

To assess the impact a change in the hand-eye parameter has on the calibration accuracy, 100 relative robot pose data and corresponding relative camera pose data were collected. The robot pose data was from a floor-mounted UR5e robot, while the camera pose data was collected from images captured by an Azure Kinect camera mounted on the last link of the robot arm. A wall-mounted 32mm, 11×8 checkerboard pattern served as the calibration target which enabled the estimation of the camera pose relative to the world via the P-n-P algorithm. Control of the robot's motion was achieved via a RoboDK interface running a Python script. The use of RoboDK interface ensures the ease of implementation through a high-level robot interaction. To avoid singularity in calibration, software checks were implemented to restrict the robot's movement to 180 degrees in rotation [47, 51]. Figure 4.2 depicts the overall setup. A total of 101 robot poses data and images of 1280×720 pixels are acquired for this evaluation of this study.

As a remark, an asymmetrical checkerboard pattern with an odd number of rows and an even number of columns is used when computing the camera pose as opposed to using a symmetrical one given the difficulty to achieve reliable tracking of the checkerboard origin

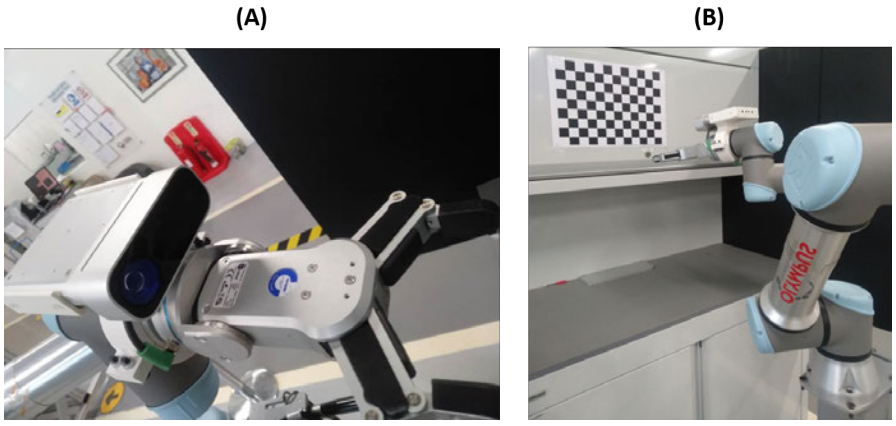


FIGURE 4.2: Hand-eye calibration setup. (A) Camera-Hand setup. (B) Experiment setup.

coordinates in the latter one. The inability to track the checkerboard origin would introduce errors in estimating the camera pose when the camera undergoes sufficient rotation as established in Chapter 3.

4.3.1 Introducing Parameter Change

A change in the calibration parameter can be observed by a change in the pose of the camera relative to the robot's hand. Therefore, in this experiment, the changes in the calibration parameter were introduce by simply applying an appropriate shift in the relative camera pose that would result in the movements in the camera. This shift in the relative camera pose simulates instances such as a slip in the camera position due to shock from abrupt robot motions, vibration, etc.

Given a relative camera pose A as a result of a relative robot pose B and a calibration parameter X, assuming X is non-singular, A can be expressed as

$$A = XBX^{-1}. \tag{4.1}$$

A pose change δ_X in the calibration parameter would result in a different value for the relative camera pose $\hat{A}$ compared with the initial value such that

$$\hat{A}\delta_X X = \delta_X X B, \tag{4.2a}$$

$$\hat{A} = \delta_X X B X^{-1} \delta_X^{-1}, \tag{4.2b}$$

$$\hat{A} = \delta_X A \delta_X^{-1}. \tag{4.2c}$$

assuming δ_X is invertible. Hence, given an initial relative camera pose A, a shift in the calibration parameter by δ_X would result in a relative camera pose of $\delta_X A \delta_X^{-1}$ under the same robot motion. During the evaluation in this study, the pose changes were added artificially to the acquired robot and camera pose data.

4.3.2 Evaluation

For the performance assessment of the various methods, the relative rotation E_θ and translation E_t errors were used. These are derived from (2.8a) and (2.8), respectively, and they are given by

$$E_\theta = \frac{1}{N}\sum_{i=1}^{N} \cos^{-1}\left(\frac{\mathrm{Tr}\left[(R_{A_i}R_X)^{-1}R_X R_{B_i}\right]-1}{2}\right), \tag{4.3a}$$

$$E_t = \frac{1}{N}\sum_{i=1}^{N} \left|R_{A_i}t_X - t_X - R_X t_{B_i} + t_{A_i}\right|, \tag{4.3b}$$

where the notation Tr denotes the trace of the matrices and N is the number of relative pose data used for the experiment

4.4 Results and discussions

4.4.1 Fixed calibration (without uncertain environment)

Figure 4.3 shows the result of the estimated hand-eye calibration parameters when the calibration parameters are fixed. Under this condition, the calibration error remains low with a range of 0.1322 to 0.2793 degrees for rotation and 0.8787 to 1.1122 mm for translation between the different methods. This accuracy would remain consistent within the reported ranges provided the calibration parameters remain unchanged during the operation of the robot.

4.4.2 Change in calibration parameters (with the uncertain environment)

Figure 4.4 shows the result of the estimated hand-eye calibration parameters when the calibration parameters are changed. It is worth mentioning that the error at parameter shift 0(0) in Figure 4.4, represents the case when no shift in the camera occurs and this is identical to the results in Figure 4.3.

From Figure 4.4 it is obvious to note that majority of the error in the estimated calibration parameters increase in an exponential manner with the changes in the calibration parameters. For Method Lu, both the rotation and translation errors diverge at parameter shift 4(1.0) suggesting the sensitivity of this method to large changes (i.e., beyond 1 degree and 4 mm) in the parameter. Methods Daniilidis and Li, where in the absence of change in calibration parameters have the highest rotation errors of 0.2514 and 0.2793 degrees respectively, appeared to be more robust than the other methods in terms of the rotation estimates. However, Method Li induces a large translation

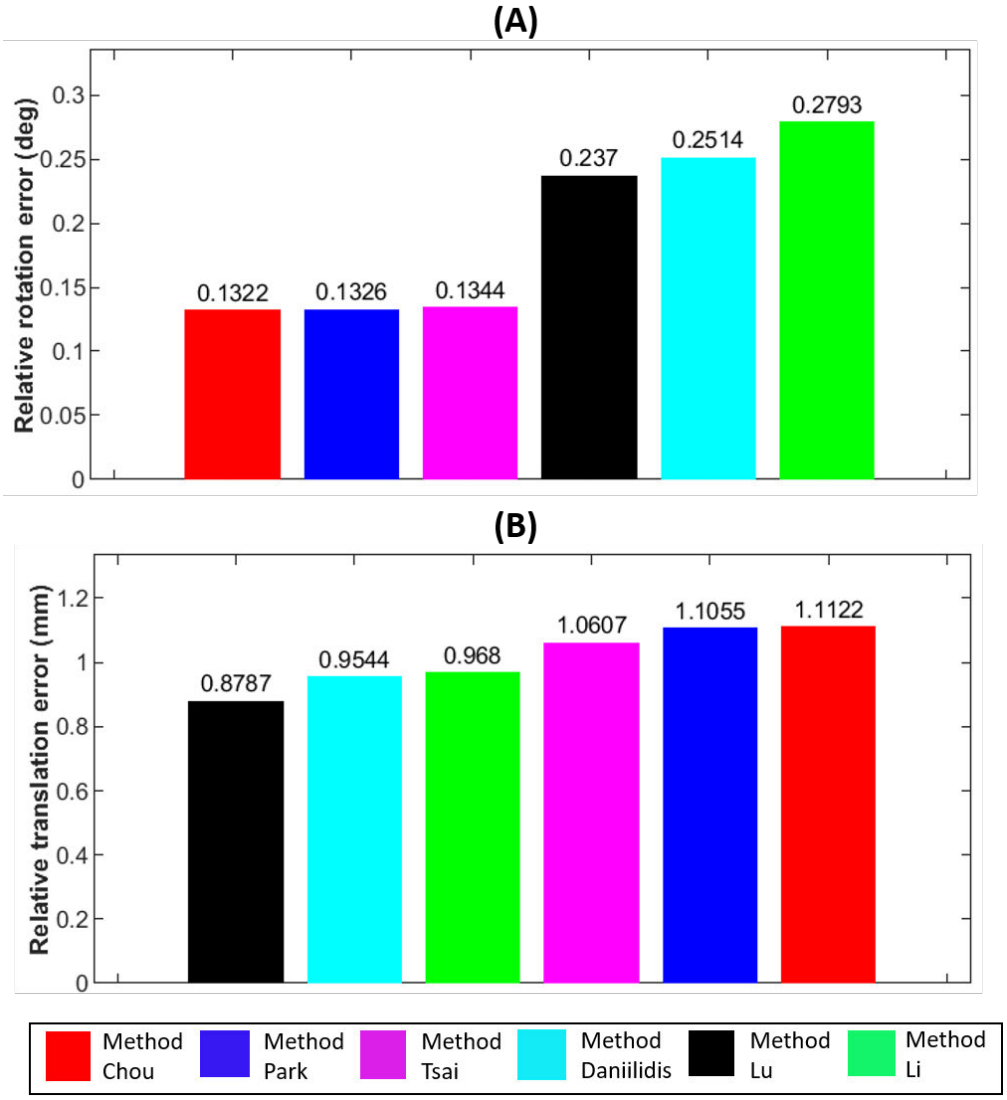


FIGURE 4.3: Rotation and translation errors for unchanged calibration parameters. (A) Relative rotation error. (B) Relative translation error. The values shown on top of the bar graphs are for ease of reading purposes.

error compared to the remaining four methods. Method Daniilidis also shows better performance than the other methods on the translation estimates.

Figure 4.5 highlights the extent to which a change in the calibration parameters can have on the accuracy of the hand-eye calibration. In Fig. 4.5, the relative change in the observed error when the camera moves 10mm and 2.5 degrees is shown. The relative change Δ_e from an initial error value to a final error value is computed as,

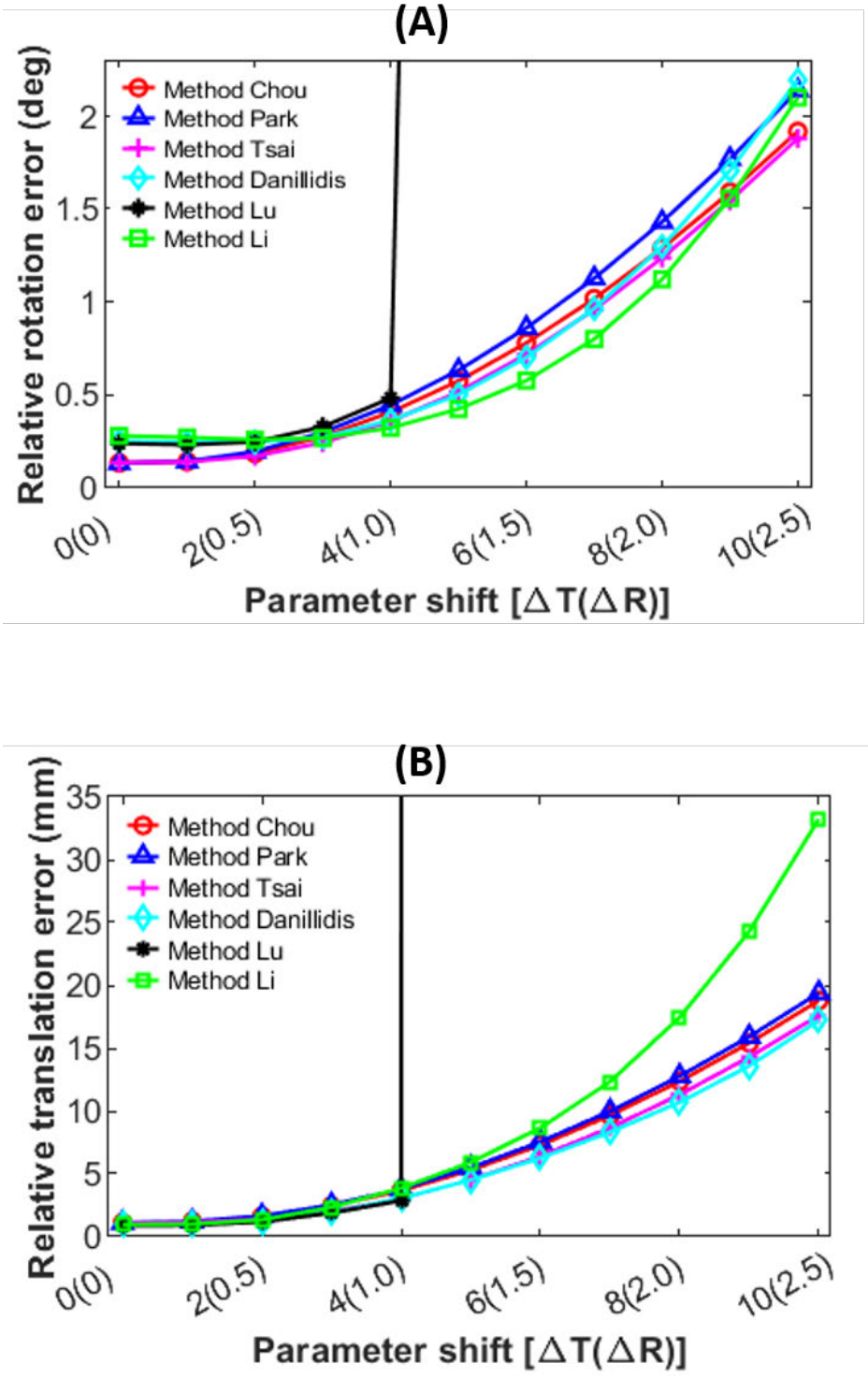


FIGURE 4.4: Rotation and translation errors for unchanged calibration parameter. ΔR is the change in the rotation of the camera relative to the robot hand about a random axis while ΔT is the norm of the change in the translation of the camera relative to the robot hand. (A) Relative rotation error. (B) Relative translation error.

$$\Delta_e = \frac{E_{p,f} - E_{p,i}}{E_{p,i}}. \tag{4.4}$$

where the first subscript index p denotes either θ or t with E_θ and E_t as defined in Equation (4.3), while the second subscript indexes i and f represent, initial and final values, respectively.

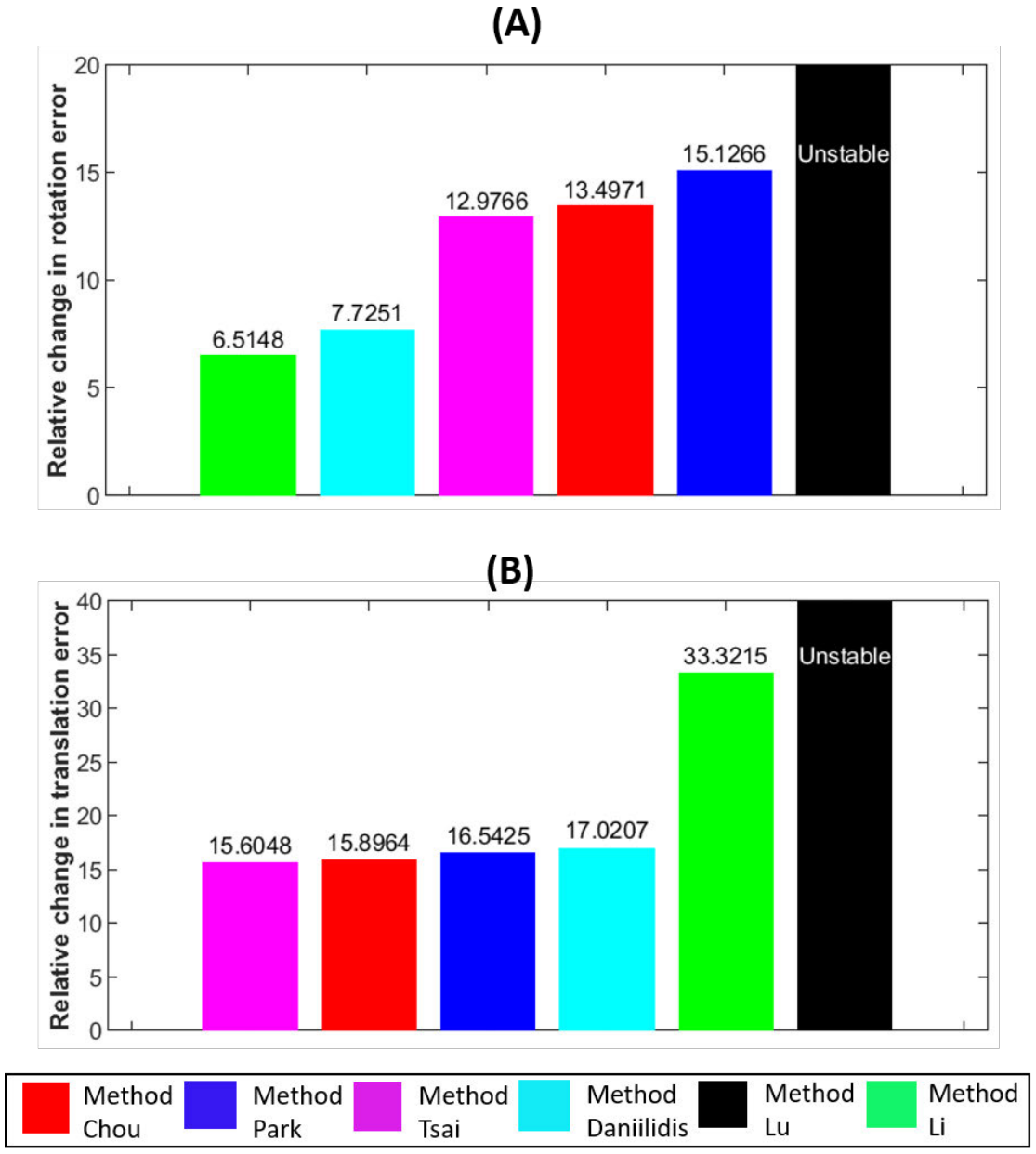


FIGURE 4.5: Relative change in rotation and translation errors when $\Delta T = 10$mm, $\Delta R = 2.5$ degrees in calibration parameter. (A) Relative change in rotation error. (B) Relative change in translation error. The values of the relative change calculated using Equation (4.4) are shown at the top of the bar graphs.

There are several interesting observations from this figure. Firstly, compared with Figure 4.3 when the calibration parameters do not change, all the methods show significant changes in the rotation and

translation errors. While the increase in error is expected, what is unexpected is the substantial increase in the amount of error, in particular with Method Lu where it diverges. Secondly, the accuracy of each method changes with parameter change suggesting despite having better accuracy initially, their robustness changes in the presence of uncertain environmental conditions. Method Li showed the least relative change in rotation error at 6.5 while Method Tsai showed the least relative change in translation error at 15.6 under this condition.

4.4.3 Discussions

The results from this experiment show that the calibration methods employed can provide good levels of accuracy as seen in Figure 4.3. However, the accuracy of these methods cannot be guaranteed, especially in an uncertain environment where external forces such as vibrations, etc, are profound. In these scenarios, with prolonged usage of the robot, the accuracy is likely to further degrade as illustrated in Figure 4.4. To ensure accuracy in the calibration, it would be necessary to keep track and update the calibration parameters as necessary by way of active hand-eye calibration. Active hand-eye calibration does not perform a one-time calibration with batch data. Rather, the calibration estimates are continuously updated while the robot is in operation by continuously acquiring the robot and camera pose data, as illustrated in Figure 4.6.

This calibration technique ensures that any unforeseen changes in the calibration parameters that may occur due to prolonged use of the robot in a harsh environment, or movement in the position of the camera due to uncontrolled vibration, etc., are taken into consideration in the computation of the estimates for the hand-eye calibration

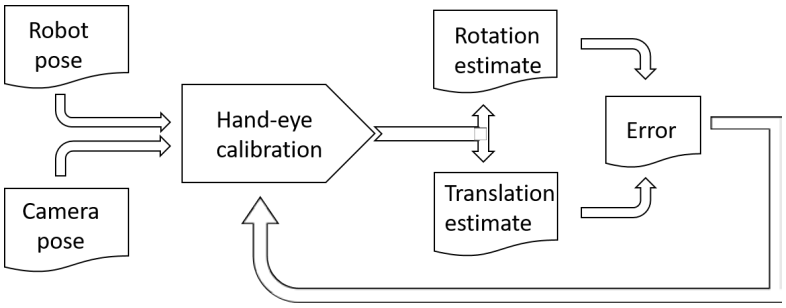


FIGURE 4.6: Active calibration

parameters. This would greatly improve the robustness as well as the accuracy of the calibration process.

To reliably achieve active calibration, two main challenges of the system have to be addressed. These are an online estimation and camera pose estimation.

1. Online estimation: The online estimation of the hand-eye calibration parameters (rotation and translation) requires a system of continuously estimating the parameters based on only the currently available robot and poses data. Note that it is usually not possible to obtain a solution to the passive hand-eye calibration problem with one data point due to noise [47]. However, for online estimation, a recursive estimator can be employed such that the error based on a certain cost function can be used to update and improve the initial estimates. A continuous estimation, correction, and update step as more data becomes available would lead the estimator to eventually converge. The recursive estimation ensures that the quality of the estimates is continuously tracked. As such, in the event of a change in the calibration parameters during operation, the error is corrected and the estimator would converge at the new values.

2. Camera pose estimation: For passive hand-eye calibration, the robot with the attached camera is moved to different poses in steps such that a calibration target remains in view of the camera at each step. The image of the captured calibration target at each step can be used to estimate the camera pose using the P-n-P (Perspective-n-Point) algorithm [16] from which the hand-eye calibration parameters can be estimated. For active hand-eye calibration, however, this poses a challenge as the pose of the camera needs to be computed for each robot pose acquired for the entire time the robot is in operation. Nevertheless, there are several suggestions that can be employed to address this challenge.

   - Scene markers: Markers such as ArUco targets described in Chapter 2 enable the direct estimation of the pose of a camera relative to the world. Their encoded pattern makes them easy to track, extract and process from the view of the camera. While this makes them suitable for active calibration, the issue of occlusion or the markers being out of view of the camera can provide additional challenges. One possible way to solve this problem by leveraging their concise nature is to use multiple markers placed in strategic positions around the work setup. For this to work, at least one of the markers must be visible in two consecutive views of the camera as the pose of the camera relative to a particular marker in each view would be required to obtain the relative pose of the camera between the motions from which the views are captured [144].
   - Structure from motion (SFM): The use of scene markers can make the deployment of active calibration more complex as

the motion of the robot would have to be constrained such that the markers are always in the camera view. The relative camera pose can instead be estimated using the SFM algorithm [17]. It is based on finding matching point correspondence between two consecutive images, which is the greatest challenge and a very active research area in computer vision. The relative rotation and translation of the camera can be estimated using the positions of the matched points in both images under the epipolar constraints. Common algorithms like Sift, Surf, Brief, Orb [145] have been used to obtain point correspondence in images, however, their performance can be affected by speed, illumination, image distortion, etc [146]. More recently, deep learning models have been used for point matching between consecutive images providing good results for different scenarios [147, 148].

- Deep learning: To eliminate the need for finding matching points in the image, deep learning models have been used to directly estimate the relative motion of a camera just feeding in consecutive images of the camera [18, 149]. This method employs only cues from the changes in the scene between both images due to the motion of the camera to predict the relative camera pose and greatly simplifies the acquisition process. However, just like in using the Structure from Motion algorithm, where the acquisition process is based on the scene features on the image, this technique can be limited to static scenes.

4.5 Summary

In this chapter, the accuracy of hand-eye calibration techniques in uncertain environments for vision-guided robots was investigated using six commonly used passive calibration methods. When there is no change in calibration parameters, these six methods are able to provide a good estimate as shown by their small rotation and translation errors estimate. However, when changes in calibration parameters are introduced, the errors increases in an exponential manner with some method even diverging. For this reason, vision-guided robots that are calibrated through passive calibration would periodically require recalibration. This indicates that there is a need to mitigate and improve that accuracy in the presence of an uncertain environment, thereby suggesting a need to explore active calibration techniques, which have not received much attention with respect to hand-eye calibration. Active calibration would be able to compensate for the changes in the calibrated parameter to ensure that the accuracy of the vision-guided robot remains high.

Chapter 5

Self-recoverable Hand-eye Calibration for Vision Guided Robots using a Guided Particle Hybrid Filter and Genetic Algorithm-based Resampling

5.1 Introduction

A significant limitation of most hand-eye calibration algorithms such as those evaluated in Chapter 3 is the requirement for offline or batch calibration and these are referred to as passive hand-eye calibration. In the passive hand-eye calibration method, once the calibration process is complete, the robot is fed with the calibration parameters that it operates with. This form of calibration does not take into consideration of the surrounding changes that could occur while the robot is in operation. Thus, this makes them sensitive to changes in the calibration parameters as a result of operational uncertainties such as vibrations, slips due to component wear from prolonged usage, or shock from abrupt changes in robot motion. The presence of these

issues impacts the accuracy of the robot's operation, which necessitates the need to take the robot offline for recalibration resulting in expensive operational downtime.

For the long-term operational guarantee of the accuracy of the robot's operation, it is important that the robot is able to adapt to and recover from any unforeseen changes to the calibration parameters. This would enable the robot to estimate the new hand-eye parameters for continued operation, thus eliminating the need for taking the robot offline for recalibration. This approach is referred to as active hand-eye calibration. To the best of the authors' knowledge, a limited set of works [150, 151] are available on this topic. Moreover, the majority of online hand-eye calibration works are focused on automating the calibration process offline rather than providing a pipeline for self-calibration during the robot's operation.

To attain the objectives of a recoverable filter, an online filter needs to be developed. One way this can be achieved is the implementation of a recursive filter such that information about the current state of the filter can be continually acquired and improved upon until an accurate state is achieved. This would allow the filter to recover to an accurate state from an inaccurate state. A popular filter that can be employed in this context is the Kalman filter. However, due to the nonlinearity of the calibration equation, the Kalman filter would be highly inadequate as it is well known for solving problems involving linear systems. However, a variant of the Kalman filter, the Extended Kalman filter is widely used for nonlinear systems [152]. This, however, requires linearising the system about the current state. As such, in highly non-linear systems as is the calibration problem, the Extended Kalman filter proves ineffective. To tackle the problem of nonlinearity in the recursive estimation of the

calibration parameters, I propose to employ the particle filter which is suitable for nonlinear, non-gaussian, multimodal systems.

While the particle filter solves the problem of nonlinearity in the calibration equation, a major limitation is the curse of dimensionality which limits its application to low-dimensional systems. The computational complexity of the particle filter increases with the dimensionality of the system [153]. Hence the implementation of the particle filter for high dimensional systems results in high computational resources that may render its use counterproductive. As the calibration problem involves a 7-dimensional system, I propose to take advantage of the fact that this can be expressed as rotational and translational sub-systems. With the rotational nonlinear subsystem with a dimensionality of 4 handled by a particle filter estimation, the linear translational subsystem therefore lends itself to a simpler estimation algorithm like the Kalman filter which also takes advantage of its linearity. Furthermore, a particle transition scheme is implemented to optimise the particle transitions and limit the effect of sample impoverishment common with particle filters. This is followed by a genetic algorithm-based resampling strategy for limiting the effect of particle degeneracy without inducing sample impoverishment. The output of the particle filter at each step is fed to the Kalman filter for the estimation of the translation parameter. Furthermore, to improve the translation estimate, an iterated state update in the Kalman filter is employed. The algorithm can be deployed in an online fashion, running simultaneously with the robot's operation, and can detect and recover from any changes in the calibration parameters. In addition, from the experimental results, it is shown that the algorithm also stands out when compared to other offline calibration algorithms highlighting its suitability for

both on- and offline calibration.

5.2 Particle filter

Particle filter uses a set of particles to estimate the posterior distribution of a dynamic system and finds application in various engineering systems [154]. A dynamic system can be represented by a state space model given by a state transition model (5.1a) and a measurement model (5.1b).

$$x_k = f(x_{k-1}, n_k), \tag{5.1a}$$

$$z_k = h(x_k, v_k), \tag{5.1b}$$

where x, z, n, and v are the system's state, measurement, process noise, and measurement noise respectively, and k is the time step. The state transition model provides information on how the state propagates in the time given an initial state, and this is defined by the function $f(\cdot)$. The measurement model provides information on the relationship between the system's state and its output, defined by the function $h(\cdot)$. Filtering is usually accomplished by sequentially estimating the prior and posterior distributions $p(x_k|z_{1:k-1})$ and $p(x_k|z_{1:k})$ respectively. For linear systems with Gaussian noise and well-defined covariance, this can be represented by the mean and covariance which is the basis of the Kalman filter. The particle filter, on the other hand, represents these distributions by a set of weighted particles $\{x_k^i, \omega_k^i\}_{i=1}^{N_s}$, where N_s is the number of particles and ω_k^i is the normalised weight of the particles. The posterior distribution

can be approximated by

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} \delta_{x_k^i} \omega_k^i, \tag{5.2}$$

where $\delta_{x_k^i}$ is the Dirac delta function at x_k^i and the weight of each particle is normalised such that $\sum_{i=1}^{N_s} \omega_k^i = 1$. The weight update of the ith particle at time k is given by

$$\tilde{\omega}_k^i = \tilde{\omega}_{k-1}^i \frac{p(z_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{k-1}, z_k)}, \tag{5.3}$$

where $q(x_k|x_{k-1}, z_k)$ is the proposal distribution from which the particles were sampled, $\tilde{\omega}_k^i$ is the unnormalised particle weight, and $p(z_k|x_k)$ is the measurement likelihood. For practical applications, the transition distribution $p(x_k|x_{k-1})$ is used as the proposal distribution such that (5.3) reduces to

$$\tilde{\omega}_k^i = \tilde{\omega}_{k-1}^i p(z_k|x_k). \tag{5.4}$$

which can be normalised as

$$\omega_k^i = \frac{\tilde{\omega}_k^i}{\sum_{i=1}^{N_s} \tilde{\omega}_k^i}. \tag{5.5}$$

The particle filter algorithm follows four major steps

1. **Initialisation:** An initial set of particles $\{x_0^i, i = 1, ..., N_s\}$ representing the initial states is drawn from an initial posterior distribution $p(x_0)$ with all the weights ω_0^i initialised to $\frac{1}{N_s}$
2. **Prediction:** Each particle is propagated according to the state transition model in (5.1a). This generates the predicted states of the particles at time step k

3. **Update:** The posterior distribution is updated by evaluating the likelihood $p(z_k|x_k)$ of each particle based on the observed measurements according to (5.1b). This leads to a weight update according to (5.4) and (5.5)

4. **Resampling:** The discrete distribution $p(\omega_k^i)$ according to the normalised weights is resampled to generate a new set of particles x_k^i. Steps 2 to 4 are repeated.

5.3 Guided particle hybrid filter

Given the hand-eye calibration problem, for the purpose of the filter, the state space model is described by

$$q_k = q_{k-1} + n_k, \tag{5.6a}$$

$$E_{\theta_k} = \cos^{-1}\left(|\hat{q}_{A_k} \cdot \hat{q}_{B_k}|\right) \tag{5.6b}$$

$$\hat{q}_{A_k} = q_{A_k} q_k \tag{5.6c}$$

$$\hat{q}_{B_k} = q_k q_{B_k}. \tag{5.6d}$$

where, q_k, q_{A_k} and q_{B_k} are the rotation state, relative camera rotation, and relative robot hand rotation at time step k represented in quaternion respectively, and E_{θ_k} is the pseudo rotation measurement with an expected value of zero. To address some of the issues associated with the standard particle filter algorithm, the following modifications to the standard particle filter algorithms is proposed.

5.3.1 Particle transition

Particle filter works well generally for estimating time-varying parameters. However, for static parameters, this is usually of concern

as the issue of sample impoverishment, where multiple instances of the same particle are generated, after resampling becomes more pronounced [155]. One method to solve the problem of sample impoverishment in this case is to artificially induce particle movement. This can be achieved by drawing new particles from a zero mean Gaussian distribution [156] centered at the current particles' position, such that

$$q_k \approx N(q_{k-1}, \Sigma_q) \tag{5.7}$$

where Σ_q is the covariance matrix of the distribution. While this method improves the sample impoverishment problem, it has the effect of increasing the variance of the posterior distribution [157]. Here, I propose to allow all the particles to move intelligently such that their movement is optimised. The movement of each individual particle at time k is based on the gradient of the particles at time k. The gradient of the particles $\Delta f(q_k)$ can be obtained from the minimisation of the function

$$f(q_k) = \frac{1}{2}||G_k q_k||^2 \tag{5.8a}$$

$$G_k = (q^+_{A_k} - q^-_{B_k}) \tag{5.8b}$$

such that

$$\Delta f(q_k) = G_k^T G_k q_k \tag{5.9a}$$

$$q_k = q_{k-1} - \eta \Delta f(q_{k-1}) \tag{5.9b}$$

where $q^{\pm}$ for a quaternion $q : (q_0, q_n)$ is given as

$$q^{\pm} = \begin{pmatrix} q_0 & -q_n^T \\ q_n & q_0 I \mp [q_n]_\times \end{pmatrix}, \tag{5.10a}$$

$$[q_n]_\times = \begin{pmatrix} 0 & -q_{n_3} & -q_{n_2} \\ q_{n_3} & 0 & -q_{n_1} \\ -q_{n_2} & q_{n_1} & 0 \end{pmatrix} \tag{5.10b}$$

where q_0 and $q_n : (q_{n_1}, q_{n_2}, q_{n_3})$ are the scalar and vector components of the quaternion q, respectively.

However, for this implementation, the modified gradient descent - Nesterov accelerated gradient [158] is used, which improved the performance of the algorithm. This is implemented as

$$v_{q_k} = \gamma_q v_{q_{k-1}} + \eta_q \Delta f(q_{k-1} - \gamma_q v_{q_{k-1}}) \tag{5.11a}$$

$$q_k = q_{k-1} - v_{q_k} \tag{5.11b}$$

where v_{q_k} and η_q are the accumulated gradient and particle step respectively, and γ_q is the momentum term which has a value between 0 and 1. As in typical gradient descent algorithms with momentum, the value chosen for γ_q acts as a damper that affects the oscillations of the particles around the convergence point.

5.3.2 Particle update

Each particle in the particle filter represents a possible orientation of the camera with respect to the robot hand. However, as measurements become available, it becomes possible to guide the particles to the true value. This can be achieved through the evaluation of the likelihood $p(z_k|q_k)$ of each particle. The likelihood of each particle

based on the measurement can be evaluated as

$$p(z_k|q_k) \propto e^{-\lambda E_\theta}, \tag{5.12}$$

where λ is a parameter to be tuned. Note here that the likelihood expression is based on the measurements for the orientation states q_k as defined in (5.6b). This makes sense as the particle filter estimates only the rotation parameter. However, the experiments showed that better estimates can be obtained by coupling the translation measurements in the estimation of the likelihood of the particles. This can be interpreted as the necessary coupling between rotation and translations in pose measurements [68]. Following this, the likelihood of the particles is expressed as

$$p(z_k^i|q_k^i) \propto e^{-\left(\lambda E_{\theta_k}^i + \beta E_{t_k}^i\right)}, \tag{5.13}$$

where β is a tunable parameter and E_t is the translation error resulting from the particles given by

$$E_{t_k}^i = ||q_{A_k}^{-T} q_{A_k}^{+} t_{k-1} - t_{k-1} - q_k^{-T} q_k^{+} t_{B_k} + t_{A_k}||, \tag{5.14}$$

where t_k is the estimated translation parameter at time k. The estimation of t_k will be discussed in Section 5.3.5.

To control the movement of the particles, each individual particle movement η_q^i is adapted based on their respective unnormalised weight $\tilde{\omega}_k^i$. The unnormalised weight gives an indication of the likelihood of a particle considering the entire particle trajectory. The normalised weights, on the other hand, limit the likelihood to the current time step. The adaptation function is of the form in (5.15)

and illustrated in Figure 5.1.

$$\eta_q^i = \Lambda e^{-\zeta \tilde{\omega}_k^i} \tag{5.15}$$

where Λ and ζ are constants that determine the overall shape of the adaptation function. The adaptation function is chosen such that the rate of the particles' movement decreases as they approach the true value. This can be interpreted as the particles decelerating as they converge. This deceleration is modeled using the non-linear function given by (5.15) rather than a linearly decreasing function to achieve more impact on the deceleration as convergence is approached. With the non-linear adaptation, the instability of the particles close to convergence is reduced as they move slower than in the case of linear adaptation. On the other hand, using a linear adaptation with a very steep slope to reduce the movement of the particles faster would leave the particles with not enough energy to achieve convergence.

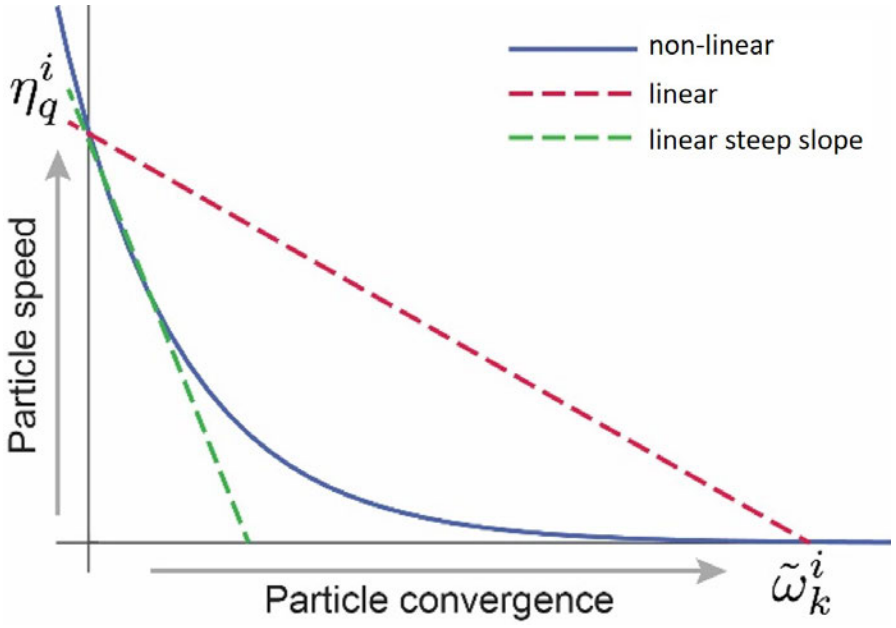


FIGURE 5.1: Non-linear versus linear adaptation functions.

From Figure 5.1, it is easy to see that for each time step, particles with lower likelihood will move faster. This occurs earlier in the

particle filtering operation and in the instances when the camera calibration parameter changes. As the particles converge to the true values, their movement slows down based on the increase in their likelihood. This also has the effect of reducing the variance of the particles.

Selection of the values of the constants Λ and ζ in (5.15) can be done by first taking note of the fact that based on the choice of the likelihood function in (5.13), the value of ω_k^i ranges from 0 to 1. To avoid the particle steps from being too high at low likelihood, which would otherwise cause instabilities, a set of values (η_l, ω_l) is chosen to represent the default step size at low likelihood. Next, another set of values (η_h, ω_h) is chosen to represent the step size at high likelihood. Based on these choices, the values of the constants Λ and ζ can be given as

$$\zeta = \frac{\log\left(\eta_h/\eta_l\right)}{\omega_h - \omega_l} \tag{5.16a}$$

$$\Lambda = \eta_l e^{-\zeta\omega_l} \tag{5.16b}$$

and the particle transition in (5.11a) can be rewritten as

$$v_{q_k}^i = \gamma_q v_{q_{k-1}}^i + \Lambda e^{-\zeta\tilde{\omega}_k^i}\Delta f(q_{k-1}^i - \gamma_q v_{q_{k-1}}^i) \tag{5.17a}$$

$$q_k^i = q_{k-1}^i - v_{q_k}^i \tag{5.17b}$$

The estimated rotation $q_{k_{est}}$ at time k is given by the quaternion that minimises the weighted sum of squared differences of the corresponding rotation matrices of the quaternion particles, measured in the Frobenius norm [159]. This is expressed as

$$q_{k_{est}} = q^T M q \tag{5.18a}$$

$$M = \sum_{i=1}^{n} \omega_k^i q_k^i q_k^{iT} \tag{5.18b}$$

The required average quaternion can be obtained by taking the largest eigenvector of M.

5.3.3 Resampling

In a standard particle filter, after a few iterations, the weight of a particle approaches one, while the weight of others becomes negligible, which is the degeneracy problem[160]. Under this condition, the filter loses its expressiveness as it is not able to properly represent the posterior distribution. This not only results in divergence of the state estimation but computational resources are spent on non-useful particles. The degeneracy problem can be solved by resampling the posterior distribution by selecting particles such that the probability of selecting a particle is proportional to its normalised weight[161]. Several methods of resampling have been proposed [161, 162], however, while these improve the degeneracy problem, they in turn create the sample impoverishment problem. To solve these two problems simultaneously, the particles are allowed to evolve based on inspiration from Genetic Algorithm (GA) [163]. The case for the use of GA stems from the fact that it allows for sustained particle diversity across all generations as new children particles are generated from parent particles such that combined parent traits are obtained. This not only solves the problem of sample impoverishment but at the same time solves the problem of degeneracy.

5.3.4 Genetic algorithm-based resampling

GA is a bio-inspired metaheuristics search algorithm based on survival of the fittest. The main components of GA are chromosome encoding, selection, crossover, and mutation. Chromosome encoding is the representation of a candidate solution in a format that preserves their information while allowing for easy processing. As such the representation of each chromosome is problem specific. The selection process chooses the best chromosomes based on a predefined fitness function. Crossover implementation attempts to combine salient features from two parent chromosomes to generate a fitter offspring, while mutation alters parts of a child chromosome (gene) to improve the diversity of the offspring.

5.3.4.1 Encoding

In the proposed GA-based resampling, each particle represents a chromosome encoded in the axis-angle representation $\{\vec{n}, \theta\}$, where θ is the rotation angle, and $\vec{n} : (\vec{n}_x, \vec{n}_y, \vec{n}_z)$ is the axis of rotation. Here, $\vec{n}$ and θ are the genes.

5.3.4.2 Crossover

From the particle chromosomes, two-parent particles are selected at random for crossover. In a typical GA, several selection schemes like a Roulette wheel, Rank, Tournament, etc are used to select candidate parents for crossover[163]. However, to keep the diversity of the particles as high as possible, the selection of parent particles is based purely on random selection. While this is counter-intuitive as fitter parents are usually used for crossover, the experiments showed that

as the iteration proceeds, the population of fitter parents increases in the particle pool while also benefiting from diversity. Each crossover operation follows the rule

$$g_{ab} = \rho a + (1 - \rho)b \tag{5.19}$$

where g_{ab} is a resulting child gene from parents a and b, and ρ is the crossover probability such that $\rho = U(0, 1)$. The parameter ρ determines the amount of information that is shared between both parents and passed to the offspring. Four children result from every crossover operation given as

$$c_1 = \vec{n}_{12}, \theta_1, \quad c_2 = \vec{n}_{21}, \theta_2, \quad c_3 = \vec{n}_1, \theta_{12}, \quad c_4 = \vec{n}_2, \theta_{21}, \tag{5.20}$$

as shown in Figure 5.2, where the subscript $\{1, 2\}$ represents the parent involved in the crossover.

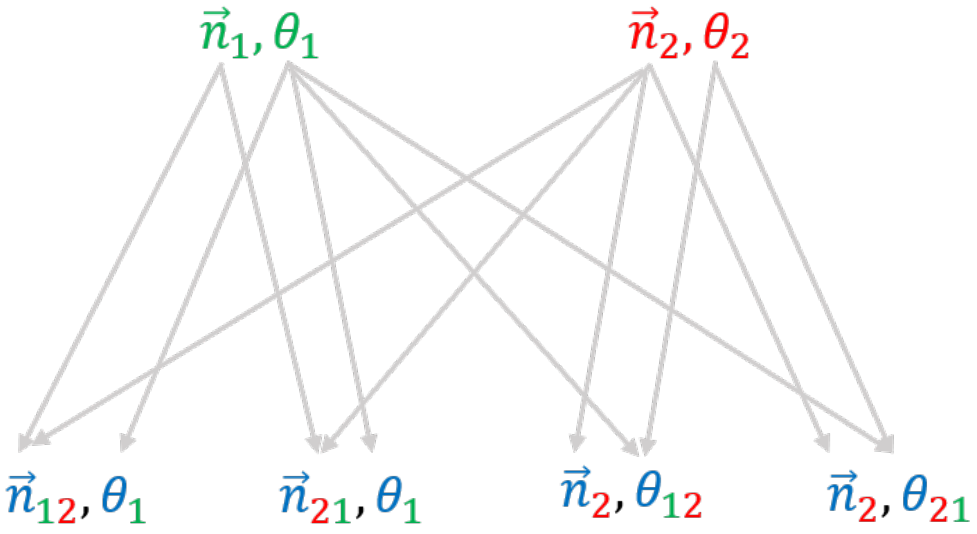


FIGURE 5.2: Proposed GA crossover scheme.

5.3.4.3 Mutation

The mutation is applied to each offspring generated from the crossover operation. This further improves and maintains the diversity of the

offspring. The mutation operation is applied to one gene $\{\vec{n}_x, \vec{n}_y, \vec{n}_z, \theta\}$ selected at random from each of the offspring according to the operation

$$g_m = g_m + \alpha\xi \tag{5.21}$$

where g_m is the gene selected for mutation, α is the mutation probability such that $\alpha = U(0, 1)$ and ξ is a zero mean Gaussian distributed random variable.

5.3.4.4 Population selection

After the crossover and mutation, the best N_p particles from the parent-children pool are selected for the next iteration of the filter, where N_p is the number of particles used in the filter.

5.3.5 Translation estimation

To better represent the density of the posterior distribution in particle filtering a large number of particles is usually required [153]. However, an even greater concern is the dimensionality of the state as the number of particles required to represent a state grows exponentially with the dimension of the state [164]. This is known as the curse of dimensionality. An increased number of particles would require higher computational resources which would undermine the real-time implementation of the algorithm. To reduce the dimensionality of the particle filter, only the rotational parameter is estimated using the particle filter scheme. The rotation estimate from each time step from the particle filter was fed as an input to a simpler estimator to estimate the translation parameter. In this work, two of such estimators - Kalman filter and Gradient decent are considered.

5.3.5.1 Kalman filter

The state prediction $t_{k|k-1}$ corrupted with an assumed Gaussian noise κ at time step k for the translation estimates is given by (5.22a). Likewise, the predicted error covariance $P_{k|k-1}$ is given by (5.22c)

$$t_{k|k-1} = t_{k-1|k-1} + \kappa \tag{5.22a}$$

$$H_k = q_{A_k}^{-T} q_{A_k}^{+} - I \tag{5.22b}$$

$$P_{k|k-1} = P_{k-1|k-1} + Q \tag{5.22c}$$

For the correction step based on the measurement z_k at time k, given by (5.23a), the Kalman gain K_k and the updated covariance $P_{k|k-1}$ is given by (5.23b) and (5.23c) respectively, while the state update is given by (5.23d).

$$z_k = H_k t_{k|k-1} - q_k^{-T} q_k^{+} t_{B_k} + t_{A_k} \tag{5.23a}$$

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R) \tag{5.23b}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{5.23c}$$

$$t_{k|k} = t_{k|k-1} + K_k (z_k - H_k t_{k|k-1}) \tag{5.23d}$$

where R is the covariance of the measurement noise with assumed Gaussian distribution. Notice that z_k in (5.23a) is a pseudo-measurement based on (5.14) with an expected value of zero. However, because the Kalman filter measurement is directly based on the particle filter's output, its measurement error is usually large prior to the convergence of the particle filter. This greatly slows down the convergence of the Kalman filter. To address this slow convergence, taking inspiration from the Iterated Extended Kalman filter [165], improvement is maded to the translation estimate $t_{k|k}$ at time k by back-propagating it n number of times to time step $k-1$ before the next

iteration $k+1$. During the back-propagation at time step k, the updated predicted covariance at time step k is unchanged. As such, the update step becomes

$$t_{k|k}^{j+1} = t_{k|k}^{j} + K_k(z_k - H_k t_{k|k}^{j}). \tag{5.24}$$

where $j = 1, 2...n$.

5.3.5.2 Gradient descent

While the Kalman filter with iterated state update provided better convergence speed and accuracy than the standard Kalman filter, an even better convergence speed was achieved using the much simpler gradient descent estimator. This can be achieved by obtaining the gradient $\Delta f(t_k)$ of the cost function $f(t_k) = \frac{1}{2}E_{t_k}$ given as

$$\Delta f(t_k) = J_k^T(J_k t_k - m_k) \tag{5.25a}$$

$$J_k = q_{A_k}^{-T} q_{A_k}^{+} - I \tag{5.25b}$$

$$m_k = q_k^{-T} q_k^{+} t_{B_k} - t_{A_k} \tag{5.25c}$$

Just like in Section 5.3.1, the Nesterov accelerated gradient [158] is also used, and is implemented as

$$v_{t_k} = \gamma_t v_{t_{k-1}} + \eta_t \Delta f(t_{k-1} - \gamma_t v_{k-1}) \tag{5.26a}$$

$$t_k = t_{k-1} - v_{t_k} \tag{5.26b}$$

where v_{t_k} and η_t are the accumulated gradient and translation state step respectively, and γ_t is the momentum term, which is similar to γ_q defined in (5.11a).

5.4 Experiment and discussion

In this section, the setup used to validate the developed active calibration algorithm and show its comparative advantage over offline calibration methods is described.

5.4.1 Setup

In the experiment, a UR5e robot arm is used to which a camera assembly is rigidly attached to the last link of the robot arm. The setup is shown in Figure 5.3. The camera assembly shown in Figure 5.3A is made up of a Microsoft Azure Kinect camera fastened to an articulated ball mount (Thorlabs TRB/M). The articulated ball mount can be moved in any direction and locked in place. This enabled the pose of the camera relative to the robot hand (Figure 5.3B) to be changed during the robot's operation to evaluate the performance of the algorithm.

During the operation of the robot, the robot moves randomly over a work surface as shown in Figure 5.3C. The work surface contains multiple random work objects and several unique 50 mm 5×5 grid Aruco markers. The small size of the markers makes them suitable for this task as they do not interfere with the work object and the use of multiple markers overcomes the problem of occlusion. The markers allow estimation of the relative camera motion between consecutive image frames, and their unique signature enables the detection of a particular marker across image frames. The algorithm was implemented in Python and ran on a Windows PC with Intel i5-2.4GHz CPU and 8GB of RAM and achieved an average computational time cost of 0.37 seconds. However, there is room for improvement in

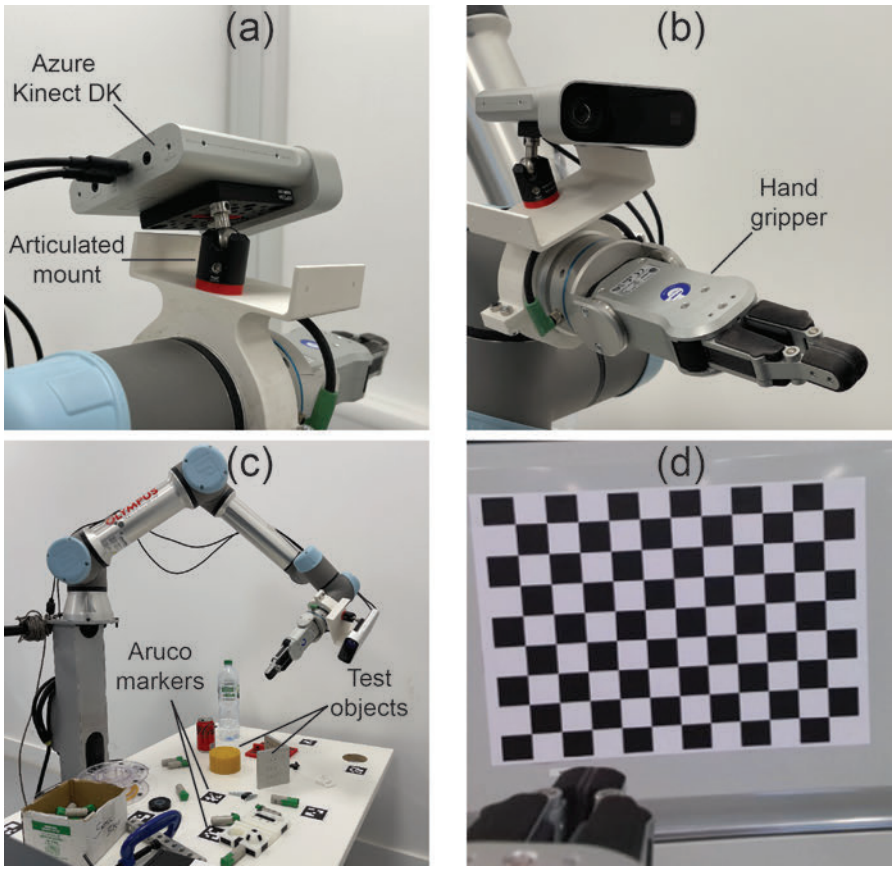


FIGURE 5.3: Experimental setup. (A) Camera assembly (B) Camera-robot hand setup (C) Robot operation setup (D) Calibration grid for offline hand-eye calibration.

the implementation of the algorithm through optimisation and possible implementation is C/C++.As the robot moves, the pose of the robot hand and the corresponding camera pose is acquired at certain intervals, and the relative poses between consecutive data acquired are estimated. In the experiment, an acquisition interval of 100 milliseconds (i.e., 10 data acquisitions per second) is used, as it was important to allow the robot to make sufficient motions before making a new acquisition. It was noted that higher acquisition rates poorly affected the hand-eye calibration accuracy. At different points in the motion of the robot, the pose of the camera is changed by moving the articulated ball mount to a different position. This effectively changes the pose of the camera relative to the robot hand and in effect, the hand-eye calibration parameter as would occur if

the robot were under the influence of an external force in an uncertain environment. In the estimation of the relative camera motion between consecutive frames, only markers that are unoccluded in both frames are used as shown in Fig. 5.4 where two markers with their frames plotted have been detected.

FIGURE 5.4: Consecutive image frames. Only markers that are unoccluded in both frames are detected (markers in white circles).

For the evaluation of the accuracy of the offline hand-eye calibration methods, an 11×8 checkerboard calibration pattern shown in Fig. 5.3D was used.

5.4.2 Evaluation metrics

Due to the absence of ground truth data for the hand-eye calibration parameters, the relative rotation e_θ and translation e_t errors derived

from (2.8a) and (2.8) respectively were used for the comparative assessment of the accuracy of the different algorithms. These are given as

$$e_\theta = \frac{1}{N}\sum_{i=1}^{N} \cos^{-1}\left(\frac{\mathrm{Tr}\left[(R_{A_i}R_X)^{-1}R_X R_{B_i}\right]-1}{2}\right), \tag{5.27a}$$

$$e_t = \frac{1}{N}\sum_{i=1}^{N} \left|R_{A_i}t_X - t_X - R_X t_{B_i} + t_{A_i}\right|, \tag{5.27b}$$

where the notation Tr denotes the trace of the matrices and N is the number of relative pose data used for the experiment.

5.4.3 Results and discussion

In this section, the performance of the proposed online active hand-eye calibration algorithms, namely

- PF-Kalman Filter (particle filter with Kalman filter)
- PF-Iterated Kalman Filter (particle filter with iterated Kalman Filter)
- PF-Gradient Descent (particle filter with gradient descent)

is compared to the offline hand-eye calibration algorithms. The offline algorithms are the quaternion [67], Lie-group [52], axis-angle [51], dual quaternion [69], simultaneous quaternion [53] and Kronecker product [166]. The result is shown in Figure 5.5. From Figure 5.5 the pose of the camera was altered at time steps 323, 512, and 891. At these points, the rotation and translation errors in all the algorithms jump up. However, from each of these points, the algorithm is able to recover from the introduced error and actively

obtains the new calibration parameters as evidenced by the reduction in the rotation and translation errors. This self-recovery ability of the algorithm makes it suitable for uncertain environments where the camera pose may be affected, for example, due to vibrations, slips due to component wear from prolonged usage or shock from abrupt changes in robot motions, or in critical applications where taking the robot out of service for recalibration may not be ideal. In these instances, offline calibration methods would not be ideal as they cannot deal with the introduced error, which is consistent with the previous study [167]. Furthermore, considering the period before the changes in the camera pose were introduced, i.e., from time steps 0 to 323, the results from Figure 5.5 also show that the algorithm performed as well or even better than the offline methods. While the rotation error oscillates around the best offline method in this experiment, the translation error was consistently below that of all the offline methods as shown in Figure 5.6. This shows that even for comparison with offline methods, the algorithm still stands out.

While the three algorithms can adapt to changes in the hand-eye calibration parameters, Figure 5.5 shows marked differences in their performance, particularly in the translation estimates. The most obvious difference is the rate of convergence in the translation estimates. The convergence rate from the standard Kalman filter implementation was low. So much so that it did not fully converge before the camera pose was changed again. While the algorithm with the iterated Kalman filter implementation produces a much-improved convergence rate compared to the standard Kalman filter, the gradient descent approach provided the highest convergence rate. However, as observed from Figure 5.7, the iterated Kalman filter methods performed better than the gradient descent approach based on the

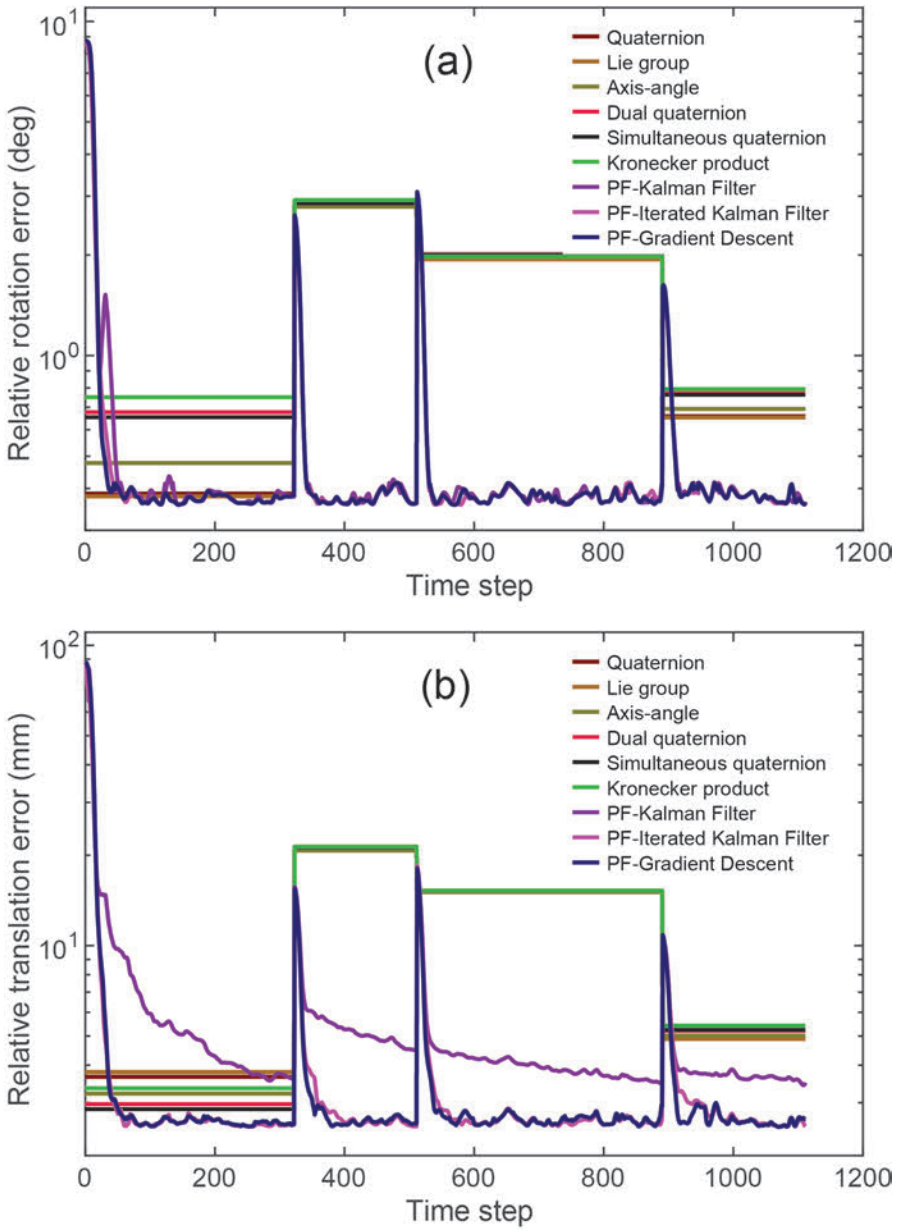


FIGURE 5.5: Comparative accuracy of the proposed active hand-eye calibration methods and the offline calibration methods. (A) Relative rotation error. (B) Relative translation error.

translation error.

5.5 Summary

This Chapter presented an algorithm for active hand-eye calibration based on a hybrid filter. The guided hybrid filter was built on the particle filter with optimised particle transition and GA resampling

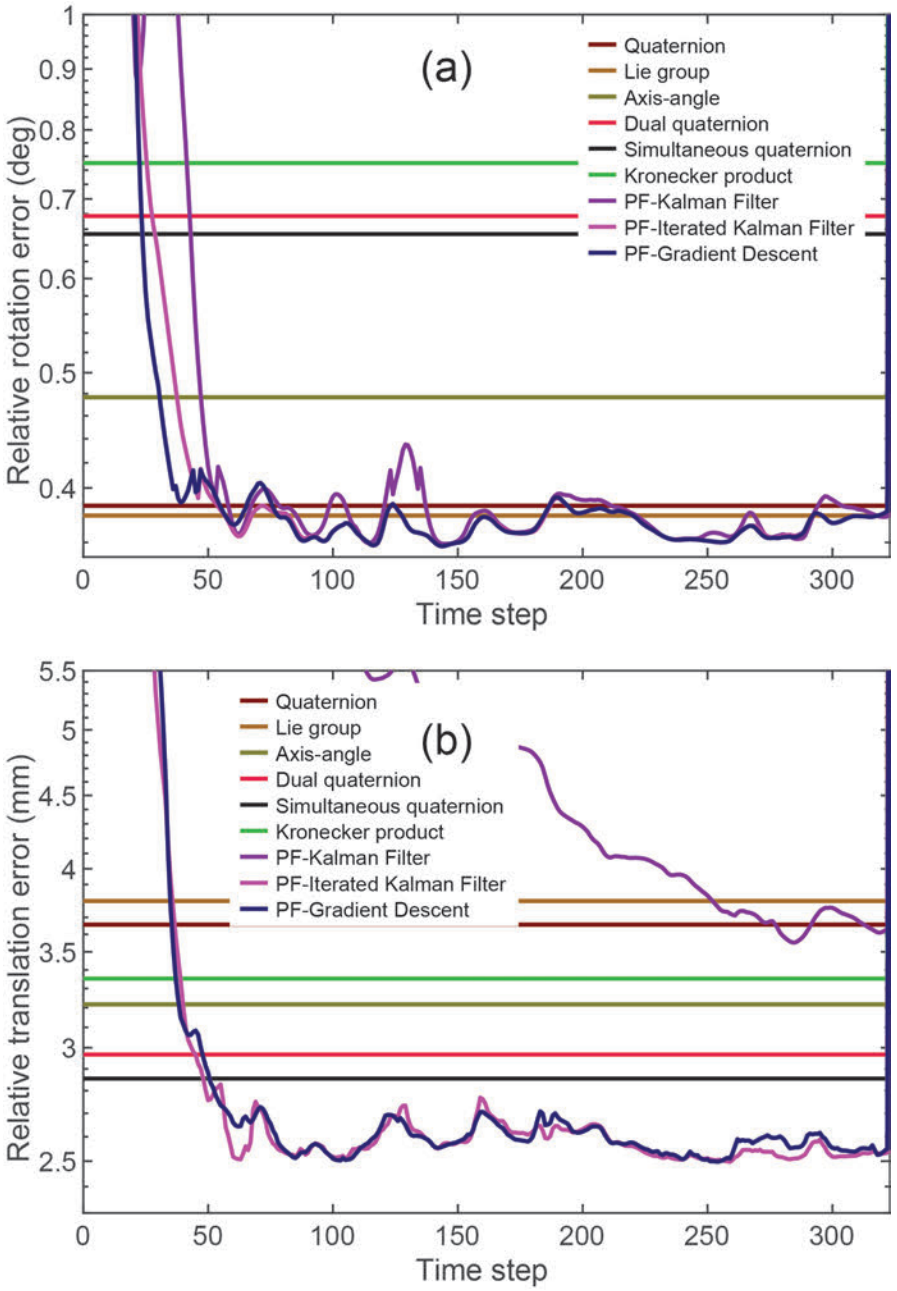


FIGURE 5.6: Comparative accuracy of different algorithms prior to changes in calibration parameters, (i.e., before time steps 323). (A) Relative rotation error. (B) Relative translation error.

and coupled with an iterated Kalman filter state estimator to reduce the workload of the particle filter. The particle filter was used to estimate the rotation parameter, while the iterated Kalman filter was used to estimate the translation parameter. The use of gradient descent for the estimation of the translation parameter was also evaluated. From the experimental results, it is verified that the

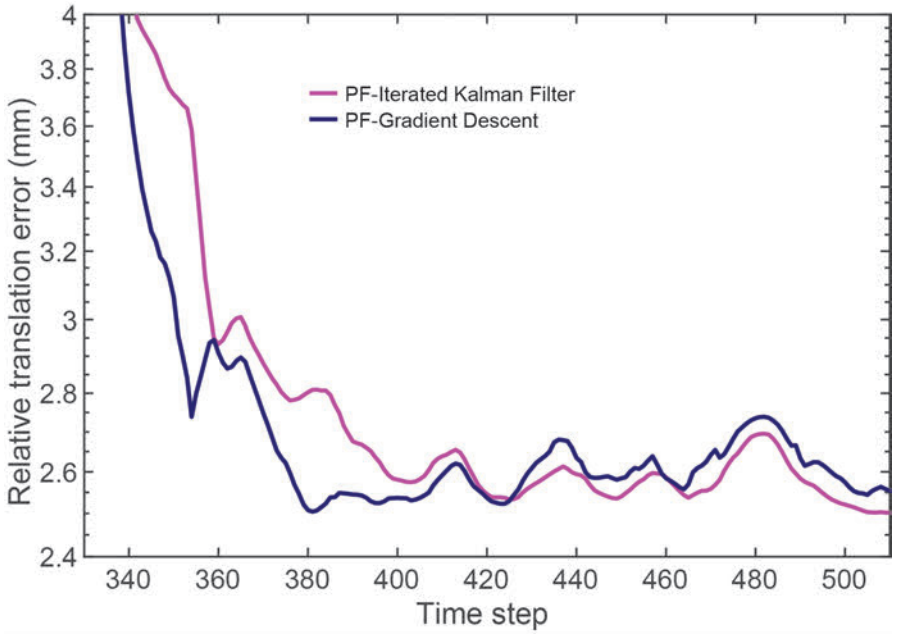


FIGURE 5.7: Comparison of translation estimate based on PF-Iterated Kalman Filter and PF-Gradient Descent.

proposed algorithm not only provides high accuracy when compared with other offline calibration methods but is able to recover from errors introduced in the robot's operation due to changes in the calibration parameters in which other offline methods would fail. From the evaluation of the algorithms used to estimate the translation parameter, the gradient descent estimator showed a higher convergence rate than the iterated Kalman filter. However, the iterated Kalman filter provided better accuracy than the gradient descent estimator. Furthermore, In all evaluations, the standard Kalman filter estimator provided the worst result in terms of accuracy and convergence rate.

Chapter 6

Concluding Remarks and future works

6.1 Conclusions

The aim of this research is to develop a self-recoverable hand-eye calibration algorithm for VGRs to enable sustained accuracy during the service life of the robot. This is important to enable a VGR to operate for a prolonged period of time without the need for recalibration. In view of this, a new algorithm for online hand-eye calibration has been developed to meet this requirement via the accomplishment of several objectives.

The first objective involved a comprehensive review of hand-eye calibration techniques for VGR. A broad overview of different hand-eye calibration techniques was looked at, as well as their comparative strengths and weaknesses. Common challenges expected in the calibration of vision-guided robots as well as practical considerations when using a calibration target were also examined.

The second objective was to assess the accuracy of current hand-eye calibration techniques through simulation and experimental studies.

The execution of this objective provided an experimental validation of the factors that affect the accuracy of hand-eye calibration for vision-guided robots. In particular, the effect of rotation and translation noise isolation as well as motion range was studied using six different algorithms. The results from this study provide insights that can inform the choice of algorithm for hand-eye calibration based on the application constraints.

The impact of changes in the calibration parameter on the accuracy of hand-eye calibration in an uncertain environment was also examined. It was shown experimentally that the accuracy deteriorates exponentially with increase in the deviation of the calibration parameter from its true value. This has a direct practical impact in an industry where operational downtime is costly. In this study, active hand-eye calibration was proposed as a solution to compensate for changes in the calibration parameter online. The requirements and challenges of active hand-eye calibration were presented and these formed the basis of the next two objectives which are the development and implementation of a self-recoverable hand-eye calibration algorithm.

The algorithm developed employed a guided particle hybrid filter with genetic algorithm-based resampling. The performance of this algorithm was validated experimentally and shown to outperform other offline-specific algorithms in terms of accuracy. The algorithm was also shown to be able to recover from errors due to changes in the calibration parameters, an attribute required for long-term operational guarantee of the accuracy of vision-guided robots.

The implementation of this algorithm was done in Python and executed on a Windows PC with Intel i5-2.4GHz CPU and 8GB of

RAM achieving a computation time of 0.37 seconds.

6.2 Future work

In this work, the placed fiducial markers in the workspace for the camera pose estimation were relied on. This technique works well as the markers do not interfere with the workspace due to their size and form, and the pose estimation procedure does not suffer from occlusion. An interesting direction for future work will involve estimating the camera pose using techniques from deep learning to completely eliminate the need for using markers in the workspace. In this case, the camera pose can be estimated from motion cues obtained from successive image frames from the camera using convolutional neural network (CNN) models. Currently, a number of deep learning models exist for estimating the relative camera motions. However, these models are developed for photogrammetric and 3D reconstruction applications and do not meet the accuracy levels that would be sufficient for accurate hand-eye calibration. Hence, the development of a deep-learning model for accurate camera localization would be an interesting research proposition.

Secondly, it would be interesting to explore the possibility of integrating knowledge of the scene motion from the camera during the robot operation into the particle filtering algorithm. One way may include the formulation of the particle likelihood with the epipolar constraint from subsequent images as a component.

Furthermore, the implementation of the guided hybrid particle filtering algorithm employs a number of tunable parameters. This may be challenging to get an optimal result. An interesting future direction

may involve the development of optimization algorithms to obtain the best combination of parameters for the application. This might involve techniques such as grid search, random search, Bayesian optimization, gradient-based optimization, etc.

Finally, the experimental validation of the algorithm was based on a Python implementation. While the results from the implementation were good, this implementation leaves a lot of room for optimisation-based improvements. An implementation in C/C++ would also be recommended to realise a greater performance improvement in terms of computational time.

References

[1] M. Pena and R. Osorio. Robot vision methodology for assembly manufacturing tasks. Electronics, Robotics and Automotive Mechanics Conference, pages 289–294, 2007.

[2] R. Lan Y. Zou. An end-to-end calibration method for welding robot laser vision systems with deep reinforcement learning. IEEE Transaction on Instrumentation and Measurement, 69 (7):4270–4280, 2020.

[3] Shunsuke Kudoh, Koichi Ogawara, Miti Ruchanurucks, and Katsushi Ikeuchi. Painting robot with multi-fingered hands and stereo vision. Robotics and Autonomous Systems, 57(3): 279–288, 2009.

[4] V. Pawar S. Hailes K. Pachtrachai, M. Allan and D. Stoyanov. Hand-eye calibration for robotic assisted minimally invasive surgery without a calibration object. IEEE International Conference on Intelligent Robots and Systems, pages 2485–2491, 2016.

[5] Mohammed Faisal, Mansour Alsulaiman, Mohammed Arafah, and Mohamed Amine Mekhtiche. IHDS: Intelligent harvesting decision system for date fruit based on maturity stage using deep learning and computer vision. IEEE Access, 8:167985–167997, 2020.

[6] The Broadcast Bridge. The rise of the robotic camera, 2023. URL https://www.thebroadcastbridge.com/content/entry/12076/the-rise-of-the-robotic-camera.

[7] RoboDK. Industrial robot applications, 2022. URL https://robodk.com/blog/industrial-robot-applications/.

[8] The Straits Times. Intelligent service robot, 2023. URL https://www.straitstimes.com/singapore/intelligent-service-robots-rolled-out-to-transform-\service-industry-amid-pandemic.

[9] Avidbots. Change the way you think about floor cleaning, 2021. URL https://avidbots.com/meet-neo/.

[10] J.-S. Shin J.-W. Lee, G.-T. Park and J.-W. Woo. Industrial robot calibration method using denavit—hatenberg parameters. International Conference on Control, Automation and Systems, pages 1834–1837, 2017.

[11] I.-F. Lu J.-C. Hsiao, K. Shivam and T.-Y. Kam. Positioning accuracy improvement of industrial robots considering configuration and payload effects via a hybrid calibration approach. IEEE Access, 8:228992–229005, 2020.

[12] Keyence. 2d vision-guided robotics, 2021. URL https://www.keyence.com/products/vision/vision-sys/2d-vgr/.

[13] Luis Alejandro Camuñas-Mesa, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. Event-driven sensing and processing for high-speed robotic vision. In 2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings, pages 516–519. IEEE, 2014.

[14] Martin Saska, Tomáš Krajník, Jan Faigl, Vojtěch Vonásek, and Libor Přeučil. Low cost mav platform ar-drone in experimental verifications of methods for vision based autonomous navigation. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4808–4809. IEEE, 2012.

[15] Feng Liang and Chun Zhang. Hardware oriented vision system of logistics robotics. In 2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), pages 6–9. IEEE, 2018.

[16] X. X. Lu. A review of solutions for perspective-n-point problem in camera pose estimation. Journal of Physics: Conference Series, 1087(5), 2018.

[17] A. Bobkov M. Alkhatib and N. Zadoroznaya. Camera pose estimation based on structure from motion. Procedia Computer Science, 0509.

[18] B. Vintimilla J. Charco and A. Sappa. Deep learning based camera pose estimation in multi-view environment. Proceedings of International Conference on Signal-Image Technology and Internet-Based Systems, pages 224–228. 10.1109/SITIS.2018.00041., 0041.

[19] A. Andreoni G. Anzolin and A. Zanfei. Robot adoption and fdi driven transformation in the automotive industry. International Journal of Automotive Technology and Management, 20(2):215–237, 2020.

[20] A. Eilertsen J. R. Mathiassen O. B. Åsebø T. Gjerstad J. Buljo E. Misimi, E. R. Øye and Ø. Skotheim. Gribbot–robotic 3d vision-guided harvesting of chicken fillets. Computers and Electronics in Agriculture, 121:84–100, 2016.

[21] E. Bjørlykhaug and O. Egeland. Vision system for quality assessment of robotic cleaning of fish processing plants using cnn. IEEE Access, 7:71675–71685, 2019.

[22] M. Arafah M. Faisal, M. Alsulaiman and M. A. Mekhtiche. Ihds: Intelligent harvesting decision system for date fruit based on maturity stage using deep learning and computer vision. IEEE Access, 8:167985–167997, 2020.

[23] D. Li M. Imran C. Zhang C. Liu J. Wan, S. Tang and Z. Pang. Reconfigurable smart factory for drug packing in healthcare industry 4.0. IEEE Transactions on Industrial Informatics, 15 (1):507–516, 2018.

[24] A. Eizad M. Park, T.-A. Le and J. Yoon. A novel shared guidance scheme for intelligent haptic interaction based swarm control of magnetic nanoparticles in blood vessels. IEEE Access, 8:106714–106725, 2020.

[25] P. Webb N. Jayaweera and C. Johnson. Measurement assisted robotic assembly of fabricated aero-engine components. Assembly Automation, 2010.

[26] S. Bai Z. Li, S. Suntharasantic and P. Chirarattananon. Aeromechanic models for flapping-wing robots with passive hinges in the presence of frontal winds. IEEE Access, 6:53890–53906, 2018.

[27] Y. Li L. Wu, H. Li and C. Li. Position tracking control of tailsitter vtol uav with bounded thrust-vectoring propulsion system. IEEE Access, 7:137054–137064, 2019.

[28] IFR. International federation of robotics, 2021. URL https://ifr.org/ifr-press-releases/news/wr-report-all-time-high-with-half-a-million-robots-installed.

[29] M. Saedan and M. H. Ang Jr. 3d vision-based control on an industrial robot. Proceedings of the IASTED International Conference on Robotics and Applications, pages 152–157, 2001.

[30] N. Herakovic. Robot vision in industrial assembly and quality control processes. INTECH Open Access Publisher, 2010.

[31] M. Zervakis L. Petit E. N. Malamas, E. G. Petrakis and J.-D. Legat. A survey on industrial vision systems, applications and tools. Image and Vision Computing, 21(2):171–188, 2003.

[32] K. Xin S. Gobee, V. Durairajah and L. L. Jie. Robotic vision based pcb inspection with iot interface. International Conference on Control, Robotics and Cybernetics, pages 27–31, 2018.

[33] X. Wang X. Fan and Y. Xiao. A combined 2d-3d vision system for automatic robot picking. Proceedings of the International Conference on Advanced Mechatronic Systems, pages 513–516, 2014.

[34] K. Zhao D. Tang H. Zhou G. Li X. Xiang Y. Zhou, Q. Fang and T. Hu. Robust task-oriented markerless extrinsic calibration for robotic pick-and-place scenarios. IEEE Access, 7:127932–127942, 2019.

[35] L. F. Moreira A. O. Fernandes and J. M. Mata. Machine vision applications and development aspects. IEEE International Conference on Control and Automation, pages 1274–1278, 2011.

[36] E. Martinez-Martin and A. P. D. Pobil. Robot vision for manipulation: A trip to real-world applications. IEEE Access, 9: 3471–3481, 2021.

[37] S. Kumar R. Kumar, S. Lal and P. Chand. Object detection and recognition for a pick and place robot. Asia-Pacific World Congress on Computer Science and Engineering, pages 1–7, 2014.

[38] Y. Zheng K. Chen F. Wang L. Mu, P. Yao and N. Qi. Research on slam algorithm of mobile robot based on the fusion of 2d lidar and depth camera. IEEE Access, 8:157628–157642, 2020.

[39] Z. Li B. Guo, H. Dai and W. Huang. Efficient planar surface-based 3d mapping method for mobile robots using stereo vision. IEEE Access, 7:73593–73601, 2019.

[40] A. Yasin K. Javed A. Shahzad, X. Gao and S. M. Anwar. A vision-based path planning and object tracking framework for 6-dof robotic manipulator. IEEE Access, 8:203158–203167, 2020.

[41] S. Yang S. Yang and X. Yi. An efficient spatial representation for path planning of ground robots in 3d environments. IEEE Access, 6:41539–41550, 2018.

[42] V. Singule A. Pochyly, T. Kubela and P. Cihak. 3d vision systems for industrial bin-picking applications. Proceedings of International Conference, Mechatronika, pages 1–6, 2012.

[43] Y.-J. Jhao L.-Y. Chen J.-D. Lee, Y.-H. Wu and H.-I. Chen. Development of mobile robot with vision inspection system and three-axis robot. International Conference on Control and Robotics Engineering, pages 6–10, 2018.

[44] M.-H. Jeong Y. Y. Kim and D. J. Kang. Mobile robot calibration. Annual Conference of the IEEE Industrial Electronics Society, pages 5504–5506 2013., 2013.

[45] F. Li W. Qi and L. Zhenzhong. Review on camera calibration. Chinese Control and Decision Conference, pages 3354–3358, 2010.

[46] X. Xu Z. Jiang X. Gong, Y. Lv and Z. Sun. High-precision calibration of omnidirectional camera using an iterative method. IEEE Access, 7:152179–152186, 2019.

[47] Y. C. Shiu and S. Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax= xb. IEEE Transactions on Robotics and Automation, 5(1):16–29, 1989.

[48] X. Gu N. Tan and H. Ren. Simultaneous robot-world, sensor-tip, and kinematics calibration of an underactuated robotic hand with soft fingers. IEEE Access, 6:22705–22715, 2018.

[49] O. Faugeras. Three-dimensional computer vision: a geometric viewpoint. MIT press, 1993.

[50] P. Pu A. Izaguirre and J. Summers. A new development in camera calibration calibrating a pair of mobile cameras. The International Journal of Robotics Research, 6(3):104–116, 1987.

[51] R. K. Lenz *et al.* R. Y. Tsai. A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. IEEE Transactions on robotics and automation, 5(3):345–358, 1989.

[52] F. C. Park and B. J. Martin. Robot sensor calibration: solving ax = xb on the euclidean group. IEEE Transactions on Robotics and Automation, 10(5):717–721, 1994.

[53] J. C. Chou and M. Kamel. Finding the position and orientation of a sensor on a robot manipulator using quaternions. International Journal Robotics Research, pages 240–254, 1991.

[54] A. Sugimoto J. Heller, M. Havlena and T. Pajdla. Structure-from-motion based hand-eye calibration using l∞ minimization. IEEE Conference on Computer Vision and Pattern Recognition, pages 3497–3503, 2011.

[55] R. Horaud N. Andreff and B. Espiau. Robot hand-eye calibration using structure-from-motion. International Journal of Robotics Research, pages 228–248, 2001.

[56] G. Wei and K. Arbterand G. Hirzinger. Active self-calibration of robotic eyes and hand-eye relationships with model identification. IEEE Transactions on Robotics and Automation, 14 (1):158–166, 1998.

[57] S. Lee and S. Ro. A self-calibration model for hand-eye systems with motion estimation. Mathematical and computer modelling, 24(5-6 pp. 49-77):49–77, 1996.

[58] S. Teller J. Brookshire. Extrinsic calibration from per-sensor egomotion. Robotics: Science and Systems VIII, pages 504–512, 2013.

[59] M. Ackerman and G. Chirikjian. A probabilistic solution to the ax=xb problem: Sensor calibration without correspondence. International Conference on Geometric Science of Information, pages 693–701, 2013.

[60] L. Haiyuan Li M. Qianli and G. Chirikjian. New probabilistic approaches to the ax = xb hand-eye calibration without correspondence. 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 4365–4371, -4371.

[61] J. Brookshire and S. Teller. Articulated pose estimation using tangent space approximations. The International Journal of Robotics Research, 35(1-3):5–29, 2016.

[62] J. Brookshire and S. Teller. Automatic calibration of multiple coplanar sensors. Robotics: Science and Systems VIIh, 33, 2012.

[63] Z. Chen B. Zhou and Q. Liu. An efficient solution to the perspective-n-point problem for camera with unknown focal length. IEEE Access, 8:162838–162846, 2020.

[64] F. R. Gantmacher. Matrix theory. Chelsea, New York, 21, 1959.

[65] R.-H. Liang and J.-F. Mao. Hand-eye calibration with a new linear decomposition algorithm. Journal of Zhejiang University-Science A, 9(10):1363–1368, 2008.

[66] H. Nguyen and Q. Pham. On the covariance of $\boldsymbol{x}$ in $\boldsymbol{ax} = \boldsymbol{xb}$. IEEE Transactions on Robotics, 34(6):1651–1658, 2018.

[67] Y.-C. Lu and J. C. Chou. Eight-space quaternion approach for robotic hand-eye calibration. IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, pages 3316–3321, 1995.

[68] H. H. Chen. A screw motion approach to uniqueness analysis of head-eye geometry. Proceedings of IEEE Computer Society

Conference on Computer Vision and Pattern Recognition, pages 145–146, 1991.

[69] K. Daniilidis and E. Bayro-Corrochano. The dual quaternion approach to hand-eye calibration. Proceedings of International Conference on Pattern Recognition, pages 318–322, 1996.

[70] R. Horaud N. Andreff and B. Espiau. On-line hand-eye calibration. International Conference on 3-D Digital Imaging and Modeling, pages 430–436, 1999.

[71] D. Condurache and A. Burlacu. Orthogonal dual tensor method for solving the ax= xb sensor calibration problem. Mechanism and Machine Theory, 104:382–404, 2016.

[72] D. Condurache and I.-A. Ciureanu. A novel solution for ax=yb sensor calibration problem using dual lie algebra. International Conference on Control, Decision and Information Technologies, pages 302–307, 2019.

[73] J. Kim S. Gwak and F. C. Park. Numerical optimization on the euclidean group with applications to camera calibration. IEEE Transactions on Robotics and Automation, 19(1):65–74, 2003.

[74] D. Henrion J. Heller and T. Pajdla. Hand-eye and robot-world calibration by global polynomial optimization. IEEE International Conference on Robotics and Automation, pages 3157–3164, 2014.

[75] Z. Zhao. Simultaneous robot-world and hand-eye calibration by the alternative linear programming. Pattern Recognition Letters, 127:174–180, 2019.

[76] L. Zhang Z. Zhang and G.-Z. Yang. A computationally efficient method for hand–eye calibration. International Journal of Computer Assisted Radiology and Surgery, 69(6):1775–1787, 2020.

[77] Jin Wu, Yuxiang Sun, Miaomiao Wang, and Ming Liu. Hand-Eye calibration: 4-D procrustes analysis approach. IEEE Transaction on Instrumentation and Measurements, 69(6): 2966–2981, June 2020.

[78] K. Strobl and G. Hirzinger. Optimal hand-eye calibration. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4647–4653, 4653,.

[79] B. Mooring Z. Roth and B. Ravani. An overview of robot calibration. IEEE Journal on Robotics and Automation, 3(5): 377–385, 1987.

[80] X. Shi and Y. Liu. Reprojection error based annealed particle filter for human upper body pose reconstruction. Proceedings of IEEE International Conference on Service Operations and Logistics, and Informatics, pages 277–281, 2014.

[81] E. R. Morales I. Ali, O. J. Suominen and A. Gotchev. Multi-view camera pose estimation for robotic arm manipulation. IEEE Access, 8:174305–174316, 2020.

[82] A. Moro A. Yamashita W. Yin, S. Pathak and H. Asama. Accurate all-round 3d measurement using trinocular spherical stereo via weighted reprojection error minimization. IEEE International Symposium on Multimedia, pages 86–867, 2019.

[83] C. Wang R. Zhu, H. Kiani Galoogahi and S. Lucey. Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. Proceedings of the IEEE International Conference on Computer Vision, pages 57–65, 2017.

[84] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. International Conference on Computer Vision, pages 383–390, 2011.

[85] G.-J. Zhang W. Wang and Z.-Z. Wei. A framework for 2d model-based pose estimation. International Conference on Computer Science and Applications, pages 48–51, 2015.

[86] J. Dong T. Wang L. Qi S. Zhang, H. Yu and H. Liu. Combining kinectic and pnp for camera pose estimation. International Conference on Human System Interaction, pages 357–361, 2015.

[87] Thermobot. Thermobot: Autonomous robotic system for thermo-graphic detection of cracks, 2021. URL https://thermobot.eu.

[88] K. Koide and E. Menegatti. General hand–eye calibration based on reprojection error minimization. IEEE Robotics and Automation Letters, 4(2):1021–1028, 2019.

[89] V. Eskov V. Eskov, V. Pyatin and L. Ilyashenko. The heuristic work of the brain and artificial neural networks. Biophysics, 64 (2):293–299, 2019.

[90] A. E. Omolara K. V. Dada A. M. Umar O. U. Linus H. Arshad A. A. Kazaure U. Gana O. I. Abiodun, A. Jantan and M. U.

Kiru. Comprehensive review of artificial neural network applications to pattern recognition. IEEE Access, 7:158820–158846, 2019.

[91] M. Mainampati and B. Chandrasekaran. Evolution of machine learning algorithms on autonomous robots. 10th Annual Computing and Communication Workshop and Conference, ution of machine learning algorithms on autonomous robots: 737–741, 2020.

[92] J. Ma Y. Guo, X. Ke and J. Zhang. A pipeline neural network for low-light image enhancement. IEEE Access, 7:13737–13744, 2019.

[93] H. Ahmed A. Avelar, I. Salgado and I. Chairez. Differential neural networks observer for second order systems with sampled and quantized output. IFAC-PapersOnLine, 51(13):490–495, 2018.

[94] O. Camacho I. Salgado, H. Ahmed and I. Chairez. Adaptive sliding-mode observer for second order discrete-time mimo nonlinear systems based on recurrent neural-network. International Journal of Machine Learning and Cybernetics, 10(10):2851–2866, 2019.

[95] M. Mera A. Guarneros, I. Salgado and H. Ahmed. Differential neural network identifier with composite learning laws for uncertain nonlinear systems. IFAC-PapersOnLine, 53(2):7897–7902, 2020.

[96] J. Hua and L. Zeng. Hand–eye calibration algorithm based on an optimized neural network. Actuators, 10(4), 2021.

[97] M. Mishra and M. Srivastava. A view of artificial neural network. International Conference on Advances in Engineering and Technology Research, pages 1–3, 2014.

[98] D. K. Chaturvedi. Soft computing: studies in computational intelligence. Springer, Berlin, Heidelberg, 2008.

[99] I. Bilbao and J. Bilbao. Overfitting problem and the overtraining in the era of data: Particularly for artificial neural networks. International Conference on Intelligent Computing and Information Systems, pages 173–177, 2017.

[100] A. P. Piotrowski and J. J. Napiorkowski. A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. Journal of Hydrology, 476: 97–111, 2013.

[101] X. Armangué J. Salvi and J. Batlle. A comparative review of camera calibrating methods with accuracy evaluation. Pattern Recognition, 35(7):1617–1635, 2002.

[102] J. Mallon and P. F. Whelan. Which pattern? biasing aspects of planar calibration patterns and detection methods. Pattern Recognition Letters, 28(8):921–930, 2007.

[103] National Instrument. N. vision. perspective and nonlinear distortion calibration., 2021. URL https://documentation.help/NI-Vision-LabView-Basics/Perspective_and_Nonlinear_Distortion_Calibration.html.

[104] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. Proceedings of IEEE International Conference on Computer Vision, pages 666–673, 1999.

[105] J. Heikkila and O. Silvén. A four-step camera calibration procedure with implicit image correction. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 1106–1112, 1997.

[106] P. F. Sturm and S. J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 432–437, 1999.

[107] L. Gao, Y. Fang, Z. Lin, and K. Chen. A versatile camera calibration technique for high accuracy 3d machine vision metrology. Mechanical Science and Technology, 5, 1998.

[108] Z. Wang Z. Wang and Y. Wu. Recognition of corners of planar checkboard calibration pattern image. Chinese Control and Decision Conference, pages 3224–3228, 2010.

[109] OpenCV. L. v. technology. camera calibration using a circle grid., 2021. URL https://longervision.github.io/2017/03/18/ComputerVision/OpenCV/opencv-internal-calibration-circle-grid/.

[110] R. Kasturi R. Jain and B. G. Schunck. Machine vision. McGraw-hill New York, 5, 1995.

[111] D. DeTone D. Hu and T. Malisiewicz. Deep charuco: Dark charuco marker pose estimation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8436–8444, 2019.

[112] T. Clarke and X. Wang. The control of a robot end-effector using photogrammetry. International Archives of

Photogrammetry and Remote Sensing, 33(B5/1):137–142, 2000.

[113] M. Goitein K. McCarthy I. Rabinowitz, J. Broomberg and J. Leong. Accuracy of radiation field alignment in clinical practice. International Journal of Radiation, Oncology, Biology, Physics, 11(10):1857–1867, 1985.

[114] C. Zhang J. Wu, M. Liu and Z. Zhou. Correspondence matching and time delay estimation for hand-eye calibration. IEEE Transactions on Instrumentation and Measurement, 69(10): 8304–8313, 2020.

[115] W. Chen K. He Y. Wang F. Zhong, Z. Wang and Y. Liu. Hand-eye calibration of surgical instrument for robotic surgery using interactive manipulation. IEEE Robotics and Automation Letters, 5(2):1540–1547, 2020.

[116] J. Weiss A. Eslami K. Huang M. Maier C. P. Lohmann N. Navab A. Knoll M. Zhou, M. Hamad and M. A. Nasseri. Towards robotic eye surgery: Marker-free, online hand-eye calibration using optical coherence tomography images. IEEE Robotics and Automation Letters, 3(4):3944–3951, 2018.

[117] F. Ernst R. Bruder, F. Griese and A. Schweikard. High-accuracy ultrasound target localization for hand-eye calibration between optical tracking systems and three-dimensional ultrasound. Bildverarbeitung, pages 179–183, 2011.

[118] T. Wang H. Li, Q. Ma and G. S. Chirikjian. Simultaneous hand-eye and robot-world calibration by solving the $ax = yb$ problem without correspondence. IEEE Robotics and Automation Letters, 1(1):145–152, 2015.

[119] G. Dwyer V. Pawar S. Hailes K. Pachtrachai, F. Vasconcelos and D. Stoyanov. Chess—calibrating the hand-eye matrix with screw constraints and synchronization. IEEE Robotics and Automation Letters, 3(3):2000–2007, 2018.

[120] X. Peng G. Yang, W. Haixia and C. Maoyong. Development of high-precise hand-eye calibration software system. Chinese Control Conference, pages 426–429, 2008.

[121] C.-H. Wu. The kinematic error model for the design of robot manipulator. American Control Conference, pages 497–502, 1983.

[122] P.-N. Le and H.-J. Kang. Robot manipulator calibration using a model based identification technique and a neural network with the teaching learning-based optimization. IEEE Access, 8:105447–105454, 2020.

[123] N. Shahriari M. Vendittelli N. Aghakhani, M. Geravand and G. Oriolo. Task control with remote center of motion constraint for minimally invasive robotic surgery. IEEE International Conference on Robotics and Automation, pages 5807–5812, 2013.

[124] J. Zhang Y.-K. Gu, W.-F. Li and G.-Q. Qiu. Effects of wear, backlash, and bearing clearance on dynamic characteristics of a spur gear system. IEEE Access, 7:117639–117651, 2019.

[125] Y. Gao Q. Li Z. Chen H. Zhang, W. Qin and J. Zhao. Disturbance elimination for the modular joint torque sensor of a collaborative robot. Mathematical Problems in Engineering, 2020, 2020.

[126] R. Bearee O. Gibaru A. Olabi, M. Damak and S. Leleu. Improving the accuracy of industrial robots by offline compensation of joints errors. 2012 IEEE International Conference on Industrial Technology, pages 492–497, 2012.

[127] S. J. D. Prince. Computer vision: Models learning and inference. Cambridge University Press, 2012.

[128] M. T. Amo C. Bergeles E. Maneas V. Pawar E. Vander Poorten J. Deprest S. Ourselin P. De Coppi T. Vercauteren G. Dwyer, F. Chadebecq and D. Stoyanov. A continuum robot and control interface for surgical assist in fetoscopic interventions. IEEE Robotics and Automation Letters, 2(3):1656–1663, 2017.

[129] W. Cheng X. Jiang X. Li, C. Sun and Y.-H. Liu. Adaptive vision-based control for robotic tiling with uncalibrated cameras and limited fov. IEEE International Conference on Control and Automation, pages 168–173, 2019.

[130] F. Leali V. Villani, F. Pini and C. Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. Mechatronics, 55:248–266, 2018.

[131] J. R. Llata E. González-Sarabia C. Torre-Ferrero S. Robla-Gómez, V. M. Becerra and J. Pérez-Oria. Working together: A review on safe human-robot collaboration in industrial environments. IEEE Access, 5:26754–26773, 2017.

[132] G. Morel F. Schramm, F. Geffard and A. Micaelli. Calibration free image point path planning simultaneously ensuring visibility and controlling camera path. Proceedings of IEEE International Conference on Robotics and Automation, pages 2074–2079, 2007.

[133] T. Rabie R. Fareh, M. Baziyad and M. Bettayeb. Enhancing path quality of real-time path planning algorithms for mobile robots: A sequential linear paths approach. IEEE Access, 8: 167090–167104, 2020.

[134] Zijian Zhao and Yuncai Liu. Hand-Eye calibration based on screw motions. In 18th International Conference on Pattern Recognition (ICPR'06), volume 3, pages 1022–1026, August 2006.

[135] Aiguo Li, Lin Wang, and Defeng Wu. Simultaneous robot-world and hand-eye calibration using dual-quaternions and kronecker product. International Journal of Physical Sciences, 5 (10):1530–1536, 2010.

[136] Wei Li, Mingli Dong, Naiguang Lu, Xiaoping Lou, and Peng Sun. Simultaneous Robot–World and Hand–Eye calibration without a calibration object, 2018.

[137] Shuwei Qiu, Miaomiao Wang, and Mehrdad R Kermani. A new formulation for Hand–Eye calibrations as Point-Set matching. IEEE Transaction Instrumentation and Measurements, 69(9): 6490–6498, September 2020.

[138] Jinbo Liu, Jinshui Wu, and Xin Li. Robust and accurate Hand–Eye calibration method based on schur matric decomposition. Sensors, 19(20):4490, October 2019.

[139] Amy Tabb and Khalil M Ahmad Yousef. Solving the robot-world hand-eye(s) calibration problem with iterative methods, 2017.

[140] OpenCV: Camera calibration and 3D reconstruction. https://docs.opencv.org/3.4.15/d9/d0c/group__calib3d.html. Accessed: 2021-12-8.

[141] Kenji Koide and Emanuele Menegatti. General Hand–Eye calibration based on reprojection error minimization. IEEE Robotics and Automation Letters, 4(2):1021–1028, April 2019.

[142] Mili Shah. Solving the Robot-World/Hand-Eye calibration problem using the kronecker product. Journal of Mechanisms and Robotics, 5(3), August 2013.

[143] F. L. Zhi Y. Daoyuan A.Y. Elatta, L. P. Gen and L. Fei. An overview of robot calibration. Information Technology Journal, 3(1):74–78, 2004.

[144] H. Bazargani and R. Laganiere. Camera calibration and pose estimation from planes. IEEE Instrumentation and Measurement Magazine, 18(6):20–27, 2015.

[145] K. Konolige E. Rublee, V. Rabaud and G. Bradski. Orb: an efficient alternative to sift or surf. Proceedings of the IEEE International Conference on Computer Vision, 2571.

[146] S. Tareen and Z. Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. Proceedings of International Conference on Computing, Mathematics and Engineering Technologies, pages 1–10. 10.1109/ICOMET.2018.8346440., 6440.

[147] Z. Zhang and W. Lee. Deep graphical feature learning for the feature matching problem. Proceedings of IEEE/CVF International Conference on Computer Vision, pages 5086–5095.

[148] X. Xu Y. Liu and F. Li. Image feature matching based on deep learning. Proceedings of IEEE International Conference on Computer and Communications, pages 1752–1756. 0936.

[149] M. Grimes A. Kendall and R. Cipolla. Posenet: a convolutional network for real-time 6-dof camera relocalization. Proceedings of IEEE International Conference on Computer Vision, utional network for real-time 6-DOF camera relocalization:2938–2946., 2946.

[150] V. Krüger B. Grossmann. Continuous hand-eye calibration using 3d points. Proceedings of IEEE International Conference on Industrial Informatics, pages 311–318, 2017.

[151] S. Michieletto S. Ghidoni E. Menegatti M. Antonello, A. Gobbi. A fully automatic hand-eye calibration system. Proceedings of European Conference on Mobile Robots, pages 1–6, 2017.

[152] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. Kalman filter and its application. In 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), pages 74–77. IEEE, 2015.

[153] S. Maarten. A tutorial on particle filters. Journal of Mathematical Psychology, 73:140–152, 2016.

[154] K. Peng J. Dong R. Jiao. Remaining useful life prediction of lithium-ion batteries based on conditional variational autoencoders-particle filter. IEEE Trans. Instrum. Meas., 69 (11):8831–8843, 2020.

[155] M. Speekenbrink. A tutorial on particle filters. Journal of Mathematical Psychology, 73:140–152, 2016.

[156] DW. Murray G. Klein. Full-3d edge tracking with a particle filter. British Machine Vision Conference, pages 1119–1128, 2006.

[157] M. West J. Liu. Combined parameter and state estimation in simulation-based filtering. In Sequential Monte Carlo methods in practice, pages 197–223, 2001.

[158] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). Doklady ANSSSR, 269:543– 547, 1983.

[159] J.L. Crassidis Y. Oshman. F.L Markley, Y. Cheng. Averaging quaternion. Journal of Guidance, Control, and Dynamics, 30 (4):1193–1197, 2007.

[160] P. S. Tariq M. C. Juan. L. Tiancheng, S. Shudong. Fight sample degeneracy and impoverishment in particle filters: a review of intelligent approaches. Expert Systems with Applications, 41 (8):3944–3954, 2004.

[161] P. M. Djuric T. Li, M. Bolic. Resampling methods for particle filtering: classification, implementation, and strategies. IEEE Signal Processing Magazine, 32(3):70–86, 2015.

[162] O. Cappe R. Douc. Comparison of resampling schemes for particle filtering. Proceedings of the International Symposium on Image and Signal Processing and Analysis, pages 64–69, 2005.

[163] V. Kumar. S. Katoch, S. S. Chauhan. A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80(5):8091–8126, 2021.

[164] T. Sebastian. Particle filters in robotics. Proceedings of the Conference on Uncertainty in Artificial Intelligence, pages 511–518, 2002.

[165] W.F. Denham S. Pines. Sequential estimation when measurement function nonlinearity is comparable to measurement error. AIAA Journal, 4(6):1071–1076, 1966.

[166] L. Wang A. Li and D. Wu. Simultaneous robot-world and hand-eye calibration using dual-quaternions and kronecker product. International Journal of Physical Sciences, 5: 1530–1536, 2010.

[167] M. Foo R.S. Matharu H. Ahmed I. Enebuse, B.K.S.M.K. Ibrahim. An accuracy assessment of hand-eye calibration techniques in uncertain environments for vision guided robots. Proceedings of International Conference on Mechatronics, 2023.