Hand-eye calibration, constraints and source synchronisation for robotic-assisted minimally invasive surgery

Krittin Pachtrachai

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

November 3, 2020

I, Krittin Pachtrachai, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

In robotic-assisted minimally invasive surgery (RMIS), the robotic system allows surgeons to remotely control articulated instruments to perform surgical interventions and introduces a potential to implement computer-assisted interventions (CAI). However, the information in the camera must be correctly transformed into the robot coordinate as its movement is controlled by the robot kinematic. Therefore, determining the rigid transformation connecting the coordinates is necessary. Such process is called hand-eye calibration. One of the challenges in solving the hand-eye problem in the RMIS setup is data asynchronicity, which occurs when tracking equipments are integrated into a robotic system and create temporal misalignment. For the calibration itself, noise in the robot and camera motions can be propagated to the calibrated result and as a result of a limited motion range, the error cannot be fully suppressed. Finally, the calibration procedure must be adaptive and simple so a disruption in a surgical workflow is minimal since any change in the setup may require another calibration procedure. We propose solutions to deal with the asynchronicity, noise sensitivity, and a limited motion range. We also propose a potential to use a surgical instrument as the calibration target to reduce the complexity in the calibration procedure. The proposed algorithms are validated through extensive experiments with synthetic and real data from the da Vinci Research Kit and the KUKA robot arms. The calibration performance is compared with existing hand-eye algorithms and it shows promising results. Although the calibration using a surgical instrument as the calibration target still requires a further development, results indicate that the proposed methods increase the calibration performance, and contribute to finding an optimal solution to the hand-eye problem in robotic surgery.

Impact Statement

In the RMIS setup, solving the hand-eye problem completes the loop of rigid transformations in the environment which allows the information exchange between the camera and robot coordinates. The information includes positions of critical anatomical structure and poses of surgical tools which will provide additional data to the surgical console and in turn assist and facilitate a surgical workflow. Furthermore, given that the two coordinates are already connected, a force feedback with respected to the poses of surgical tools in the environment can be implemented to protect neighbouring tissues and avoid tool clashing.

The proposed algorithms in this thesis update the calibration pipeline to deal with the challenges in solving the hand-eye problem and provide more accurate calibration results in the RMIS environment than existing hand-eye algorithms. The formulated methods can then be used as a model in future researches and the developed constraints provide a path towards accurate hand-eye calibration in RMIS through correctly modelling motion constraints for the calibration. This finding can be implemented into surgical robot systems to design the robot and camera motions for calibrating the hand-eye matrix.

Acknowledgements

As I was writing this, my mind came across a long list of people I should mention as this thesis would not have been possible without the love and support I was given by many people throughout my long PhD journey. It is not possible to name all of you and I must apologise for not being able to, but please acknowledge that I cannot achieve it without you.

First and foremost, I would like to thank Prof. Danail Stoyanov for giving me the chance to do the PhD and to help creating MSc Robotics and Computation in UCL. This past 4 years of doing research under his supervision, designing teaching materials and supervising MSc students have given me invaluable experiences and all of this could not have happened without the opportunity offered by him back in 2015.

Next, I am deeply grateful to Dr. Francisco Vasconcelos whose contributions create and shape this very thesis. There were countless of times we had to discuss research-related problems in length and he has never failed to provide great advice. I can wholeheartedly say that his input is instrumental to my research.

I also would like to express my gratitude to Dr. Simon Di Maio and Dr. Mahdi Azizian at Intuitive Surgical, CA for providing the CAD model of the instrument and the dVRK community for their support. The part contributes to one of the main chapters in this thesis and also a further research idea that can potentially reduce the complication of the hand-eye problem.

Special thanks to Prof. Simon Julier and Prof. João P. Barreto for taking the time in going through the thesis, agreeing to be examiners and giving constructive comments to perfect the work.

To my parents and grandmother, Somchai, Vipa and Poonsri Pachtrachai. I thank you for the moral and also financial supports during the journey. Their words of encouragement and casual family voice calls are the things that keep me going during tough times. I also would like to extend my gratitude to my siblings, Voraprot, Pavaris, Pantharee Pachtrachai, and my sister-in-law, Alexandra Kessel for creating a warm environment whenever I am back in Thailand. Furthermore, special thanks to my girlfriend, Maggie Vong and her family for continuous moral supports and encouragements that never cease to cheer me up.

I wish to gratefully acknowledge the support and assistance from the people in the research group Surgical Robot Vision (SRV), Wellcome/EPSRC Centre for Interventional and Surgical Sciences (WEISS), UCL. With special mentions to Dr. Maximilian Allan and Dr. George Dwyer for always helping me collecting data. To Dr. Agostino Stilli, Emmanouil Dimitrakakis and Beatrice van Amsterdam for helping me with the workload from MSc Robotics. To Dr. Lukas Lindenroth, Dr. Evangelos Mazomenos, Patrick Brandão, Mirek Janatka and Claudia D'Ettorre for always inviting me to socialise with the group, albeit my constant turning down. Finally, I wish to express my thanks to my friends in the UK, the US, Thailand, Japan, Sweden, and Germany whom I have not mentioned and have always been there for me throughout the journey.

Contents

List of Figures

List of Tables

Notation

The following notations will be used throughout the thesis.

x	a scalar variable.
$\vec{x}$	A vector.
$\vec{0}, \mathbf{0}_N$	A zero vector and a zero square matrix of size N.
$[\vec{x}]_\times$	Skew symmetric matrix of a vector $\vec{x}$.
$SO(3)$	Special orthogonal group.
$SE(3)$	Special Euclidean group.
$se(3)$	Lie algebra of $SE(3)$.
$F_1, F_2, F_{\text{cam}}, \ldots$	Frames assigned at the coordinate frame 1, 2, cam, respectively
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$	An arbitrary matrix with the exception of the letters $\mathbf{R}$.
$\mathbf{R}$	An arbitrary rotation matrix in $SO(3)$.
$\mathbf{K}(\cdot)$	A matrix as a function of the bracketed parameters.
${}^1\mathbf{T}_2$	A rigid transformation mapping from frame 2 to frame 1.
$\mathbf{I}_N$	An identity matrix of size N.
$\mathfrak{a}, \mathfrak{b}$	4×4 Lie algebra for rigid transformations $\mathbf{A}$ and $\mathbf{B}$, respectively.
$\mathbf{a}, \mathbf{b}$	Real terms of dual quaternion representations of the transformations $\mathbf{A}$ and $\mathbf{B}$, respectively.
$\mathbf{a}', \mathbf{b}'$	Dual terms of dual quaternion representations of the transformations $\mathbf{A}$ and $\mathbf{B}$, respectively.
$\mathbf{a}.\mathbf{b}$	Quaternion multiplication between two quaternions $\mathbf{a}$ and $\mathbf{b}$.

Acronyms

The following acronyms will be used throughout the thesis.

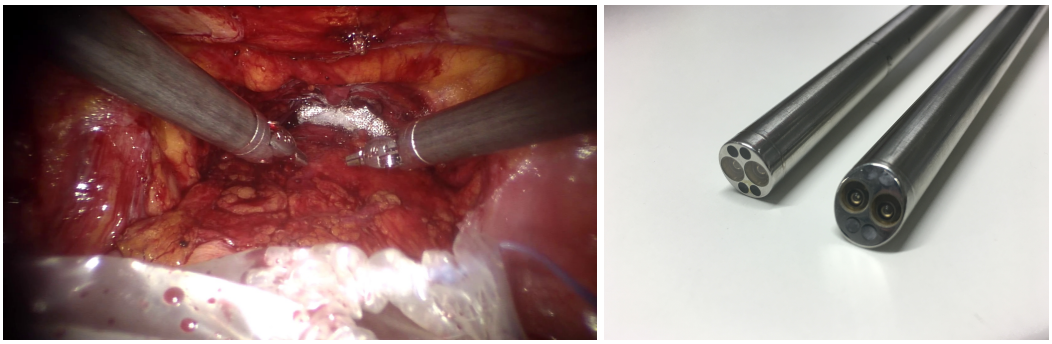
RMIS	Robotic-assisted minimally invasive surgery
CAI	Computer-assisted interventions
RCM	Remote centre of motion
ROS	Robot operating system
ANOVA	Analysis of Variance
DH	Denavit-Hartenberg
POE	Product of exponential
SfM	Structure from motions
SLAM	Simultaneous Localisation and Mapping
SVD	Singular Value Decomposition
RANSAC	Random sample consensus
PSM	Patient side manipulater
ECM	Endoscope control manipulator
CMOS	Complementary metal–oxide–semiconductor
RMS	Root mean square
DoF	Degree of freedom
FIR	Finite impulse response
FFT	Fast Fourier transform
GCC	Generalised cross-correlation
CAD	Computer aided design
dVRK	da Vinci Research Kit

Chapter 1

Introduction

Surgical practice has been transformed by the use of miniaturised tools to provide access to an operative site with a small incision and less trauma. The technique benefits patients in terms of post-operative recovery time as the procedure has become less invasive. Combining with robotic technology, which has emerged as a solution to challenging problems in many fields over the past decade [1], a technique called robotic-assisted minimally invasive surgery (RMIS) is created. A clear example of such systems can be seen in one of the commercialised surgical robots, da Vinci®(Intuitive Surgical, CA). Deploying the robot allows an operation to be remotely carried out by a surgeon, who controls articulated surgical instruments and visualises the operative site with a stereo camera [2–5], as shown in Figure 1.1(a)-1.1(b). RMIS has already been used in several procedures such as prostatectomy [6], thoracic surgery [7], colorectal surgery [8], hysterectomy [9], cardiac surgery [10], nephrectomy [11].

The RMIS setup introduces the potential to implement a platform for computer-assisted interventions (CAI) such as augmented reality [12, 13]; directly overlaying an intra- and pre-operative image onto the feedback from the camera to provide additional visualisation of information about anatomical structure and the location of sub-surface tumours and blood vessels (see Figure 1.2). In [14], the authors suggest that the technique can be used to assist novice surgeons in localising the operative site as they may have difficulty in performing RMIS; the removal of tactile feedback reduces a surgeon's awareness of anatomical structures that cannot



(a) (b)

Figure 1.1: (a) An example of what surgeons see from the console when controlling the movement of surgical instruments and a scope. (b) Tips of 0 degree and 30 degrees laparoscopes. A laparoscope works as an imaging sensor providing a view of an operative site. In RMIS, a laparoscope is attached to one of the manipulators to provide surgeons a full control of laparoscope's movement whilst performing an operation.

be visualised by a laparoscopic or endoscopic camera. This technique, coupled with external tracking systems and/or a manual registration, has already been tested in clinical interventions such as liver segmentectomy (in three patients) [15] and prostatectomy [14]. This combination of robotic surgery and an image-guided technique has been shown to be useful in the localisation of lesions and important neighbouring anatomical structures [16–18] and is likely to increase a patient's safety as the accuracy of the operation is increased [12, 13].

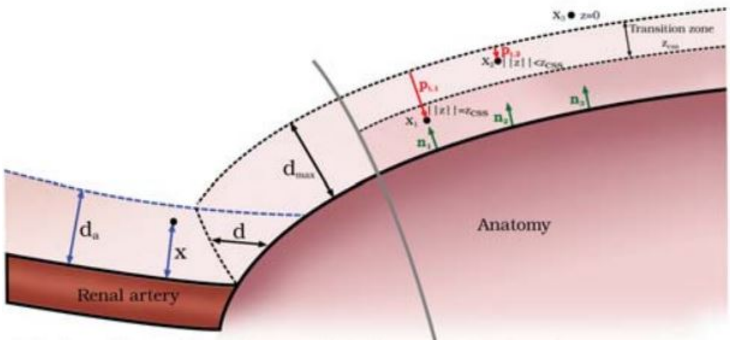
One of the other key CAIs is virtual fixtures, also known as active constraints or motion constraints [21]. Given that a surgeon is physically disconnected from an operative site in an RMIS environment (and therefore given limited haptic sensation) [22], the risk of intra-operative injury is increased. Indeed, this has been reported to be one of the major challenges in RMIS [23,24]. Therefore, virtual fixtures can be applied during surgical training to increase the subject's proficiency [25] in a real surgery. Examples of this technique are shown in Figure 1.3. Virtual fixtures assist in manipulation tasks by anisotropically regulating a surgeon's motions. The assistance can constrain the movement of the end-effector in a restricted region [19, 20, 26–28] with or without guiding of the movement along pre-defined trajectories [22, 29, 30]. The force generated using virtual fixtures can also protect

Figure 1.2: An example of an augmented reality application during laparoscopic partial nephrectomy. The colour-coded model is overlaid onto the small tumour. In the operation, surgeons need to make a cut within a green zone, whilst simultaneously preserving the blue zone. This is achieved by registering ultrasound imaging to the pre-operative CT volume data and using an electromagnetic tracking system to localise the probe [14].

neighbouring tissue or prevent tool clashing [31] by generating a repulsive force, and it can provide a more dexterous end-effector movement at an operative site, which in turn increases safety and potentially accelerates an operation by guiding tool movement along a pre-defined trajectory [20]. The technique has already been tested during surgical training, and the evaluation suggests that the end-effector movement is more precise [32].

Visual servoing is another potential CAI in RMIS. This the technique relies on using imaging information from a camera to control the robot's motion [35]. Since robots offer a precise and automatic positioning system, visual servoing creates the potential for robot assisted surgery with automated tasks; for instance, automatic camera and tool positioning (see Figure 1.4 [33]), tremor compensation [36] and motion compensation [37, 38]. The technique can therefore reduce a surgeon's workload and fatigue during an operation because the procedures are typically time-consuming [33]. Although implementations of this technique have not been translated to a real surgical robot due to other problems such as registration, tracking and real-time capability [39], visual servoing presents the potential to partially automate surgical tasks, which can facilitate workflow in the operating room.

However, all the information necessary for the interventions is captured in the

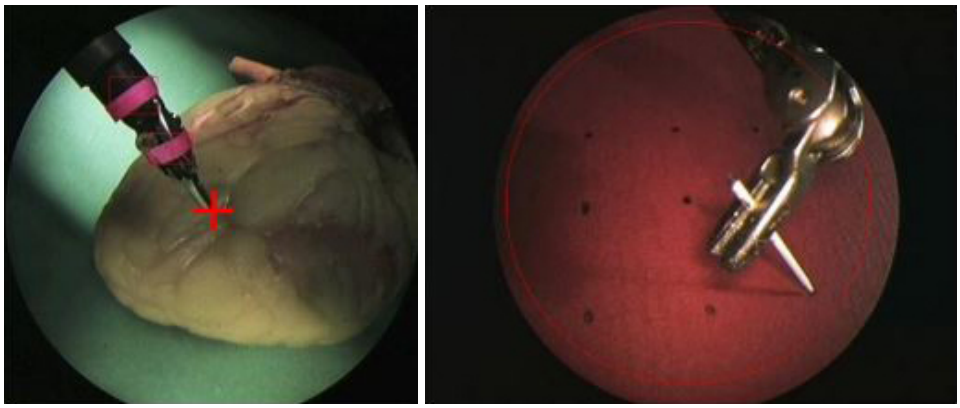


(a)

(b)

Figure 1.3: (a) An illustration of virtual fixtures. When a surgical tool is moving into renal artery neighbourhood (dashed blue line), the active repulsive force will be generated to protect delicate anatomical structure (dashed black line) [19]. (b) An illustration of a robotic-assisted trajectory using virtual fixtures. The red line represents a pre-defined trajectory that the end-effector has to follow [20].

camera coordinate frame, which is not yet linked to the robot coordinate frame. This presents a problem because useful information in the camera frame cannot be directly accessed and updated by the robot kinematic, which surgeons use to control the instruments and camera. Therefore, the mapping between the coordinates of the laparoscopic camera and the robot must be estimated. Such mapping is called the hand-eye transformation. When the transformation is known, information exchange, which links the information happening in the camera reference using the surgical vision frame [40] to the robot kinematics becomes possible.



(a) (b)

Figure 1.4: (a) Automatic camera re-positioning using visual servoing technique. The red cross represents the centre of the field of view and the algorithm positions the end-effector of the tool at the centre of the frame. (b) A laparoscopic view of a surgical tool holding a needle. Visual servoing is used to position the needle at the target. Both examples require an imaging feedback from a camera whilst accordingly controlling the robot through the kinematic to achieve the task. This can be developed further to an automated suturing in laparoscopy [33, 34].

1.1 Problems in hand-eye calibration

Hand-eye calibration is a classic problem in the robotics field that aims to identify the transformation that maps two rigidly attached frames, usually a robot end-effector (hand) and a camera (eye). Many hand-eye calibration algorithms have been proposed since the 90s and they offer accurate calibration result, but hand-eye calibration is still a challenging problem when working with RMIS due to several factors.

1.1.1 Noise sensitivity

The hand-eye problem is conventionally constructed from a chain of rigid transformations. Without noise in the system, the problem can be solved algebraically in the least squares sense to recover the hand-eye transformation [41]. This is not the case in the real system when the position of the robot slightly deviates and the images are noisy. Moreover, the motions of the robot and the camera in an operative site are usually small and heavily perturbed by different noise characteristics. For example, the noise in the KUKA motors is different from that of the da Vinci robot and the latter is more difficult to model. Despite this uncertainty in the

input, the algorithm should be able to yield sub-millimetre calibration accuracy as the accuracy requirement for CAI applications are very high, which ensures that the information is accurately mapped between the two coordinate frames. Therefore, in the context of RMIS, hand-eye calibration is sensitive to noise and more challenging in practice. This presents a problem that currently available hand-eye algorithms cannot solve, instead ending up with an inaccurate hand-eye matrix. There are two main sources of noise in the calibration: camera motion and robot motion.

The noise in camera motions occurs mostly because of the error in the camera calibration which can be caused by several factors; an erroneous grid detection algorithm, an inaccurate intrinsic and extrinsic parameters estimation, poor quality imaging devices. Although the error in the camera motion is often considered negligible in the field of computer vision as the camera parameters can be refined to yield sub-pixel accuracy [42], the error creates a larger impact on the hand-eye matrix estimation, as shown in the experimental sections. However, the finding of the intrinsic parameters are not the focus of this thesis as we are mainly looking for the constraints derived from the kinematic chain of robotic systems.

On the other hand, due to geometric (imprecision in manufacturing) and non-geometric errors (backlash, elasticity, and joint compliance) caused by stresses and strains from surgical instruments, the measurement of joint positions is not perfectly accurate. This creates a discrepancy between the transformation computed by the forward kinematics of the robot model and the real robot pose [43]. Although currently available robot arms have sub-millimetre repeatability in their positioning systems, this does not guarantee their accuracy. Furthermore, according to experiments in the literature [41, 44, 45], it has been shown that even a sub-millimetre error in the translation component has a noticeable impact on the calibration results. In practice, although this effect can be minimised with a more accurate robot calibration [46], the calibration of the robot has to be repeated when there is a slight change in the system as such a change invalidates the previous calibration. Therefore, including a robot calibration in a robotic system that requires setup changes may not be practical.

1.1.2 Limited motion range

Apart from noise sensitivity in the system, motion range is also a challenging problem in the hand-eye problem. A wide input motion range suppresses the effect of noise and in turn increases calibration accuracy [44], but such a range is not feasible in all surgical robot setups. The motions of surgical tools are usually very small as they are confined around the remote centre of motion (RCM) to ensure that instruments are confined to a motion in the vicinity of the trocar entry ports [47].

This mechanical constraint is designed to minimise a chance of robot arm damaging the surrounding tissues at the entry port [48, 49]. For example, da Vinci systems fix the RCM at the intersection of the rotation axes of the last setup joint and the first joint [50, 51]. The system also provides a specific calibration instrument dedicated for placing the RCM at the right position in a real surgery [52]. On the other hand, for robot arms that do not have this mechanical constraint such as KUKA arms, software approaches [52,53] can be used to ensure that the instrument pivots around the designated RCM.

Apparently, the systems without the constraint can move more freely and has a wider motion range than the ones with the constraint. For instance, the camera motion captured from the robot arm KUKA LBR IIWA R800 has a significantly wider motion range than the one from the da Vinci surgical robot as shown in Chapter 3. Therefore, while the conventional formulation for the hand-eye problem works well in an industrial context, it poses a problem for the RMIS setup for which the scale of the workspace is significantly smaller.

Furthermore, a small motion range can also transform the well-posed hand-eye problem into an ill-posed problem, which is more difficult to solve as it is underconstrained. Although the classic formulation of the problem has been used since the 1980s and many solutions have been proposed using different mathematical domains, the degree of degeneracy and ill-posed configurations remain the same across all algorithms as the original formulation is never changed.

Alternatively, path planning algorithms can ensure that the appropriate calibration motions are captured [54], and automated pose selection methods can guar-

antee well-conditioned hand-eye constraints [55, 56]. However, when the robot is confined to a limited range, the problem remains [57].

1.1.3 Asynchronous data streams

The camera attached to the robot and the robot itself are usually operated at different frequencies. This creates a temporal signal misalignment in the data stream which makes the well-posed hand-eye formulation invalid, since the hand-eye problem requires a corresponding robot motion for any given camera motion [58]. Nevertheless, it is arguable that the synchronisation step is not necessary for the hand-eye problem because of the following reasons.

First, the calibration step is typically an offline procedure; users can prepare the setup and perform the calibration beforehand by discretely capturing several sets of images of the calibration grid and the corresponding kinematic data, such that a temporal misalignment does not affect the data. However, this option does not work if there is a need for a re-calibration which arises when there is a change in the setup. For example, a laparoscopic lens can be fogged because of the difference in temperature between the operating room and a patient's peritoneum or insufflation gas [59, 60], which creates a need to clear the visual field. A change in the setup invalidates the previous calibration result. Moreover, although users can still discretely capture the data for the calibration after the setup change, the new temporal misalignment is not known and the resulting calibration process (which is already complicated) will further disrupt the operation.

Second, an embedded system producing a common timestamp can alternatively be implemented into an RMIS system such that every operation and communicated message is synchronised, which allows users to either manually synchronise the data or programmatically only obtain data based on the registered timestamp. One of the closest configuration to the description and most commonly known software architectures is Robot Operating System (ROS), although it is not a real-time system and not yet viable for a real surgery. However, this requires compatibility between every sensor and camera in the system and the embedded system which cannot always be achieved. Moreover, the hand-eye problem does not only apply

to a camera-robot setup but also to every setup that has more than one working coordinate system [61–63]. Therefore, as robotic systems become more complex and distributed and while support packages for their sensors have not yet been developed, the common timestamp for the whole system may not be available [64].

1.1.4 Usability

In computer vision and robotics related applications, regardless of the equipment used, performing calibration is generally a necessary step before deployment to ensure the equipment's accuracy, the connection in the systems, and the system's functionality [42, 44, 56, 65]. The problem is usually expressed into equations and therefore requires users to gather the data for the calibration. There is a series of criteria that the data must satisfy.

All of the existing hand-eye calibration methods require several images of a physically known object, which usually consist of a series of images of a calibration grid taken with a camera along with the corresponding robot poses. The point of view of the camera must be varied as much as possible to maximise calibration accuracy.

Therefore, in addition to introducing a calibration grid as a part of RMIS, the requirement for the calibration also poses a problem in an operation as the medical staff in an operation room are not properly trained to resolve a computer vision and robotics problem. The calibration procedure must therefore be simplified and minimal such that medical staff can perform the calibration routine without the need to consult a technician.

1.1.5 Adaptability

During an operation, surgeons may need to change surgical tools and scopes when it is appropriate as mentioned earlier in Section 1.1.3, but any change in a robotic system invalidates the previous calibration result, i.e. a relative pose between robot and camera coordinate systems are changed. This requires another calibration procedure to find the correct transformation. Some may argue that the change may be insignificant and the need for adaptive-to-change hand-eye algo-

rithm is not necessary, but even sub-millimetre error can be translated into an observable discrepancy in poses and in turn the calibration result [41, 44, 45, 66]. In order to guarantee that the mapping between the two coordinate frames are as accurate as possible, every change in the robotic system must be accounted for.

However, it is not feasible to repeat the whole calibration process for every change in the setup. Surgeons have to reuse a calibration grid which has to be resterilised and recreate the scene for the calibration. This could cause a disruption in the surgical workflow and delays in an operation which in turn potentially impact on patient safety. Therefore, the hand-eye algorithm should be adaptive to changes of the setup and able to update the parameters accordingly [67] so that the workflow during surgical procedures are not affected or are affected as least as possible by the calibration.

1.2 Contributions

This thesis aims to develop a modification to the hand-eye calibration pipeline specifically for the surgical environment. The experiments will be conducted with synthetic and real data from the KUKA LBR IIWA 7 R800, IIWA 14 R820 and the da Vinci Surgical Robot in order to validate the findings by comparing performance with existing hand-eye calibration algorithms.

Analysis of variance (ANOVA) is applied to the raw experimental results to verify whether the comparison is statistically significant and the p-value is reported accordingly when it is greater than 0.05. ANOVA is used here because the performance of the proposed algorithm is compared with the existing methods using the same input data and the normal distribution is assumed. The code for the work and the comparison studies are packaged in a repository which is available online at `https://github.com/surgical-vision`.

The contributions of the thesis are listed as follows.

- **Re-formulate the hand-eye problem**: A new formulation for solving the hand-eye problem is developed based on the screw motion theory which has been commonly used in robotic applications. Using the screw motion

to define transformations avoids ill-posed configurations in rigid transformations [68, 69].

- **Investigate the effect of stereo information in the hand-eye problem**: Stereo information provides three additional constraints to camera motion. In a real system where mappings of coordinate systems are not exact, the additional constraints can reduce uncertainty in the camera poses as they all define the same rigid transformation. Both monocular and stereo laparoscopes are used throughout this research to validate the difference in the performance.

  [66] *K. Pachtrachai, F. Vasconcelos, F. Chadebecq, M. Allan, S. Hailes, V. Pawar and D. Stoyanov. Adjoint transformation algorithm for hand-eye calibration with applications in robotic assisted surgery. Annals of Biomedical Engineering, 46(10): 1606-1620, Oct 2018.*

- **Develop a formula for hand-eye calibration using RCM**: The RCM in a robotic system is usually defined by the user to specify the location the camera pivots around. Camera motion in RMIS is confined to the area surrounding this location, which does not provide a sufficient motion range for the calibration. We use the user-defined RCM to develop a formulation that is less sensitive to the input motion range to deal with the problem.

  [70] *K. Pachtrachai, F. Vasconcelos, G. Dwyer, S. Hailes, and D. Stoyanov. Hand-eye calibration with a remote centre of motion. IEEE Robotics and Automation Letters, 4(4):3121–3128, Oct 2019.*

- **Develop a synchronisation algorithm to pre-process data streams for the calibration**: We explore the constraint embedded in the conventional hand-eye formulation to check if there is any dependency between the camera and robot motions. The synchronisation algorithm is developed from the relationship between the two motions.

  [58] *K. Pachtrachai, F. Vasconcelos, G. Dwyer, V. Pawar, S. Hailes, and D. Stoyanov. Chess—calibrating the hand-eye matrix with screw constraints and*

synchronization. IEEE Robotics and Automation Letters, 3(3):2000–2007, July 2018.

- **Investigate the feasibility of using other objects as a calibration target**: The calibration grid is commonly used in calibration problems since it is easy to detect using the grid detection algorithm [71]. Also, its physical dimension is known which allows us to determine the mapping between the dimension in the real world and the camera frame using the pinhole camera model [42]. If we apply the same principle to an arbitrary object, we can use any object as a calibration object. This will considerably reduce the complexity in the calibration procedure which will potentially solve usability and adaptability issues, provided we use a calibration object that is familiar to medical staff. We apply the tracking method in [72] to determine the pose of the object and evaluate the performance with the conventional calibration algorithm.

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

Following this chapter, the mathematical background necessary for studying the hand-eye problem and the literature on the problem are introduced in Chapter 2. Then, the new hand-eye formulation using the adjoint transformation, alternating optimisation and stereo information is presented in Chapter 3, which shows the improvement in calibration performance compared to the methods described in the literature. Chapter 4 and 5 describe the potential formulations and solutions to mitigate the problem of limited motion range that exists in the current hand-eye formulation and to reduce the complexity of the calibration procedure. Data synchronisation and recovery methods are elucidated in Chapter 6 which act as a data pre-processing step to optimise the input for the calibration. Finally, discussions and future studies are detailed in Chapter 7.

Chapter 2

Mathematical background and the hand-eye problem

2.1 Rigid transformation representations

The mathematical representations and the concept of rigid transformations are major parts of the formulation and experimentation presented in this thesis. Hand-eye algorithms use different representations to formulate the problem, which yield different calibration accuracy. This chapter explains the core concept of rigid transformations and the commonly used representations of the rotation component in 3D space.

A rigid transformation is a linear mapping in Euclidean space, consisting of a rotation matrix $\mathbf{R}$ (a 3×3 matrix) and a translation component $\vec{t}$ (a 3×1 vector). The set of rigid transformations are defined as a special Euclidean group, $SE(3)$. Transformations are used to express the relationship between the two coordinate systems. For instance, from a robot base to each joint of a robot, or from a robot coordinate system to a camera coordinate system. The general form of an arbitrary rigid transformation ${}^{1}\mathbf{T}_2$ mapping between two arbitrary frames F_2 and F_1 is,

$$ {}^{1}\mathbf{T}_2 = \begin{bmatrix} \mathbf{R} & \vec{t} \\ \vec{0}^T & 1 \end{bmatrix} \tag{2.1} $$

To apply a transformation to a point $\vec{v} = [v_x, v_y, v_z]^T$, the point must be ex-

pressed in normalized homogeneous coordinates with the last entry equal to 1. A transformed point is simply calculated by matrix multiplication,

$$\begin{bmatrix} \vec{v}_{\text{new}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \vec{t} \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} \vec{v} \\ 1 \end{bmatrix} \tag{2.2}$$

Alternatively, Eq. 2.2 can also be represented as a matrix equation,

$$\vec{v}_{\text{new}} = \mathbf{R}\vec{v} + \vec{t} \tag{2.3}$$

Unlike the translation component in a rigid transformation which is only a 3×1 vector, a rotation operator consists of an axis of rotation and a degree of rotation and can be represented in different domains depending on the type of applications. In this thesis, we are interested in the rotation matrix, quaternions, dual quaternion and Lie Algebra and the last two representations involve the translation component in the conversion.

2.1.1 Rotation matrix

In 3D space, a rotated component is represented by a matrix multiplication between a rotation matrix and a vector, $\mathbf{R}\vec{v}$. The matrix $\mathbf{R}$ is a 3×3 matrix and is a linear map defining a projection of one basis in Euclidean space to another basis. The matrix is usually defined as,

$$\mathbf{R} = \begin{bmatrix} \vec{x}_1 \cdot \vec{x}_0 & \vec{y}_1 \cdot \vec{x}_0 & \vec{z}_1 \cdot \vec{x}_0 \\ \vec{x}_1 \cdot \vec{y}_0 & \vec{y}_1 \cdot \vec{y}_0 & \vec{z}_1 \cdot \vec{y}_0 \\ \vec{x}_1 \cdot \vec{z}_0 & \vec{y}_1 \cdot \vec{z}_0 & \vec{z}_1 \cdot \vec{z}_0 \end{bmatrix} \in SO(3) \tag{2.4}$$

where $\vec{x}_i, \vec{y}_i$ and $\vec{z}_i$ are the orthonormal bases in Euclidean space. This matrix contains several properties that are useful in hand-eye calibration:

- $\mathbf{R}^T = \mathbf{R}^{-1} \rightarrow \mathbf{R}^T\mathbf{R} = \mathbf{I}_3$
- The column and row vectors in the rotation matrix $\mathbf{R}$ are mutually orthogonal.
- Each vector and row vector in the rotation matrix $\mathbf{R}$ is a unit vector.

- The determinant of a rotation matrix is always 1.

A composition of two rotation matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ can be computed by a matrix multiplication $\mathbf{R}_1\mathbf{R}_2$. Note that the operation is not commutative, i.e. $\mathbf{R}_1\mathbf{R}_2 \neq \mathbf{R}_2\mathbf{R}_1$.

Although there are nine elements in this representation, the properties of the rotation matrix create six constraints on the elements which only leave three degrees of freedom (DoF) for the rotation matrix. The other representations such as angle-axis representation and Euler-angle representation use different formulations of the elements in the rotation matrix, but they are not well-defined and present an ill-posed problem in a certain configuration. For example, angle-axis is not well-defined as there are more than one representations, $(\vec{r}, \theta)$ and $(-\vec{r}, -\theta)$ representing the same rotation where $\vec{r}$ is the axis of rotation and θ is the angle of rotation.

Euler-angle representation is also not suitable for the calibration problem as the optimisation will lose one degree of freedom when one rotation component is aligned with the other, i.e. the change in the remaining two rotation components only results in 1 DoF rotation. This situation should be avoided in an optimisation problem like hand-eye calibration as it is possible that a parameter in one DoF could converge faster than the others and cause the problem to lose DoF. Therefore, these representations will not be considered in this thesis.

2.1.2 Quaternions

Quaternions are four-dimensional number systems containing four real numbers. A quaternion is usually written as,

$$\mathbf{q} = q_w + q_x i + q_y j + q_z k \tag{2.5}$$

where i, j, k are the imaginary terms such that $i^2 = j^2 = k^2 = ijk = -1$. A quaternion can also be written as a composition of a vector and a real number,

$$\mathbf{q} = [q_w, q_x, q_y, q_z]^T = [q_w, \vec{q}^T]^T \tag{2.6}$$

Multiplication between two quaternions **p** and **q** uses vector dot and cross products in the computation. This operation is not commutative.

$$\mathbf{p}.\mathbf{q} = \begin{bmatrix} p_w q_w - \vec{p}^T \vec{q} \\ q_w \vec{p} + p_w \vec{q} - \vec{p} \times \vec{q} \end{bmatrix} = \begin{bmatrix} p_w & -p_x & -p_y & -p_z \\ p_x & p_w & -p_z & p_y \\ p_y & p_z & p_w & -p_x \\ p_z & -p_y & p_x & p_w \end{bmatrix} \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{2.7}$$

Unit quaternion, **q** represents rotation components. The form of this quaternion can be written as,

$$\mathbf{q} = \cos\frac{\theta}{2} + (r_x i + r_y j + r_z k)\sin\frac{\theta}{2} \tag{2.8}$$

where θ is the angle of rotation and $[r_x, r_y, r_z]^T$ is a unit vector representing an axis of rotation. Similarly to the rotation matrix, a successive rotation in the quaternions domain is represented as quaternion multiplication and the operation is not commutative. To apply a rotation to a vector $\vec{v} = [v_x, v_y, v_z]^T$, one has to evaluate the following product.

$$\vec{v}_{\text{new}} = \mathbf{q}.[0, v_x, v_y, v_z]^T.\mathbf{q}^\star \tag{2.9}$$

where $\mathbf{q}^\star$ is a quaternion conjugation of **q**,

$$\mathbf{q}^\star = \cos\frac{\theta}{2} - (r_x i + r_y j + r_z k)\sin\frac{\theta}{2} \tag{2.10}$$

Quaternion representation is widely used in the fields of robotics and computer vision. The representation does not have a gimbal lock problem, and it is computationally cheaper to interpolate the rotational motion in the quaternion domain than in the rotation matrix.

2.1.3 Dual quaternions

So far we have only considered the expression of the rotation component of a rigid transformation in the different domains. To represent a complete rigid transformation, another quaternion, called the dual term $\mathbf{q}'$, is introduced in addition to

the rotation term. Dual quaternions have the following form,

$$\hat{\mathbf{q}} = \mathbf{q} + \varepsilon \mathbf{q}' \tag{2.11}$$

where ε is a dual unit such that $\varepsilon^2 = 0$. The real term $\mathbf{q}$ represents a rotation component and its conversion remains the same in this representation (Eq. 2.8). The dual term $\mathbf{q}'$ is calculated by the quaternion multiplication $[0, \frac{1}{2}\vec{t}^T]^T.\mathbf{q}$ and has the form of,

$$\mathbf{q}' = \begin{bmatrix} -\vec{t}^T \vec{r} \sin\frac{\theta}{2} \\ \vec{t}\cos\frac{\theta}{2} + \vec{t} \times \vec{r} \sin\frac{\theta}{2} \end{bmatrix} \tag{2.12}$$

Similarly to the quaternion representation of a rotation matrix, successive transformations can be performed in the same manner. The resulting transformation is represented by a chain of dual quaternions multiplied in the same order as a chain of rigid transformations. The multiplication, in this case, is the quaternion multiplication.

The dual quaternion representing a rigid transformation has an orthogonal property between the real and the dual terms. The dot product between the two quaternions must equal zero, i.e. $\mathbf{q}^T \mathbf{q}' = 0$. This property is used in hand-eye calibration algorithms incorporating a dual quaternion formulation [73].

2.1.4 Lie algebra

To express a transformation in its Lie group $se(3)$, we have to use the matrix logarithm (logm) and the matrix exponential (expm) to map from $SE(3)$ to $se(3)$. Let $\mathfrak{t}$ be a Lie algebra of a transformation $\mathbf{T}$, we have,

$$\mathfrak{t} = \text{logm}\left(\begin{bmatrix} \mathbf{R} & \vec{t} \\ \vec{0}^T & 1 \end{bmatrix}\right) = \begin{bmatrix} [\vec{\omega}]_\times & \vec{v} \\ \vec{0}^T & 0 \end{bmatrix} \tag{2.13}$$

where $\vec{\omega}$ is a 3×1 vector representing the axis of rotation with a norm equal to the degree of rotation θ and the vector $\vec{v}$ is calculated by the following formula,

$$\vec{v} = (\mathbf{I}_3 + \frac{1-\cos\theta}{\theta^2}[\vec{\omega}]_\times + \frac{\theta - \sin\theta}{\theta^3}[\vec{\omega}]_\times^2)^{-1}\vec{t} \tag{2.14}$$

The conversion back to the $SE(3)$ domain uses the exponential map. The translation component of the transformation is calculated by pre-multiplying the matrix in Eq. 2.14 to the left-hand side of the equation. On the other hand, the rotation component has to be calculated using Rodrigues' formula.

$$\mathbf{R} = \mathbf{I}_3 + \frac{\sin\theta}{\theta}[\vec{\omega}]_\times + \frac{1-\cos\theta}{\theta^2}[\vec{\omega}]_\times^2 \tag{2.15}$$

$$\mathbf{T} = \mathrm{expm}(\begin{bmatrix} [\vec{\omega}]_\times & \vec{v} \\ \vec{0}^T & 0 \end{bmatrix}) \tag{2.16}$$

2.2 Hand-eye problem formulation

The hand-eye calibration problem was first introduced in the paper by Tsai and Lenz [44]. The calibration aims to determine the unknown rigid transformation $\mathbf{X}$ that maps a robot coordinate system to a camera coordinate system or vice versa depending on the formulation. The problem arises when a camera is rigidly attached to a robot arm as shown in Figure 2.1. It is conventionally formulated as a linear system of equations [41].

$$\mathbf{AX} = \mathbf{XB} \tag{2.17}$$

$$\begin{bmatrix} \mathbf{R}_A & \vec{t}_A \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_X & \vec{t}_X \\ \vec{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_X & \vec{t}_X \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_B & \vec{t}_B \\ \vec{0}^T & 1 \end{bmatrix} \tag{2.18}$$

where $\mathbf{A}$ represents the relative motion of a camera between F_{cam} and an observed reference frame F_{grid} at a calibration object, usually a calibration grid, and $\mathbf{B}$ is the relative motion of a robot arm between F_{robot} and a fixed reference frame at the base of the robot F_{base}. These transformations can be written as the product of two rigid transformations,

$$\mathbf{A} = {}^{\text{cam}}\mathbf{T}_{\text{grid}}(\tau)({}^{\text{cam}}\mathbf{T}_{\text{grid}}(\tau'))^{-1} \tag{2.19}$$

$$\mathbf{B} = {}^{\text{robot}}\mathbf{T}_{\text{base}}(\tau)({}^{\text{robot}}\mathbf{T}_{\text{base}}(\tau'))^{-1} \tag{2.20}$$

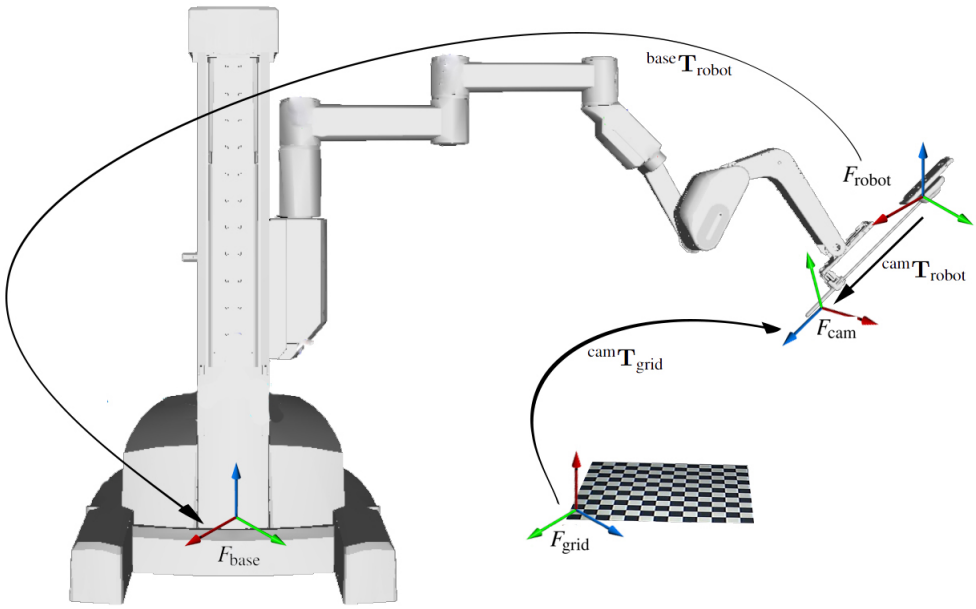


Figure 2.1: Experimental setup for the classic hand-eye calibration procedure with a da Vinci surgical robot system. The camera is attached to the end-effector and $^{cam}\mathbf{T}_{robot}$ is the camera's pose relative to the end-effector [50].

where τ and τ' are discrete time values indicating that the two transforms are captured at different time instances. Therefore, with N different measurements, Eq. 2.19 and 2.20 can be established for all different pairwise combinations and create $\binom{N}{2}$ different hand-eye equations which can be used together to solve the problem. However, in a noise-free case, only two motions whose rotation axes are not parallel can be used to solve the problem [44].

Another formulation ($\mathbf{AX} = \mathbf{YB}$) was first proposed in [74] which aims to determine both hand-eye and robot-world transformations simultaneously. In this formulation, the matrices $\mathbf{A}$ and $\mathbf{B}$ are defined differently from Eq. 2.17; $\mathbf{A}$ is a camera pose with respect to the grid coordinate and $\mathbf{B}$ is a robot pose with respect to the robot base. Several solutions have been proposed to solve the equation [62, 75–77]. However, the original paper proves that the formulation contains similar characteristics as the original hand-eye equation in terms of the rotation estimation. Furthermore, regarding the error propagation on the translation estimation, it is shown in [78] that although the revised formulation slightly outperforms the original formulation when the degree of rotation is less than 90 degrees, the superiority is not

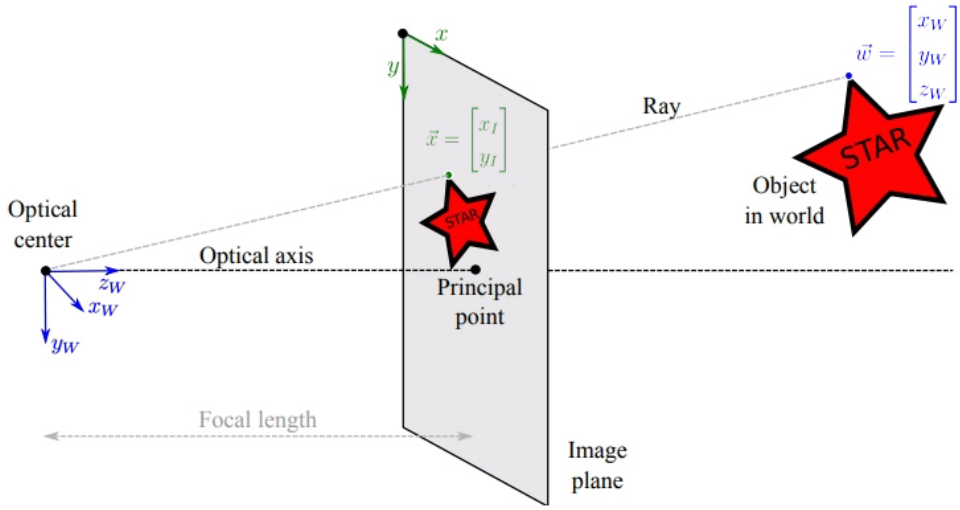


Figure 2.2: The schematic shows the pinhole camera model, commonly used in the field of computer vision. The optical centre is located at the world coordinate system (u, v, w). The distance between the image plane (virtual image) and the optical centre is defined as the focal length [42].

significant and the equation is more sensitive to ill-posed configuration than the original equation.

2.3 Data collection for hand-eye calibration

To solve the hand-eye problem, the rigid transformations ${}^{\text{cam}}\mathbf{T}_{\text{grid}}$ and ${}^{\text{robot}}\mathbf{T}_{\text{base}}$ must be determined beforehand in order to compute the relative transformations **A** and **B**, respectively. This section explains the methods used to compute these transformations together with the criteria for the selection of pairs of transformations for the construction of a set of optimal hand-eye equations.

2.3.1 Extrinsic parameters estimation

Extrinsic parameters estimation is one of the data pre-processing steps in hand-eye calibration. The aim is to determine ${}^{\text{cam}}\mathbf{T}_{\text{grid}}$ for Eq. 2.19 and 2.20. Given a pre-calibrated camera, i.e. a case in which the intrinsic parameters of a camera are known, the method determines the pose of the camera with respect to a fixed reference frame, usually defined on a calibration object [42].

We adopt the pinhole camera model to map a point in the world coordinate system to the camera coordinate system. The point in the world coordinate is trans-

formed by an extrinsic matrix and then intrinsic matrix to obtain the image shown in the image plane as illustrated in Eq. 2.21 and Figure 2.2.

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & \tau_x & 0 \\ 0 & f_y & \tau_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \Lambda \begin{bmatrix} \mathbf{R} & \vec{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \tag{2.21}$$

Given a point in 3D coordinates that lies on the calibration grid (x_W, y_W, z_W) and $z_W = 0$, its corresponding position in the image and the predetermined intrinsic parameters, Eq. 2.21 represents a homography between (x_I, y_I) and (x_W, y_W) which can be solved in the least squares method and refined by using non-linear optimisation. This method is widely used in the field of computer vision and it can achieve sub-millimetre accuracy; however, this also depends on the performance of the grid/corner detection algorithms and the distortion of the camera. The outliers in the corners detection and the error in the distortion parameters can cause a small error in the extrinsic parameters, which in turn creates a significantly different calibration result in a numerically sensitive problem such as the hand-eye problem. The influence of simulated noise to the calibration result is demonstrated later in Section 3.4.

2.3.2 Forward kinematics

A robotic system is a combination of joints and links from the base of the robot to the end-effector, in which joints can be revolute, prismatic or fixed. To localise the end-effector of a robot, frames are assigned at joints and links to create a chain of transformations from the base of the robot to the end-effector.

Forward kinematics is a method determining the robot's pose in subsequent frames relative to the frames defined on the robot. The most commonly used technique to solve the forward kinematics problem is called the Denavit-Hartenberg convention. The convention assigns four parameters (called "DH parameters") a_i, d_i, α_i and θ_i on the length and twist of each link and axes for each joint i as shown in Figure 2.3. Suppose that a frame i and a subsequent frame j are assigned

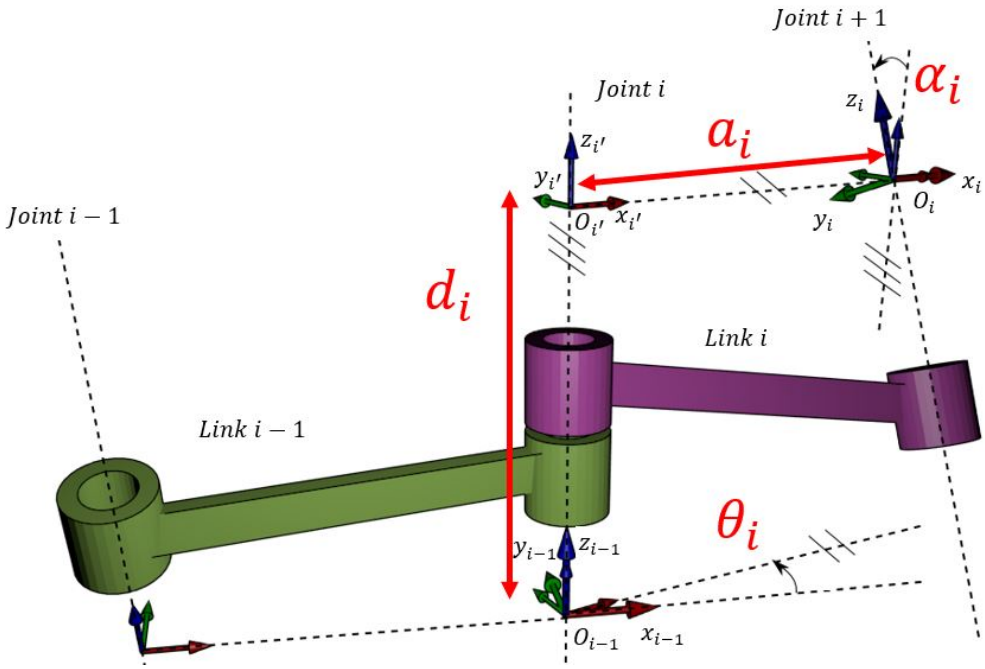


Figure 2.3: The schematic shows the definition of the parameter a_i, d_i, α_i and θ_i in links and joints and the convention on how to assign frames on to each joint [79].

in a robot. Using the described convention, the transformation between frame i and frame j can be written as,

$$^{i}\mathbf{T}_j = \mathrm{Rot}(z_i, \theta_j)\mathrm{Tran}(z_i, d_j)\mathrm{Tran}(x_j, a_j)\mathrm{Rot}(x_j, \alpha_j) \tag{2.22}$$

where the pure rotations ($\mathrm{Rot}(z, \theta_j)$ and $\mathrm{Rot}(x, \alpha_j)$) and translations ($\mathrm{Tran}(z, d_j)$ and $\mathrm{Tran}(x, a_j)$) are defined by,

$$\mathrm{Rot}(z, \theta_j) = \begin{bmatrix} \cos\theta_j & -\sin\theta_j & 0 & 0 \\ \sin\theta_j & \cos\theta_j & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.23}$$

$$\mathrm{Rot}(x, \alpha_j) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_j & -\sin\alpha_j & 0 \\ 0 & \sin\alpha_j & \cos\alpha_j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

$$\mathrm{Tran}(z, d_j) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

$$\mathrm{Tran}(x, a_j) = \begin{bmatrix} 1 & 0 & 0 & a_j \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.26}$$

Therefore, for a manipulator of n joints, the pose of the end-effector with respect to the base can be calculated as follows,

$$^{0}\mathbf{T}_n = ^{0}\mathbf{T}_1 \, ^{1}\mathbf{T}_2 \ldots ^{n-1}\mathbf{T}_n \tag{2.27}$$

According to the literature, two further conventions are used in robotics research: modified DH convention [79] and the product of exponential formula (POE) [80]. We are only interested in the modified DH convention in this research as the POE formula is not as commonly used in hand-eye calibration as either of the DH conventions. The modified convention formulates the transformation between frame i and frame j as follows,

$$^{i}\mathbf{T}_j = \mathrm{Rot}(x_i, \alpha_i)\mathrm{Tran}(x_i, a_i)\mathrm{Rot}(z_j, \theta_j)\mathrm{Tran}(z_j, d_j) \tag{2.28}$$

Similarly to the estimation of extrinsic parameters, forward kinematics has its own imperfection in defining the transformation. As the method relies heavily on the physical dimension of the robot provided by the robot manual, it does not account for geometric (imprecision in manufacturing) and non-geometric errors (backlash, elasticity, joint compliance). While such errors can be minimised using robot calibration methods [43, 55, 56, 65], they can still be propagated to the end-effector pose which creates a noticeable impact on the hand-eye calibration problem.

Methods	Solver
Solve rotation and translation separately	Linear least squares [41, 44, 81, 82] Quaternion methods [75, 83, 84] Dual-quaternion methods [45, 85]
Solve rotation and translation simultaneously	Dual-quaternion methods [73, 86] Non-linear methods [87–91]
Solve the problem using only one data stream	Structure from motion (SfM) [92–94] Gröbner basis [95]
Probabilistic method	Dirac-delta function [61, 62, 64, 96, 97]

Table 2.1: A summary of previous studies related to hand-eye calibration algorithms

2.3.3 Data selection

The lemmas and proofs in Tsai and Lenz's study indicate that high number of motions and a wide range of motion can suppress the effect of noise in the system [41, 44]. Therefore, it is always preferable to include all the captured motion in the calibration given that the motion has a wide range. However, no criteria exist regarding the convergence of optimisation in the calibration, i.e. there does not exist any proof where the convergence occurs in relation to the number of motions and range of motion.

Opposite to the effect of the range of motion, any pair of the transformations that creates a small relative transformation in rotation and translation can increase the effect of noise in the system as shown in the proofs in [44]. Therefore, data selection plays a vital role in the hand-eye problem. In order to achieve optimal calibration performance, the rotational movement in the relative transformations should be maximised, whilst minimising the distance between the optical centre of the camera and the calibration object [57].

However, planning paths such that the robot motion satisfies the criteria, whilst feasible in a general robotic application, is not so in a surgical context, as movement of the robot and camera movement is confined to a particular area.

2.3.4 Hand-eye calibration

There is an extensive literature on hand-eye calibration algorithms as the problem dates back to the 1980s. All of the algorithms developed in the literature derive their solutions from the classic hand-eye equation (Eq. 2.17). The existing hand-

eye calibration techniques in the literature review are categorised and summarised in Table 2.1 based on how they work. The most common approach is to formulate and solve the problem using the quaternion or dual quaternion domains and the most common calibration object is the chessboard calibration grid. Except for the SfM and Gröbner basis methods, the calibration procedure consists of acquiring several images of a planar checkerboard with different camera orientations, capturing the joint data and using the forward kinematics to compute the motion of the robot end-effector, and applying these data streams to the hand-eye calibration algorithm. In the next several sections, the related work will be explained in detail together with its drawbacks.

2.3.5 Decoupling the hand-eye solution

Eq. 2.17 can be decoupled into two equations as follows,

$$\mathbf{R}_A\mathbf{R}_X = \mathbf{R}_X\mathbf{R}_B \tag{2.29}$$

$$\mathbf{R}_A\vec{t}_X + \vec{t}_A = \mathbf{R}_X\vec{t}_B + \vec{t}_X \tag{2.30}$$

As shown in the equations, decoupling rotation components from Eq. 2.17 makes rotation estimation independent of the noise in the translation component which agrees with the screw representation of a rigid transformation [45]. The solvers in this section use the geometric relationship between the rotational axes of the camera and robot motion to determine the rotation component of the hand-eye matrix and then solves Eq. 2.30 for $\vec{t}_X$ using the least-squares method.

The most classic and commonly used method in the field of computer vision and robotics is the algorithm developed in [41, 44]. The extensive proofs in the paper state that the sum of rotation axes of **A** and **B**, the difference between the rotation axes of **A** and **B** and the rotation axis of the hand-eye transformation are orthogonal to each other. The relationship can be used to compute the axis of rotation

as follows,

$$\begin{bmatrix} [\vec{r}_{a1}]_\times + [\vec{r}_{b1}]_\times \\ [\vec{r}_{a2}]_\times + [\vec{r}_{b2}]_\times \\ \vdots \\ [\vec{r}_{aN}]_\times + [\vec{r}_{bN}]_\times \end{bmatrix} \vec{n}_X = \begin{bmatrix} \vec{r}_{a1} - \vec{r}_{b1} \\ \vec{r}_{a2} - \vec{r}_{b2} \\ \vdots \\ \vec{r}_{a2} - \vec{r}_{b2} \end{bmatrix} \tag{2.31}$$

where $\vec{r}_{ai}$ and $\vec{r}_{bi}$ are the unit rotation axes of $\mathbf{A}$ and $\mathbf{B}$, respectively and $[]_\times$ indicates a skew symmetric representation of a vector. The vector $\vec{n}_X$ is the rotation axis of $\mathbf{R}_X$ with a norm of $\frac{1}{2\cos\frac{\theta_X}{2}}$, i.e.

$$\vec{r}_X = 2\cos\frac{\theta_X}{2}\vec{n}_X \tag{2.32}$$

where θ_X is the angle of rotation of $\mathbf{X}$. Eq. 2.31 and 2.32 can be used to find the rotation component of the hand-eye matrix. Then, we can use a simple linear method to solve Eq. 2.30 for the translation component as follows,

$$\vec{t}_X = \begin{bmatrix} \mathbf{R}_{A1} - \mathbf{I}_3 \\ \mathbf{R}_{A2} - \mathbf{I}_3 \\ \vdots \\ \mathbf{R}_{AN} - \mathbf{I}_3 \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{R}_X\vec{t}_{B1} - \vec{t}_{A1} \\ \mathbf{R}_X\vec{t}_{B2} - \vec{t}_{A2} \\ \vdots \\ \mathbf{R}_X\vec{t}_{BN} - \vec{t}_{AN} \end{bmatrix} \tag{2.33}$$

where the superscript † represents the pseudoinverse operator. This method was published in 1989 and is still commonly used in many research fields because it provides good calibration accuracy. However, the method is highly dependent on the range of motion, which is one of its problems in a surgical environment. The solution in [81] works similarly to this method, but it incorporates an arctangent function instead of using the vector product, which yields the same result.

Park et al. also published a work on hand-eye calibration using Lie algebra [82] to determine the rotation component. The study makes use of Eq. 2.29 and incorporates Lie algebra to further derive the following relationship.

$$\vec{\omega}_A = \mathbf{R}_X\vec{\omega}_B \tag{2.34}$$

where $\vec{\omega}_A$ and $\vec{\omega}_B$ are the rotation axes with the norm of angles of rotation in the transformations **A** and **B**, respectively. Eq. 2.34 implies that the angles of rotation of the transformations **A** and **B** are the same size regardless of the rotation axes. With the application of Lie algebra [98] and the Cayley-Hamilton theorem (the polynomial expression of any algebraic function applied to a matrix), the optimal value of $\mathbf{R}_X$ can be expressed as,

$$\mathbf{R}_X = (\text{sqrtm}\,(\mathbf{M}^T\mathbf{M}))^{-1}\mathbf{M}^T \tag{2.35}$$

where $\mathbf{M} = \Sigma\vec{\omega}_B\vec{\omega}_A^T$ and the function sqrtm represents the square root of a matrix which again can be calculated using the Cayley-Hamilton theorem. The solved rotation matrix is later used to find the translation component in the same manner as in the aforementioned works (Eq. 2.33). However, this solution needs more than 50 samples of pairwise combination of **A** and **B** to achieve convergence.

It is well known that quaternion is sometimes a better representation of a 3×3 rotation matrix [99]. Therefore, apart from calibrating the matrix in the $SE(3)$ and Lie algebra $se(3)$ domains, Jack et al. also proposes using the algorithm to solve the hand-eye problem in the quaternion domain [83, 84]. The paper uses the quaternion representation to re-arrange Eq.2.29 into the least-squares problem.

$$\begin{bmatrix} a_0 - b_0 & -(\vec{a} - \vec{b})^T \\ \vec{a} - \vec{b} & [\vec{a} + \vec{b}]_\times + (a_0 - b_0)\mathbf{I}_3 \end{bmatrix} \mathbf{q} = 0 \tag{2.36}$$

Eq. 2.36 can be solved by using singular value decomposition (SVD), while the method for solving the translation component remains unchanged in the paper. This paper also provides an extensive theoretical study of the method, but does not compare the performance with the existing algorithm. A comparative study was later performed [45] in which the authors solved the problem using dual quater-

nions. The proposed method outperformed the quaternion-domain methods.

$$\begin{bmatrix} a_0' - b_0' & -(\vec{a}' - \vec{b}')^T \\ \vec{a}' - \vec{b}' & [\vec{a}' + \vec{b}']_\times + (a_0' - b_0')\mathbf{I}_3 \end{bmatrix} \mathbf{q} = \begin{bmatrix} -a_0 - b_0 & -(\vec{a} - \vec{b})^T \\ \vec{a} - \vec{b} & [\vec{a} + \vec{b}]_\times + (a_0 - b_0)\mathbf{I}_3 \end{bmatrix} \mathbf{q}' \tag{2.37}$$

The algorithm using dual quaternions proposed in [45] determines the rotation component the using Eq. 2.36, and uses a similar representation to solve Eq. 2.30 for the translation component. This method was later improved upon by non-linear optimisation with the epipolar constraint [85].

The advantage of solving the problem separately is that the rotation estimation does not suffer from the perturbation in the translation as a result of decoupling the rotation components from the translation. However, the optimisation also loses the constraints from the translation equation (Eq. 2.30) and may therefore yield an imperfect rotation estimation which is often the case in any calibration problem, which will, in turn, cause an error in translation estimation due to the formulation in Eq. 2.30.

2.3.6 Coupling the hand-eye solution

Apart from having a well-defined rotation component, transforming Eq. 2.17 to the dual quaternion domain also allows the problem to be solved simultaneously for the rotation and translation components [73,86]. Such formulation also assumes that the angles of rotation in $\mathbf{R}_A$ and $\mathbf{R}_B$ are very close due to the screw constraints and therefore the scalar components of the dual quaternion equation can be neglected. Denote $\vec{\tilde{\mathbf{a}}}$ and $\vec{\tilde{\mathbf{a}}}'$ as the real and dual components of the transformation $\mathbf{A}$ without the scalar components. The dual quaternion formulation can be re-written as,

$$\begin{bmatrix} \vec{\tilde{\mathbf{a}}} - \vec{\tilde{\mathbf{b}}} & [\vec{\tilde{\mathbf{a}}} + \vec{\tilde{\mathbf{b}}}]_\times & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ \vec{\tilde{\mathbf{a}}}' - \vec{\tilde{\mathbf{b}}}' & [\vec{\tilde{\mathbf{a}}}' + \vec{\tilde{\mathbf{b}}}']_\times & \vec{\tilde{\mathbf{a}}} - \vec{\tilde{\mathbf{b}}} & [\vec{\tilde{\mathbf{a}}} + \vec{\tilde{\mathbf{b}}}]_\times \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{q}' \end{bmatrix} = S(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) \begin{bmatrix} \mathbf{q} \\ \mathbf{q}' \end{bmatrix} = 0 \tag{2.38}$$

The matrix S can be stacked up to a $6n \times 8$ matrix, where n is the number of non-parallel rotation axis motions. By using singular value decomposition, the

solution of Eq 2.38 can be written by linear combination of two dual quaternions $\mathbf{u}, \mathbf{v} \in \mathbb{R}^8$ spanning the null space of the matrix S.

$$\begin{bmatrix} \mathbf{q} \\ \mathbf{q}' \end{bmatrix} = \alpha_1 \mathbf{u} + \alpha_2 \mathbf{v}, \qquad \alpha_1, \alpha_2 \in \mathbb{R} \tag{2.39}$$

To solve Eq. 2.39, we can use the property of dual quaternions, the unit rotation quaternion and the orthogonality conditions, to formulate the problem into a quadratic equation to solve for the real numbers α_1 and α_2. The solution gives a more accurate estimation of the translation component of the hand-eye matrix than the classic solution. However, it does not always give the correct solution. The roots of any quadratic equations are always either "two real roots", "one real root" or "two complex roots". Therefore, in the presence of noise, it cannot be guaranteed that a formulated quadratic equation will give correct roots or even the real ones. In addition, the scalar components of the dual quaternion $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ (defined in Eq. 2.8 and 2.12) are not always equal, since the angles of rotation in the two coordinate systems are not the same due to the noise in the system. Therefore, using this algorithm with noisy localisation of tools or sensors may not be suitable in a surgical application.

In any calibration problem, a non-linear optimisation is often applied at the end of the algorithms to refine the solution. Horaud and Dornaika developed an objective function to optimise the rotation and translation components at the same time.

$$f(\mathbf{q}, \vec{t}) = \phi(\mathbf{q}, \vec{t}) + \lambda(1 - \mathbf{q}^T \mathbf{q})^2 \tag{2.40}$$

where $\phi(\mathbf{q}, \vec{t})$ is simply Eq. 2.17 in the quaternion representation and λ is a large scalar value to enforce the normality of the unit quaternion. One downside of this formulation is that the function adds rotation and translation errors together without any scaling factor. The two units the metrics are representing are different. Therefore without an appropriate scaling factor, the equation is invalid. This issue is pointed out in [87], where the factors are estimated from the probability density function for both rotation and translation errors.

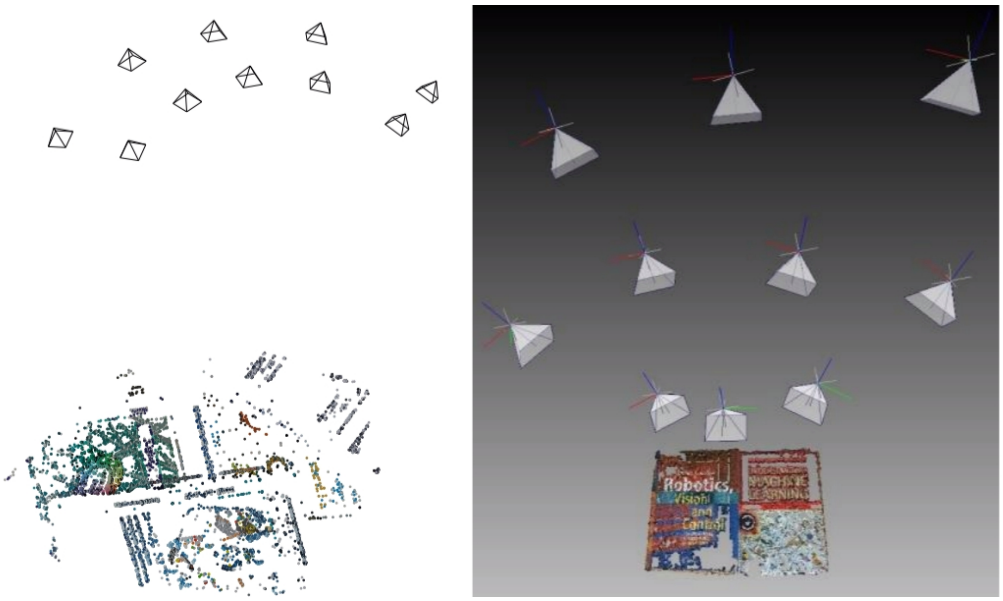
Bundle adjustment can also be applied in the hand-eye calibration problem. The method is presented in [88] and [85]. The hand-eye matrix is optimised by finding the matrix that minimises the re-projection error of the calibration grid corners in a stereo camera. The problem of using this function is that it needs either the estimation of the transformation between a robot and a calibration grid or a prediction of the camera pose using the estimated hand-eye matrix. To estimate the former parameter, the matrix can be obtained by completing the chain of transformations which can be problematic since different pairs of robot and camera poses typically produce different matrices due to noise, in which case there is no prior knowledge as to which pair is more accurate. On the other hand, camera pose prediction is more commonly used in research on the hand-eye problem, but this involves using a pair of robot poses which are usually less accurate than those of a camera. The uncertainty in the robot poses will propagate to the predicted camera pose and create a bias in the optimisation and validation processes.

A technique using an invariant point was proposed in 2016 [89] in which the hand-eye matrix is refined using a single invariant point. In the paper, the point is defined at the centre of a crosshair and the algorithm estimates the location of the point and the hand-eye matrix at the same time. The final objective function works similarly to the bundle adjustment.

A similar method to a single invariant point method is proposed in [100]. The authors use bundle adjustment to refine the initialised hand-eye solution from the Kronecker product [101]. RANSAC is used in the paper to reject outlier measurements. Then, the modified version of the bundle adjustment is optimised based on $3m+6$ variables (6 from the hand-eye matrix and $3m$ from the used 3D points in the function). Such objective function is

$$\min_{\mathbf{X},\,{}^{\text{grid}}\mathbf{T}_{\text{cam}}} \sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}||x_{ij} - P_j(\mathbf{X}\mathbf{B}_j^{-1}\,{}^{\text{grid}}\mathbf{T}_{\text{cam, i}})||^2 \tag{2.41}$$

where v_{ij} is 1 if the ith point can be observed from the jth view, otherwise it is 0. x_{ij} is the ith point from the total of m points from the view j^{th} from the total of n views.



(a) (b)

Figure 2.4: The examples of camera poses and the reconstructed scene using Structure-from-motion to calibrate the hand-eye matrix [93, 94].

P_j is the projection function of the camera from the view j and $\mathbf{X}$ is the hand-eye matrix. While the refinement of the hand-eye matrix increases calibration accuracy, using the Kronecker product solution as the initialised solution may not give a good estimation of the solution as it needs to be projected back to $SO(3)$.

2.3.7 Hand-eye calibration without hand or eye orientation

There are special cases in the hand-eye solution that work on the same formulation (Eq. 2.17), but use only one data stream; e.g., either only hand poses or camera poses. The work in [95] solves the hand-eye problem when the positioning system of the robot arm is not properly calibrated. The method uses Gröbner basis [102] to solve Eq. 2.30 for the initialisation and non-linear optimisation to subsequently refine the solution. Although the algorithm yields good accuracy, with a precision of 10 millimetres, the existing algorithms using both robot and camera poses can yield a better calibration accuracy.

On the other hand, there are previous works that do not require extrinsic parameters in the calibration pipeline. SfM is an imaging technique that estimates 3D structures or scenes from several camera poses. Unlike the previous system, work

in [92–94] proposes solving the hand-eye problem without using "eye" information, but instead sets of correspondences are used in the estimation. Using the same principle, simultaneous localisation and mapping (SLAM) can also be applied [103] to solve the hand-eye problem. However, both SLAM and SfM methods do not work well with laparoscopic video as a dynamic scene of an operative site, including non-rigid tissues, occlusions and fast camera motions can make localisation very challenging [104].

The literature shows that the hand-eye calibration algorithms with one data stream can give good calibration accuracy, despite the expensive computational cost. Furthermore, it may work better to have a priori knowledge of the unused data streams since the currently available camera calibration and robot positioning systems can be refined to sub-millimetre accuracy.

2.3.8 Probabilistic methods

Some of the most recent studies in hand-eye calibration describe the probabilistic methods [61, 62, 64, 96, 97]. The proposed methods are designed for an online hand-eye calibration in which two data streams are not fully synchronised, i.e. the starting times of the two data streams are not the same due to the hardware setup which invalidates Eq. 2.19 and 2.20 as the time values τ and τ' in $\mathbf{A}$ and $\mathbf{B}$ are not synchronised.

The algorithm uses the Dirac delta function on $SE(3)$ to define the distribution of the transformations and convert Eq. 2.17 into the linear convolution of Dirac delta functions. This formulation yields two key equations.

$$\mathbf{M}_A\mathbf{X} = \mathbf{X}\mathbf{M}_B \tag{2.42}$$

$$\mathrm{Ad}(\mathbf{X}^{-1})\Sigma_A\mathrm{Ad}^T(\mathbf{X}^{-1}) = \Sigma_B \tag{2.43}$$

where $\mathbf{M}_A$ and $\mathbf{M}_B$ are the means of every possible combination of $\mathbf{A}$ and $\mathbf{B}$, Σ is

the covariance of the transformation matrix and the $Ad()$ operator is defined as

$$\mathrm{Ad}(\mathbf{X}) = \begin{bmatrix} \mathbf{R} & 0_{3\times3} \\ [\vec{t}]_\times \mathbf{R} & \mathbf{R} \end{bmatrix} \tag{2.44}$$

The algorithm requires a mean and a covariance of two sets of transformations ${}^{\mathrm{cam}}\mathbf{T}_{\mathrm{grid}}$ and ${}^{\mathrm{robot}}\mathbf{T}_{\mathrm{base}}$ as input and it then solves Eq. 2.42 and 2.43 using the least-squares method for the hand-eye matrix $\mathbf{X}$. [96] also proposes several methods to compute the average of the transformations. The solutions have been tested with scrambled data and time-delayed data streams and the results show that the algorithm can overcome the synchronicity issue in the hand-eye problem. However, this method does not solve the problem of different sampling rates. For example, when two large pools of $\{\mathbf{A}\}$ and $\{\mathbf{B}\}$ are presented in random orders, this algorithm requires at least one corresponding in $\{\mathbf{A}_i\}$ for a $\mathbf{B}_j$ such that it satisfies Eq. 2.17. This is not always the case when there are different sampling rates from different sensors.

In [61], the authors also propose a solution to synchronise the hand-eye data streams using the cross-correlation technique in the frequency domain. This approach only uses the rotation component of the hand-eye problem to find the delay; the translation component remains unused.

The probabilistic approaches in the literature may be suitable when we have two data streams with different starting times, but the formulation can be invalidated as some of the poses in one data stream do not have corresponding poses in the other. We will show in the chapter 6 that the probabilistic approach fails when there are different sampling rate in the data streams.

2.4 Discussion

The hand-eye problem arises when two coordinate systems are working together in one system, typically a robot arm with a camera rigidly mounted on the end-effector. It aims at determining the rigid transformation that links the two systems. Although this is a classic problem and many solutions have already been

proposed, they nevertheless have flaws and are not applicable to an RMIS setup, which requires a high calibration accuracy.

Starting from the data fed into the calibration, every transformation **A** and **B** has to be fully synchronised and yields a valid set of hand-eye equations: i.e. a solution exists. Since sets of transformations $\{\mathbf{A}\}$ and $\{\mathbf{B}\}$ are typically collected from different systems, which have different activation times and sampling rates, the temporal misalignment causes asynchronicity in the data streams and invalidates Eq. 2.17. Without synchronised data streams, the screw constraints on both rotation and translation components are not satisfied and the system of equations cannot yield the correct solution. Therefore, it is necessary to pre-process the data so that all the transformations in the data streams conform to the screw constraints.

The data used in the calibration is evaluated from two main steps: camera calibration is used to determine the camera pose (extrinsic parameters) and the robot pose is calculated using forward kinematics. The estimation of the extrinsic parameters is usually more accurate than the robot pose as bundle adjustment can be refined to sub-pixel accuracy. This shows that in most systems, calibration error is caused by the error in the robot pose, which is propagated from the kinematic chain to the estimated hand-eye matrix. In the context of RMIS, calibration accuracy is important as any discrepancy in the localisation could cause a misalignment in the applications. Hence, the developed hand-eye algorithm must be able to deal with the inaccuracy in the relative transformations from both the camera and robot kinematic chain.

From the surgeons' perspective, the calibration procedure is complicated as they are not trained to either acquire the data or prepare the setup for the calibration. Therefore, using the conventional hand-eye calibration methods may not be suitable. The conventional method requires the use of a calibration target that needs to be sterilised, and the calibration needs to be updated every time the setup of the robot is changed. Consequently, the operation workflow can be severely disrupted by the calibration process, which is not desirable. The calibration workflow must be simple and adapt to changes in the surgical environment in order to appropriately

update the hand-eye matrix without disrupting the overall operation.

Another problem relates to the formulation of the hand-eye problem itself. The original hand-eye equation as shown in the literature, relies on a high number of motions, and a wide motion range to suppress the calibration error. While many motions can easily be acquired over the course of a procedure, collecting motion with a sufficiently wide range for the calibration can be challenging. Due to the setup in keyhole surgery, camera motion is confined to the area around the RCM which only allows very small rotational and translational movement. This presents a problem in calibrating the hand-eye matrix as R_A and R_B are close to identity which makes the rotation component of the hand-eye equation unsolvable. This leaves only Eq. 2.30 to be solved which is an under-constrained problem. Therefore, the ideal algorithm should be able to make use of a small motion to calibrate the hand-eye matrix.

The problems in solving the hand-eye problem involve the synchronicity of the data, noise in the camera and the robot kinematic chain, the calibration workflow, and finally the motion range problem. The hand-eye algorithm, in the context of RMIS, should be able to overcome these challenges, which are not yet overcome by the currently available hand-eye calibration algorithms.

Chapter 3

Adjoint transformation formulation for hand-eye calibration

As described in Chapter 2, the hand-eye problem is formulated as an equation of homogeneous matrices (Eq. 2.17). Throughout the years, many closed-form hand-eye solutions have been proposed using different parameterisations e.g. the special Euclidean group $SE(3)$ [41, 44, 87], its Lie algebra group $se(3)$ [67, 82], quaternions [75, 83, 84], and dual quaternions [45, 73, 85]. According to the literature, among all parameterisations, Lie algebra and dual quaternions outperform the others. These two representations both describe a rigid motion as a screw motion.

Screw motion theory is based on the fact that an arbitrary rigid body transformation can be represented by a rotation about a certain axis and a translation along the same axis. The theory is frequently used in the visual servoing domain (vision-based control) where the robot pose is changed according to the quantities measured in the imaged frame [105] e.g. a coordinate point in an image, or an orientation of a line in an image. Using screw motion to represent rotation components prevents ill-posed cases when the composite transformation has parallel rotation axes, which renders the classic solution unsolvable (see Eq. 2.31) and is usually the case in RMIS applications; the axes of the robot and camera coordinate frames are parallel to each other and have similar rotation axes (see Figure 3.1(b)).

Apart from parameterisation, the use of a stereo camera can also greatly improve calibration accuracy, since each measurement creates additional geometric

constraints. Most hand-eye calibration algorithms are derived for an application with a monocular camera, but the formulation can be directly applied to a stereo camera as well by considering the motion of the two cameras as two independent monocular camera motions [87]. Using a stereo camera also creates an additional constraint that can be added to the formulation. For example, the cost function derived in [85, 106] uses the stereo information to posteriorly refines the hand-eye solution by minimising the epipolar error.

This chapter introduces a novel method for formulating the hand-eye problem that incorporates motions from a stereo camera and the adjoint transformation which uses the new constraints derived from screw motion theory. The presented algorithm uses stereo transformation and the property of the screw motion to calibrate the matrix. The method is experimentally validated using datasets from both simulation and robots, the da Vinci Surgical Robot and a KUKA LBR IIWA R800.

3.1 Hand-eye calibration with the adjoint transformation

The proposed algorithm uses Lie algebra and dual quaternions to formulate the problem. This formulation allows the decoupling of the rotation and translation parts such that the parallel axes do not create ill-posed cases and that it can be solved using alternating optimisation. The algorithm then jointly solves the equation for the rotation component [45] and uses the adjoint transformation to estimate the hand-eye matrix. Similarly to most state-of-the-art approaches, non-linear optimisation using Levenberg-Marquadt is applied at the end of the algorithm to refine the hand-eye matrix [88]. This calibration pipeline is similar to the camera calibration algorithm which iteratively solves for the extrinsic and intrinsic parameters in alternating steps, followed by a local optimisation refinement step [42]. The alternating optimisation algorithm is chosen here because it is usually faster than the numerical solution and can avoid local minima [107].

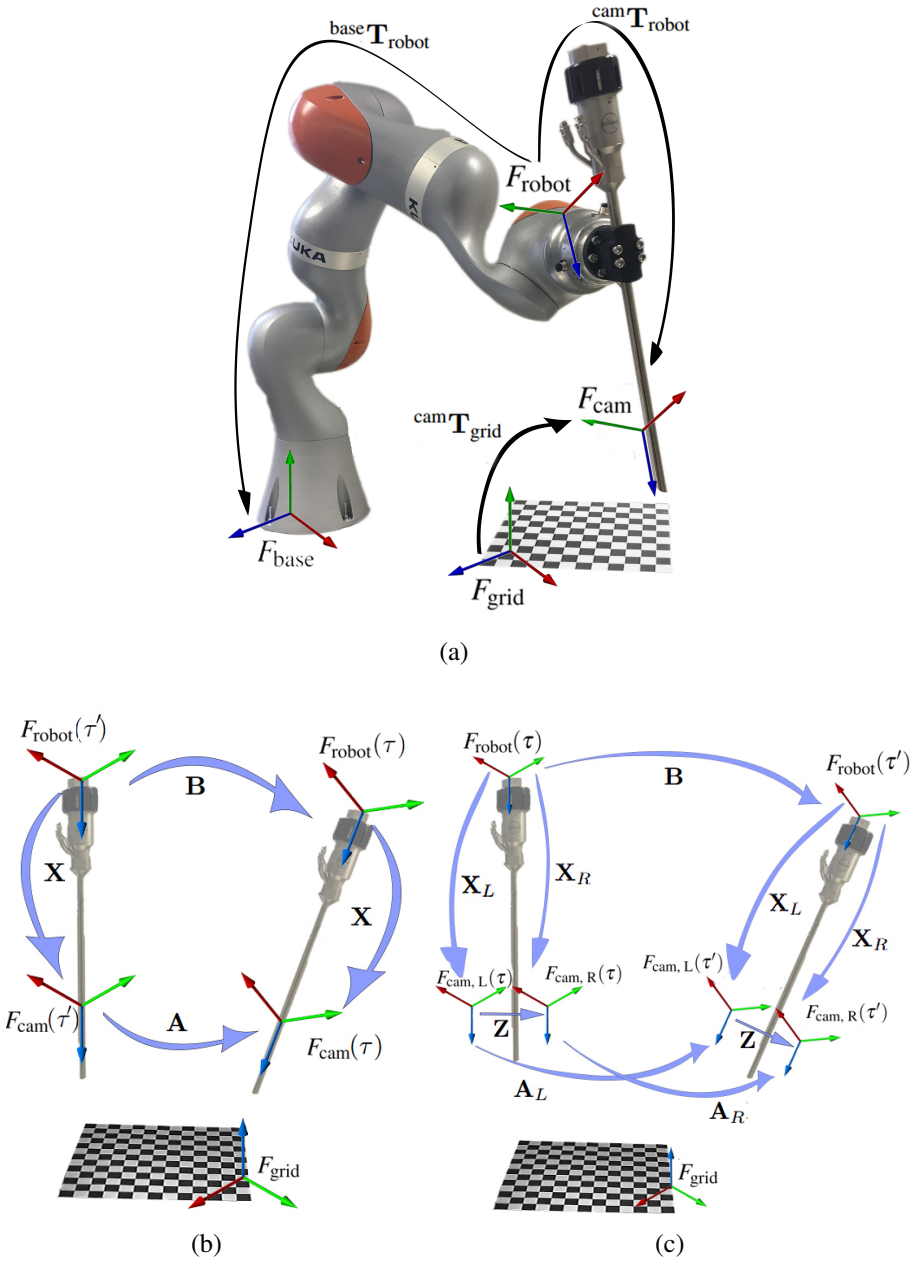
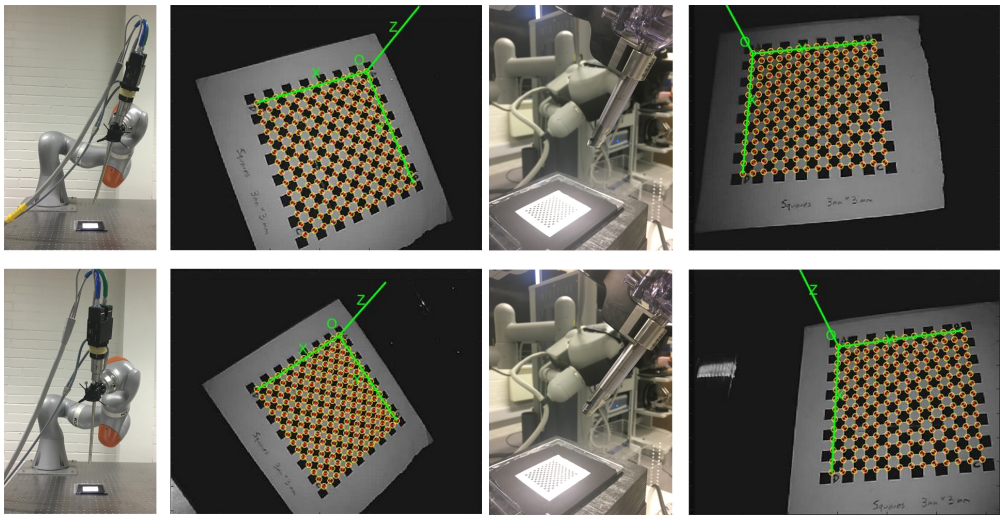


Figure 3.1: (a) Experimental setup for the classic hand-eye problem. The camera is attached to the end-effector of the robot, in this case, the flange of a KUKA arm. Hand-eye calibration is the method used to determine the missing transformation ${}^{\text{cam}}\mathbf{T}_{\text{robot}}$ which defines a pose of robot's frame with respect to the camera's pose. (b) The schematic showing the example of relative transformations for the robot frame and camera frame as mathematically represented by Eq. 2.19 and 2.20. (c) The schematic for the stereoscopic formulation of the hand-eye problem as mathematically represented by Eq. 3.13 - 3.18. $\mathbf{A}_{LR}$ is the transformation between $F_{\text{cam, L}}(\tau)$ and $F_{\text{cam, R}}(\tau')$, while $\mathbf{A}_{RL}$ is the transformation between $F_{\text{cam, R}}(\tau)$ and $F_{\text{cam, L}}(\tau')$



(a) (b)

Figure 3.2: (a) Example images of the KUKA's pose and its corresponding image side by side. KUKA is moved to several positions to collect images of the calibration grid from several points of view, while the pose of the KUKA is acquired using its API. (b) Example images of da Vinci's pose and its corresponding image side by side.

To solve the problem, Eq. 2.17 is decoupled into two parts as follows,

$$\mathbf{R}_A\mathbf{R}_X = \mathbf{R}_X\mathbf{R}_B \tag{3.1}$$

$$\mathbf{R}_A\vec{t}_X + \vec{t}_A = \mathbf{R}_X\vec{t}_B + \vec{t}_X \tag{3.2}$$

where $\mathbf{R}_X$ and $\vec{t}_X$ are the rotation component and the translation component of the hand-eye transformation, respectively. Then, the quaternion representation is used to rewrite Eq. 3.1. Let $\mathbf{a}$, $\mathbf{q}$ and $\mathbf{b}$ be quaternions that represent the rotation components of the transformations $\mathbf{A}, \mathbf{X}$ and $\mathbf{B}$. Thus, we have the following equation,

$$\mathbf{a}.\mathbf{q} = \mathbf{q}.\mathbf{b} \tag{3.3}$$

The quaternion multiplication can be written in a linear equation form and therefore, Eq. 3.3 can be rearranged into a matrix form as shown in Eq. 2.36. The classical solution to this equation for determining the rotation component uses singular value decomposition [45].

In order to find the translation component, an additional constraint that uses an adjoint transformation is used to formulate the equation. Post-multiplying Eq. 2.17 by the hand-eye matrix $\mathbf{X}^{-1}$ yields $\mathbf{A} = \mathbf{X}\mathbf{B}\mathbf{X}^{-1}$. Let $\mathfrak{a}$ and $\mathfrak{b}$ be the Lie algebra representations for the rigid transformations $\mathbf{A}$ and $\mathbf{B}$, respectively. Then by using the relationship between these two matrices provided in Chapter 2, we can convert the hand-eye equation into the following,

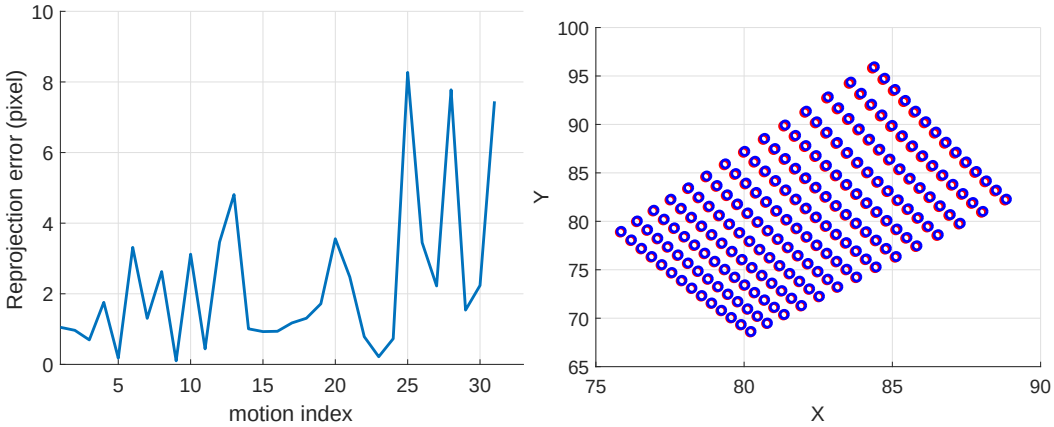
$$\operatorname{expm}(\mathfrak{a}) = \mathbf{X}\operatorname{expm}(\mathfrak{b})\mathbf{X}^{-1} \tag{3.4}$$

Using the property of the matrix exponential function, $\mathbf{X}\operatorname{expm}(\mathfrak{b})\mathbf{X}^{-1}$ can be converted to $\operatorname{expm}(\mathbf{X}\mathfrak{b}\mathbf{X}^{-1})$ since a rigid transformation is always invertible [108].

From here, to solve the equation in the $se(3)$ domain, we equate the exponent terms on both sides of Eq. 3.4. However, this step does not directly follow from Eq. 3.4 as the matrix exponential function is a surjective function: i.e. any rigid transformation $\mathbf{T}$ can have more than one correspondence in the Lie algebra domain. Therefore, we need to further prove that the exponential mapping in the context of the hand-eye problem is uniquely defined for every possible transformation such that we can safely assume the equality of the exponent terms, $\mathfrak{a} = \mathbf{X}\mathfrak{b}\mathbf{X}^{-1}$.

Rigid transformations contain three DoF in rotation and another three in translation. According to Eq. 2.13-2.16, the mapping between the rotation components is not dependent on the mapping of the translation components [108]. Therefore, the uniqueness of the rotation mapping can be proven separately from the translation case.

According to [82], the exponent term of the rotation component is uniquely defined when the trace of a rotation matrix is not equal to -1, i.e. when the angle of rotation is not $\pm\pi$. Therefore, if we avoid 180 degrees of rotations during the calibration procedure, the uniqueness of this rotation mapping can be safely assumed. Note that it is already difficult for these particular motions to appear in practice, since the calibration target must be retained in the field of view throughout the procedure, and this constrains the camera and the robot motions. For the translation mapping, according to Eq. 2.14, the translation mapping is dependent on the



(a) (b)

Figure 3.3: (a) Re-projection error when adding a small Gaussian noise of the standard deviation 0.025 mms into the system. (b) Examples of how the grids are projected back to the images after adding noise; Red dots represent ground truth and blue dots represent the re-projected grid after adding noise.

rotation and since we prove the injectivity of this mapping, we can also infer the injectivity of the translation mapping. Hence,

$$\mathfrak{a} = \mathbf{X}\mathfrak{b}\mathbf{X}^{-1} \tag{3.5}$$

$$\begin{bmatrix} [\vec{\omega}_A]_\times & \vec{v}_A \\ \vec{0}^T & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_X & \vec{t}_X \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} [\vec{\omega}_B]_\times & \vec{v}_B \\ \vec{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_X^T & -\mathbf{R}_X^T\vec{t}_X \\ \vec{0}^T & 1 \end{bmatrix} \tag{3.6}$$

$$\begin{bmatrix} \vec{\omega}_A \\ \vec{v}_A \end{bmatrix} = \begin{bmatrix} \mathbf{R}_X & \mathbf{0}_{3\times 3} \\ [\vec{t}_X]_\times \mathbf{R}_X & \mathbf{R}_X \end{bmatrix} \begin{bmatrix} \vec{\omega}_B \\ \vec{v}_B \end{bmatrix} \tag{3.7}$$

where $\vec{\omega}_A, \vec{\omega}_B, \vec{v}_A$ and $\vec{v}_B$ are the Lie Algebra components of the transformations. The rotation component in Eq. 3.7 is essentially the orthogonal Procrustes problem [109] and the condition has already been satisfied by the solution of Eq. 2.36. The translation component of Eq. 3.7 indicates the relation between $\vec{v}_A$ and $\vec{v}_B$ and this is used in the estimation of the translation component for the calibration.

$$\vec{v}_A = [\vec{t}_X]_\times \mathbf{R}_X \vec{\omega}_B + \mathbf{R}_X \vec{v}_B \tag{3.8}$$

According to the specification in the respective documentations of the robots, the KUKA has a repeatability of 0.1 mms and the da Vinci positioning system is accurate within only a cube of 125 cm^3 in volume, while the laparoscopic camera calibration can be refined to sub-pixel accuracy [50, 110]. Although repeatability does not directly imply accuracy, it indicates an offset in the positioning system, if there is an error. However, this error in the positioning system creates more than sub-pixel errors when the grid is projected onto images as shown in Figure 3.3(a). Therefore, in our applications, the relative camera motion **A** is more accurate than the robot motion **B**. Hence, from Eq. 3.8, we can avoid using the rotation component of **B** by substituting $\mathbf{R}_X\vec{\omega}_B$ with $\vec{\omega}_A$.

$$\vec{v}_A = [\vec{t}_X]_\times \vec{\omega}_A + \mathbf{R}_X \vec{v}_B \tag{3.9}$$

Then Eq. 3.9 is transformed into the quaternion form so that we can jointly solve the equation with Eq. 2.36.

$$\begin{bmatrix} 0 & -(\vec{v}_A - [\vec{t}_X]\vec{\omega}_A - \vec{v}_B)^T \\ \vec{v}_A - [\vec{t}_X]\vec{\omega}_A - \vec{v}_B & [\vec{v}_A - [\vec{t}_X]\vec{\omega}_A + \vec{v}_B]_\times \end{bmatrix} \mathbf{x} = \vec{0} \tag{3.10}$$

The solution to Eq. 2.36 and 3.10 is a refined solution of the rotation component. For the translation component, we use the last three rows of Eq. 3.7. Since $[\vec{t}]_\times \vec{\omega}_A$ is equivalent to $-[\vec{\omega}_A]_\times \vec{t}$, we can arrive at the following equation:

$$[\vec{\omega}_A]_\times \vec{t}_X = \mathbf{R}_X \vec{v}_B - \vec{v}_A \tag{3.11}$$

Eq. 3.11 consists of three equations. By collecting N motions, there will be $3 \times N$ different equations and the translation component of the hand-eye matrix must satisfy all of the equations as well as Eq. 3.10. Therefore, unlike most decoupled hand-eye solutions which only use Eq. 2.29 to solve for the rotation component, the constraint from the translation equation (Eq. 3.11) can increase the accuracy of the estimation of the rotation component.

The main algorithm operates on this formulation. It first determines the rota-

Algorithm 1 Hand-eye calibration for a monocular camera

```
procedure HANDEYE(^cam T_grid, ^robot T_base)
    A, B ← relative transformations from Eq. 2.19 and 2.20
    R_0, t_0 ← initialise the identity matrix
    repeat
        x ← solve Eq. 2.36, 3.10 using t_0
        R_refine ← convert quaternion x to rotation matrix
        R_0 ← R_refine
        t_refine ← solve Eq. 3.11 using R_0
        t_0 ← t_refine
    until the solution converges
    X ← R_0, t_0
    X ← refine X_0 by minimising residuals in Eq. 3.12
    return X                                  ▷ Hand to eye transformation
end procedure
```

tion component using the solution from Eq. 2.36 and uses the solution to solve Eq. 3.11 for the translation component. Then, the algorithm will go back to refine the rotation component using the solution of Eq. 3.11 and continues this alternating optimisation until the hand-eye solution converges. The convergence criteria of the algorithm is a less than 10^{-4} change in both rotation and translation components for more than 20 iterations, although as with all iterative methods this criterion can be adjusted. According to the results displayed in Figure 3.7, the solution of ATA converges to the same result regardless of hand-eye initialisation, which is an important and practically valuable property of our method.

At the end of the algorithm, the hand-eye transformation is refined further by using the Levenberg-Marquadt algorithm to minimise the residue in the hand-eye equation. The algorithm finds the optimal solution $\hat{\mathbf{q}}_X$ in the dual quaternion domain that has the corresponding rigid transformation $\mathbf{X}$ for the hand-eye equation. For all possible motions N, our objective function $\Phi(\hat{\mathbf{q}}_X)$ can be written in Eq. 3.12 [111].

$$\Phi(\hat{\mathbf{q}}_X) = \sum_{i=1}^{N} ||\hat{\mathbf{a}}_i \otimes \hat{\mathbf{q}}_X - \hat{\mathbf{q}}_X \otimes \hat{\mathbf{b}}_i||^2 \tag{3.12}$$

To summarise, the hand-eye solution initialisation incorporating the adjoint transformation algorithm using a monocular vision is described in Algorithm 1.

3.2 Stereoscopic formulation of hand-eye calibration

The conventional hand-eye constraint from Eq. 2.17 is not limited to monocular vision but can also be extended to stereo vision, which increases calibration accuracy by introducing additional relative motion constraints. The new formulation using a stereo vision can be created using these additional constraints.

In the stereo setup (Figure 3.1(c)), the two cameras are rigidly attached and they are rigidly maintained at the robot's end-effector. Therefore, each relative robot motion **B** creates two sets of camera motions in two different frames such that Eq. 2.17 can be formed for the two cameras and represented as follows,

$$\mathbf{A}_L\mathbf{X}_L = \mathbf{X}_L\mathbf{B} \tag{3.13}$$

$$\mathbf{A}_R\mathbf{X}_R = \mathbf{X}_R\mathbf{B} \tag{3.14}$$

In order to use the following formulation, the stereo camera has to be pre-calibrated, i.e. the rigid transformation linking the coordinate systems of the two camera is readily determined from stereo camera calibration. Let **Z** be the transformation linking the left and the right cameras together. Hence, the relationship between the solutions in Eq. 3.13 and 3.14 can be written as following,

$$\mathbf{Z}\mathbf{X}_L = \mathbf{X}_R \tag{3.15}$$

Substituting Eq. 3.15 into Eq. 3.13 and 3.14 creates three additional equations which are related by the same hand-eye solution as shown in Eq. 3.16-3.18.

$$\mathbf{Z}^{-1}\mathbf{A}_R\mathbf{Z}\mathbf{X}_L = \mathbf{X}_L\mathbf{B} \tag{3.16}$$

$$\mathbf{A}_{LR}\mathbf{Z}\mathbf{X}_L = \mathbf{X}_L\mathbf{B} \tag{3.17}$$

$$\mathbf{Z}^{-1}\mathbf{A}_{RL}\mathbf{X}_L = \mathbf{X}_L\mathbf{B} \tag{3.18}$$

where $\mathbf{A}_{LR} = {}^{\text{cam,L}}\mathbf{T}_{\text{grid}}(\tau)({}^{\text{cam,R}}\mathbf{T}_{\text{grid}}(\tau'))^{-1}$ and $\mathbf{A}_{RL} = {}^{\text{cam,R}}\mathbf{T}_{\text{grid}}(\tau)({}^{\text{cam,L}}\mathbf{T}_{\text{grid}}(\tau'))^{-1}$. The schematic of these relationships is shown in Figure 3.1(c). By simultaneously

Algorithm 2 Hand-eye calibration for a stereo camera

procedure HANDEYE($^{\text{cam,L}}\mathbf{T}_{\text{grid}}, {}^{\text{cam,R}}\mathbf{T}_{\text{grid}}, {}^{\text{robot}}\mathbf{T}_{\text{base}}, \mathbf{Z}$)
  $\mathbf{A}, \mathbf{B} \leftarrow$ relative transformation Eq. 3.13, 3.14, 3.16-3.18
  $\mathbf{R}_{L,0}, \vec{t}_{L,0} \leftarrow$ initialise identity matrix
  repeat
    $\mathbf{x} \leftarrow$ solve Eq. 2.36, 3.10 using $\vec{t}_0$
    $\mathbf{R}_{L,\text{refine}} \leftarrow$ convert quaternion $\mathbf{x}$ to rotation matrix
    $\mathbf{R}_{L,0} \leftarrow \mathbf{R}_{L,\text{refine}}$
    $\vec{t}_{L,\text{refine}} \leftarrow$ solve Eq. 3.11 using $\mathbf{R}_{L,0}$
    $\vec{t}_{L,0} \leftarrow \vec{t}_{L,\text{refine}}$
  until the solution converges
  $\mathbf{X}_{L,0} \leftarrow \mathbf{R}_{L,0}, \vec{t}_{L,0}$
  $\mathbf{X}_L \leftarrow$ refine $\mathbf{X}_{L,0}$ by minimising residuals in Eq. 3.12
  $\mathbf{X}_R \leftarrow$ solve Eq. 3.15 using $\mathbf{X}_L, \mathbf{Z}$
  return $\mathbf{X}_L$ and $\mathbf{X}_R$ ▷ Hand to eye transformation
end procedure

solving Eq. 3.13 and Eq. 3.16-3.18, it is shown later in Section 3.4 that we can obtain a more accurate hand-eye solution on the left-side of the camera regardless of the selection of the algorithms.

Similarly to the case of a monocular camera, the solution solved here can also be further refined by the Levenberg-Marquardt algorithm to find the minima for the hand-eye equation as shown in 3.12. The only difference is that we have the additional motions from the stereo information which remains fixed. To summarise, the initialisation using the stereoscopic formulation is described in Algorithm 2.

3.3 Experimental procedure

The algorithm was tested with both synthetic and real data. To generate synthetic data, we created a closed-loop sequence of rigid transformations, two of which were constant transformations: the transformation linking the base and the grid coordinate system ${}^{\text{base}}\mathbf{T}_{\text{grid}}$ and the ground truth for the hand-eye transformation ${}^{\text{cam}}\mathbf{T}_{\text{robot}}$.

The data generated in the experiment was in the form of a $6 \times N$ matrix where N was the number of camera poses used in the calibration. Each row represented a Lie algebra representation of a transformation where the first three elements stood

for the translation component and the last three elements were the rotation component (Rodrigues' representation). Using this representation, random robot poses ${}^{\text{base}}\mathbf{T}_{\text{robot},i}$ were simulated, where i indicated the index of the pose. The criteria for data generation depended on the simulated parameters of interest in each experiment. The transformation from the camera coordinate system to the grid coordinate system ${}^{\text{cam},i}\mathbf{T}_{\text{grid}}$ could then be simulated by completing the loop of rigid transformations as shown in Eq. 3.19.

$$ {}^{\text{cam},i}\mathbf{T}_{\text{grid}} = {}^{\text{cam}}\mathbf{T}_{\text{robot}}\left({}^{\text{base}}\mathbf{T}_{\text{robot},i}\right)^{-1}\,{}^{\text{base}}\mathbf{T}_{\text{grid}} \tag{3.19} $$

The data generated this way are noise-free and always satisfies Eq. 2.17 without any errors. From here, we applied Gaussian noise to the transformations ${}^{\text{cam}}\mathbf{T}_{\text{grid}}$ and ${}^{\text{base}}\mathbf{T}_{\text{robot}}$ to simulate real-world noise characteristics before they were fed into the hand-eye calibration functions. Eq. 3.20 shows how the noise is added to an arbitrary transformation $\mathbf{T}$,

$$ \mathbf{T}_{\text{corrupted}} = \mathbf{T}\begin{bmatrix} \text{rodrigues}(\sigma_r\vec{v}_r) & \sigma_t\vec{v}_t \\ \vec{0}_{1\times 3} & 1 \end{bmatrix} \tag{3.20} $$

where σ_r, σ_t are the noise intensity in the rotation and the translation components, respectively and $\vec{v}_r, \vec{v}_t$ are 3×1 a rotation vector and a translation vector that are generated randomly by the built-in Gaussian noise function in MATLAB. Thus, the value σ_r and σ_t act as the standard deviations of the generated noise intensity. Although Gaussian noise may not be an ideal noise characteristic for the simulation of robotic systems in general, it resembles the error from the camera parameters (Figure 3.4).

The tested algorithm then estimated ${}^{\text{cam}}\mathbf{T}_{\text{robot}}$ using the noisy data and we compared the estimated transformation with the ground truth. This process was run 1000 times for each simulation parameter.

To evaluate the accuracy, we simply computed the residue in the rotation and translation estimations separately. The translation error was computed from the

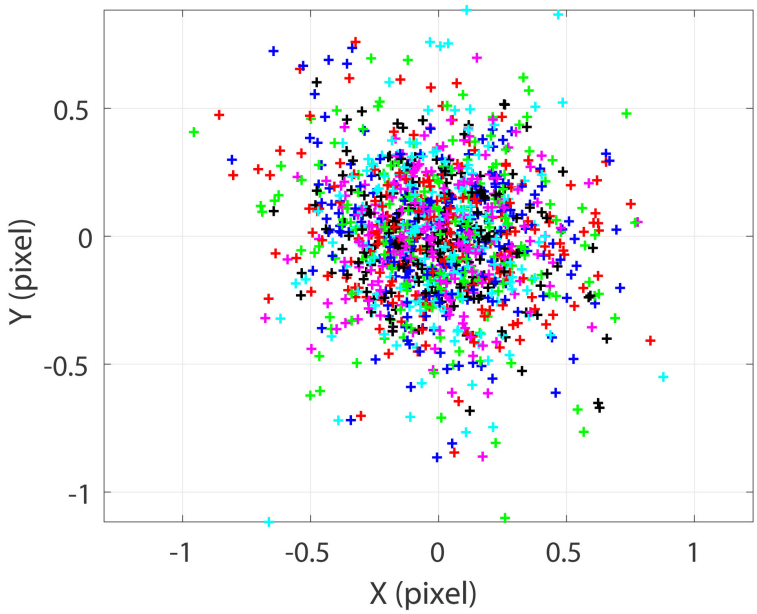


Figure 3.4: Re-projection error from using the estimated intrinsic and extrinsic parameters from the calibration toolbox. The error resembles the shape of a 2-D Gaussian noise.

norm of the difference between the ground truth and the estimated value. The error in the rotation component was calculated from the magnitude of the residual rotation between the ground truth and the estimation using Rodrigues' formula. The equations for computing errors in translation and rotation are shown in Eq. 3.21 and Eq. 3.22-3.23, respectively.

$$E_{\text{translation}} = ||\vec{t}_{\text{est}} - \vec{t}_{\text{gt}}|| \tag{3.21}$$

$$\vec{\delta\omega} = \text{rodrigues}(\mathbf{R}_{\text{est}}\mathbf{R}_{\text{gt}}^{-1}) \tag{3.22}$$

$$E_{\text{rotation}} = ||\vec{\delta\omega}|| \tag{3.23}$$

where $t_{\text{est}}, t_{\text{gt}}$ are 3×1 vectors representing an estimated translation vector and the ground truth of the translation component, respectively; $\mathbf{R}_{\text{est}}, \mathbf{R}_{\text{gt}}$ are the rotation matrices of each transformation; and Rodrigues is a function that returns a 3×1 vector describing the rotation error.

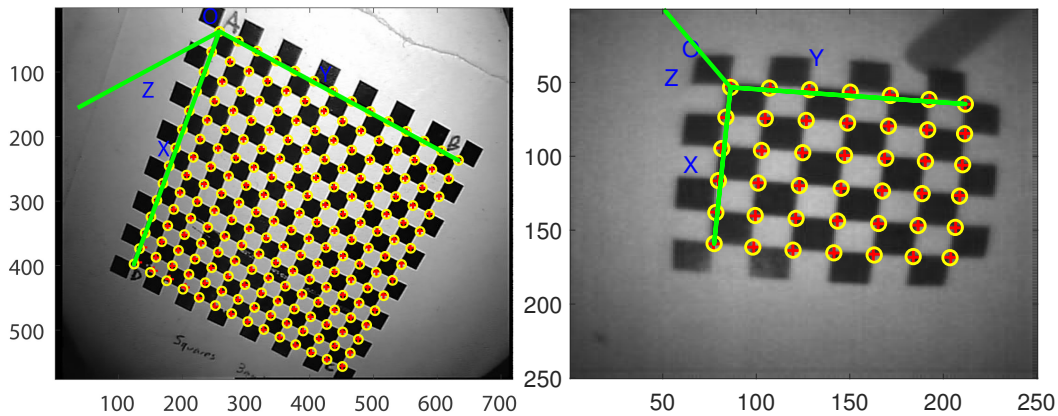
The experiments were run on a 2.6GHz Intel Core i7-4510U laptop and the processor took less than 0.5 seconds to complete one calibration routine to obtain

Figure 3.5: Experimental setup for capturing data. The specially designed scope with the NanEye stereo camera is mounted on the flange of the KUKA arm. The scope is 420 mms in length [112].

the initialisation for the refinement, while the others took less than 0.1 seconds. Increasing the number of motions did not significantly increase the computational cost in the calibration routine, whereas increasing the intensity of the noise increased the difficulty for the algorithm to converge, for both the ATA algorithm and the refinement using the Levenberg-Marquardt algorithm.

ATA was also tested with real robots to study its robustness and accuracy in the presence of noise sources that might be pose-dependent and non-Gaussian. The robots used in the experiments were the KUKA LBR IIWA 7 R800, KUKA LBR IIWA 14 R820 and the da Vinci Surgical Robot Standard as shown in Figure 3.2(a), 3.5 and 3.2(b), respectively. A stereo camera was used in the experiment along with a zero degree endoscope which is attached to the end-effector of both robots.

The frames on the KUKA arms were assigned following the standard DH convention [35] and Eq. 2.22 was used to construct a robot pose based on the joint configurations, while the convention used in the da Vinci Standard was the modified version (Eq. 2.28) based on its manual [50]. The DH parameters of the LBR IIWA 7, LBR IIWA 14 and the da Vinci Standard (ECM arm) are listed in table 7.1, 7.2, 7.3 in the Appendix. Note that the frame assignment may differ from one source to the others as some orientation can be specified arbitrarily.



(a) (b)

Figure 3.6: Images of calibration grid captured by (a) a da Vinci camera and (b) a NanEye stereo camera. The re-projected grid align very well on both calibration grid, but the image quality from a NanEye stereo is poorer. The units in the image axes are pixels.

In addition to the da Vinci scope, we also tested the algorithm with a NanEye stereo camera. This camera is a combination of two 1 mm CMOS imaging sensors. Each of the cameras provides a 249×250 resolution, which is much less than the da Vinci camera [113]. The scope was customised and attached to the KUKA as shown in Figure 3.5. Testing the algorithms with this setup would show that although the camera calibration algorithm accurately yields the camera's parameters, the calibration performance still depends on the quality of the captured images and it does not necessarily result in accurate hand-eye calibration. As shown in Figure 3.6, the image captured by a NanEye camera had a significantly poorer quality than that captured by the da Vinci camera.

In the calibration-performance evaluation, measuring an error metric in the experiment with real data is more challenging than experimenting with synthetic data as the accurate ground truth for the hand-eye transformation is not known. In [77], re-projection error is used in both optimising the matrix and assessing calibration performance. Another approach is to use Eq. 3.24 to predict the camera pose from the robot pose. This method is commonly used to assess the performance of hand-eye calibration algorithms. However, the method is similar to using the re-projection error to assess calibration performance because the predicted camera pose can be

used in Eq. 2.21 and obtain a single metric (the error in pixel instead of the error in the rotation and the translation components) to compare performance.

$$^{\text{cam}}\mathbf{T}_{\text{grid}}(\tau_{\text{predicted}}) = (\mathbf{XBX}^{-1})^{\text{cam}}\mathbf{T}_{\text{grid}}(\tau') \tag{3.24}$$

The alternative is to use the experimental setup as the evaluation criteria. In the setup of both experiments with KUKA and da Vinci and the camera calibration toolbox, the z axes of the frame assigned at the robot base and the grid were always pointing upwards and parallel to each other [50, 110, 114]. Therefore, we could use the calibrated hand-eye matrix to compute the transformation from the base coordinate to the grid coordinate for every camera and robot pose and averaged them using the formula in [64] to obtain $^{\text{grid}}\mathbf{T}_{\text{base}}$ as shown in Eq. 3.25 and 3.26. The second equation can be solved using the Levenberg-Marquardt algorithm.

$$\mathbf{Y}_i = (^{\text{cam},i}\mathbf{T}_{\text{grid}})^{-1}\ \mathbf{X}(^{\text{base}}\mathbf{T}_{\text{robot},i})^{-1} \tag{3.25}$$

$$^{\text{grid}}\mathbf{T}_{\text{base}} = \underset{^{\text{grid}}\mathbf{T}_{\text{base}}}{\arg\min} \sum_{i=1}^{N} (\text{logm}(^{\text{grid}}\mathbf{T}_{\text{base}}^{-1}\mathbf{Y}_i))^T\ \text{logm}(^{\text{grid}}\mathbf{T}_{\text{base}}^{-1}\mathbf{Y}_i) \tag{3.26}$$

The angle between the third column of the transformation $^{\text{grid}}\mathbf{T}_{\text{base}}$ and the vector $[0,0,1]^T$ can then be used to evaluate how well the algorithm can recover the orientation of $^{\text{base}}\mathbf{T}_{\text{grid}}$ as shown in Eq. 3.27.

$$\text{rotation error} = ||\arccos([0,0,1] \cdot\ ^{\text{grid}}\vec{z}_{\text{base}})|| \tag{3.27}$$

where $^{\text{grid}}\vec{z}_{\text{base}}$ is the third column of the transformation $^{\text{grid}}\mathbf{T}_{\text{base}}$.

For the translation component, we can use the RCM position in the da Vinci setup. The position is constant regardless of how the manipulator or scope is ori-

ented; it depends on which coordinate systems the point is observed from. Hence,

$$\begin{aligned}
\vec{p}_{\text{grid}} &= {}^{\text{grid}}\mathbf{R}_{\text{base}}\vec{p}_{\text{base}} + {}^{\text{grid}}\vec{t}_{\text{base}} \\
\vec{p}_{\text{grid}} - {}^{\text{grid}}\vec{t}_{\text{base}} &= {}^{\text{grid}}\mathbf{R}_{\text{base}}\vec{p}_{\text{base}} \\
||\vec{p}_{\text{grid}} - {}^{\text{grid}}\vec{t}_{\text{base}}|| &= ||\vec{p}_{\text{base}}|| \\
\text{translation error} &= (||\vec{p}_{\text{grid}} - {}^{\text{grid}}\vec{t}_{\text{base}}|| - ||\vec{p}_{\text{base}}||)^2
\end{aligned} \tag{3.28}$$

where $\vec{p}$ denotes the RCM position. The RCM $\vec{p}_{\text{grid}}$ is calculated by finding the position at which the z-axes of every camera pose intersect. The method is shown in detail in Section 4.2. Since the transformation ${}^{\text{base}}\mathbf{T}_{\text{grid}}$ is calculated from the hand-eye matrix, the recovered ${}^{\text{grid}}\vec{t}_{\text{base}}$ must also satisfy Eq. 3.28. Therefore, we can use the equation to validate the hand-eye calibration performance in terms of the translation component. Although the relationship between $\vec{p}_{\text{cam}}$ and $\vec{p}_{\text{robot}}$ can also be used to directly evaluate calibration performance without the need to calculate ${}^{\text{grid}}\mathbf{T}_{\text{base}}$, the transformations ${}^{\text{cam},i}\mathbf{T}_{\text{grid}}$ and ${}^{\text{base}}\mathbf{T}_{\text{robot},i}$ are noisy and the calculated RCM $\vec{p}_{\text{cam}}$ and $\vec{p}_{\text{robot}}$ may not be constant.

However, Eq. 3.28 does not apply to the motion of the KUKA robot as it does not have the RCM. The remaining method involves using the residue of the hand-eye equation to evaluate calibration performance in terms of the translation component. Although the residue of the matrix product $(\mathbf{AX})^{-1}(\mathbf{XB})$ does not directly represent the error in the translation estimation, it should serve as a measure of how close the estimated solution is to the optimal solution in this formulation.

For robots, more than 25 poses were collected. We randomly selected N poses as an input to each calibration algorithm in a succession of calibration trials (N is run from 3 to 13, i.e. 2 to 12 motions for successive motions). The selection of measurements was randomly repeated 100 times to obtain statistically meaningful results. For each calibration routine, Eq. 3.27 was used to evaluate calibration accuracy in terms of the rotation, Eq. 3.28 in terms of the translation for da Vinci data and computed the error from 2.17 for the KUKA data. Furthermore, we also used the result of the predicted camera pose from Eq. 3.24 to assess calibration performance in terms of re-projection errors. The evaluation of each sample was

then averaged across the number of samples.

For each experiment, ANOVA was applied on the selected raw calibration results because the noise generated in the experiments was Gaussian, and the error from the camera parameters resembles the normal distribution. The purpose is to test whether the difference in the comparisons is statistically significant: i.e., the analysis was applied to 100×4 raw calibration errors for both rotation and translation. Except for the result shown in Figure 3.8, the independent samples t-test was used, since we only have two raw results.

3.4 Experiments with synthetic data

This section shows the comparison of the performance of ATA and the algorithms in the literature in terms of their calibration accuracy. We first display the convergence rate of ATA when the algorithm uses different initialisations. Then we validate the stereoscopic formulation by comparing calibration accuracy with the monocular formulation. The algorithms applied here are the proposed method "ATA", classic hand-eye solution "TSAI" [44], classic dual quaternion solution "DQ" [73] and improved dual quaternion solution "IDQ" [45]. These abbreviations are used consistently throughout this section.

However, since Figure 3.8 shows that the formulation using stereo information increases calibration accuracy regardless of the selected algorithms, the plots shown in subsequent Figures are the results using the stereoscopic formulation.

The table showing the p-value for the selected simulated parameters is also displayed next to the respected figures. If the p-value of the test is less than 5 percent (< 0.05), we conclude that the difference in calibration performance between each algorithm does not occur by chance and it is statistically significant. The same technique is also applied in the experiment with real data.

3.4.1 Effect of error in the robotic positioning system

In this section, we observe the effect of inaccuracy in the robot positioning system on the re-projected grid. Poses of a robot were randomly generated and corrupted with 0.025 mms Gaussian noise to simulate the robot with an absolute accu-

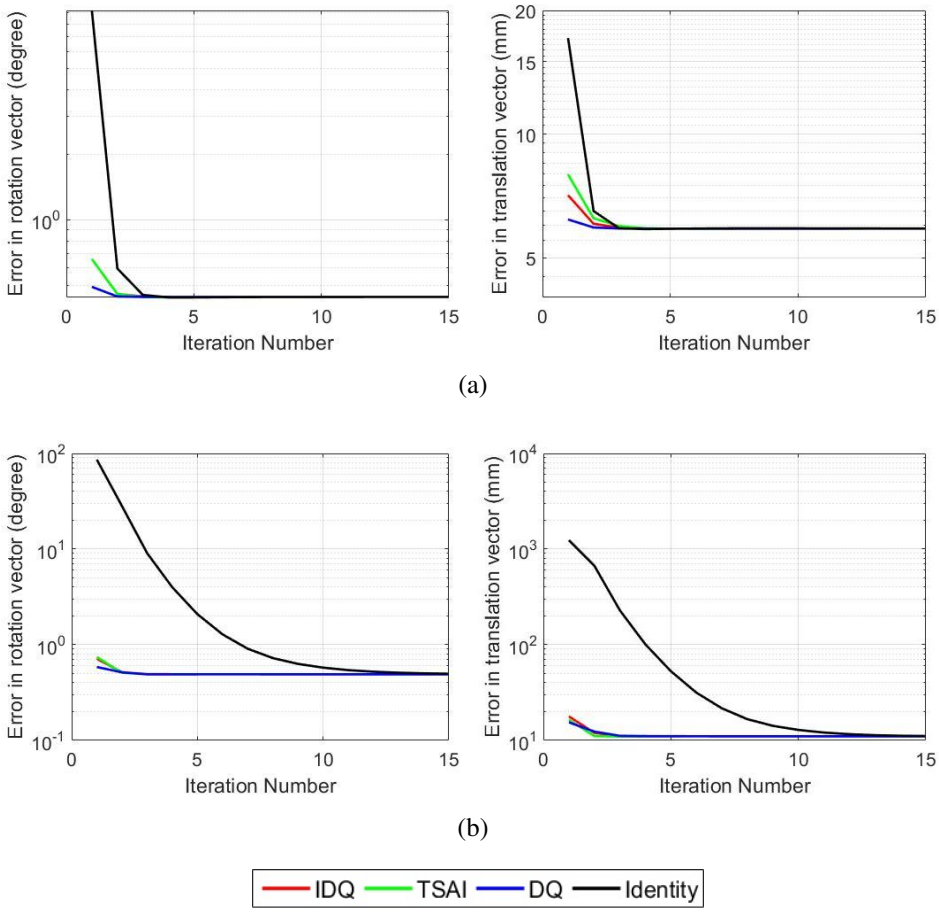


Figure 3.7: Comparison of convergence rates of ATA with different initialisation methods is displayed using the mean of the translation and rotation errors for improved dual quaternions (IDQ), Tsai's linear method (TSAI) and dual quaternions (DQ). (a) The ground truth solution is close to the identity matrix. (b) The ground truth solution is far from the identity matrix.

racy of 0.1 mm in the positioning system. After that, we compared the re-projected checkerboard from using uncorrupted poses with one using corrupted poses. Note that we use the ground truth of ${}^{\text{base}}\mathbf{T}_{\text{grid}}$ and ${}^{\text{cam}}\mathbf{T}_{\text{robot}}$ to complete the loop and create the comparison. Figure 3.3(a) shows that the majority of the re-projection errors are higher than 1 pixel. This indicates that the error in the robot positioning system has a larger impact than the error in the camera calibration.

3.4.2 Convergence rate with different initialisation methods

Since ATA requires initialisation to start the alternation solver, we also conducted an experiment to show how the error converges when different methods are

applied to compute the initial solution. The initialisation in this experiment was created from different hand-eye algorithms as well as the identity matrix. ATA was tested in two situations: when the hand-eye matrix was close to the identity matrix (Figure 3.7(a)) and when it was far from the identity matrix (Figure3.7(b)). The results showed that ATA converged to the same solution regardless of the starting point. This confirms the property of the alternating optimisation that every iteration is globally optimising the problem and can avoid local minima [107].

Table 3.1: P-value of the raw results shown in Figure 3.8. The unit of the σ_t is mm.

	$\sigma_r = 0.2^\circ$ $\sigma_t = 0.4$	$\sigma_r = 0.3^\circ$ $\sigma_t = 0.6$	$\sigma_r = 0.4^\circ$ $\sigma_t = 0.8$	$\sigma_r = 0.6^\circ$ $\sigma_t = 1.2$	$\sigma_r = 0.7^\circ$ $\sigma_t = 1.4$	$\sigma_r = 0.8^\circ$ $\sigma_t = 1.6$	$\sigma_r = 0.9^\circ$ $\sigma_t = 1.8$
ATA ($\mathbf{R}$)	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
ATA ($\vec{t}$)	< 0.01	< 0.01	< 0.01	< 0.01	0.0139	0.0112	< 0.01
IDQ ($\mathbf{R}$)	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
IDQ ($\vec{t}$)	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
TSAI ($\mathbf{R}$)	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
TSAI ($\vec{t}$)	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
DQ ($\mathbf{R}$)	0.0493	0.0375	0.0229	< 0.01	0.0100	0.0169	< 0.01
DQ ($\vec{t}$)	0.0378	0.0291	0.0400	0.0672	0.0520	0.0526	0.0539

3.4.3 Inclusion of stereo information

In this experiment, we used eight motions as input to each calibration algorithm and noise was increased from 0 to 1 degree with an increment of 0.1 degrees in the rotation component. Noise in the translation was increased from 0 to 2 mm with an increment of 0.2 mm.

Figure 3.8 shows the comparison between different hand-eye formulations under increasing additive noise: monocular and stereoscopic formulation. In the same experimental setup, using the stereoscopic formulation yielded a more accurate calibration result. This verifies that the three additional constraints can suppress the influence of noise and result in a better calibration performance. Moreover, the figure also shows that ATA is more robust than the other methods when the noise coefficient is increased. This result is more evident in Figure 3.10(a)-3.10(c).

Note that, in this section, we used the independent samples t-test to verify the statistical significance of the comparison since there were only two independent datasets and ANOVA requires three or more datasets. As shown in Table 3.1, most

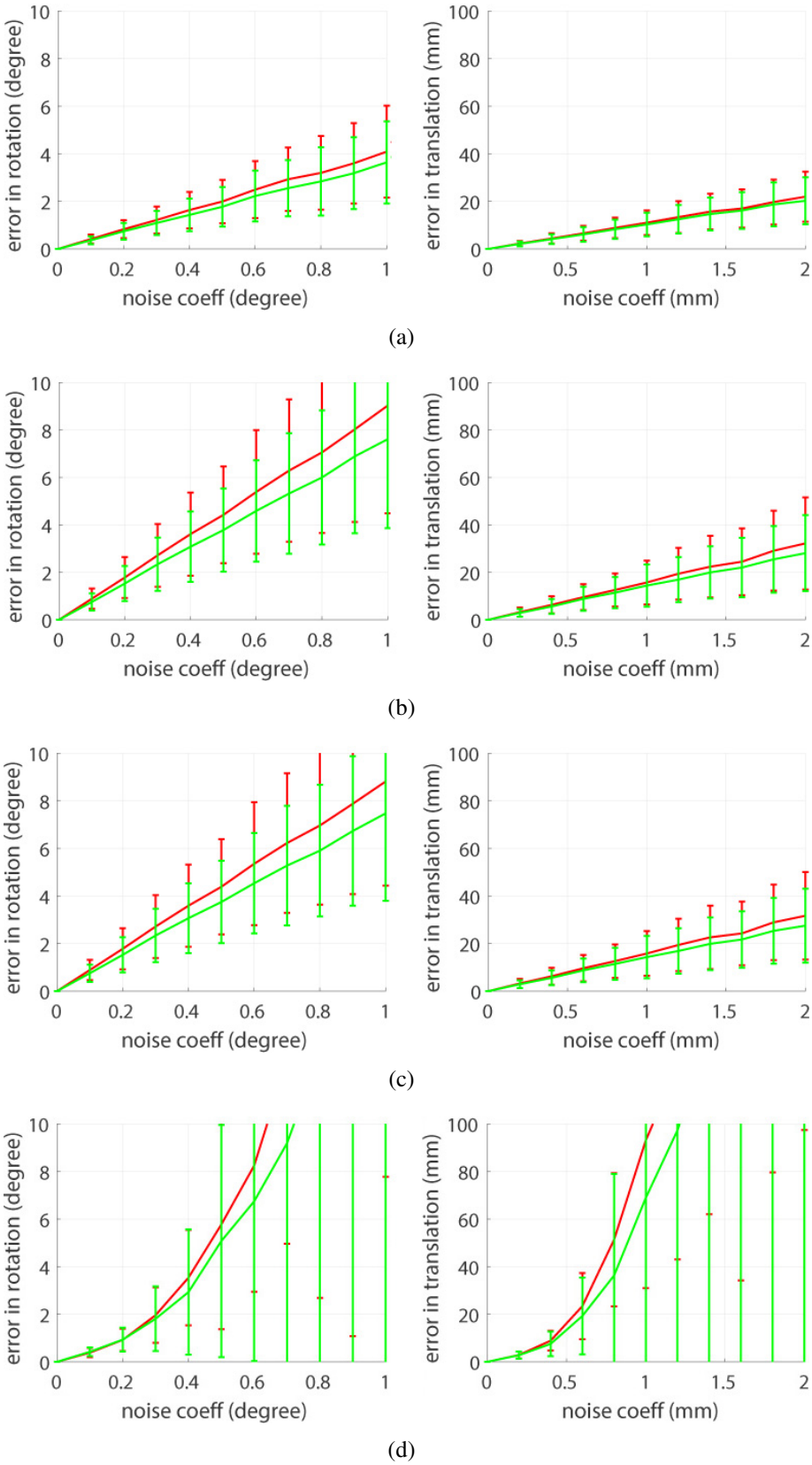


Figure 3.8: Comparison of monocular and stereoscopic formulations using different hand-eye algorithms. Distribution of the translation and rotation errors are shown for the monocular (red) and the stereoscopic (green) cases. Independent samples t-test is applied and shown in Table 3.1. (a) ATA, (b) IDQ, (c) Tsai and (d) DQ.

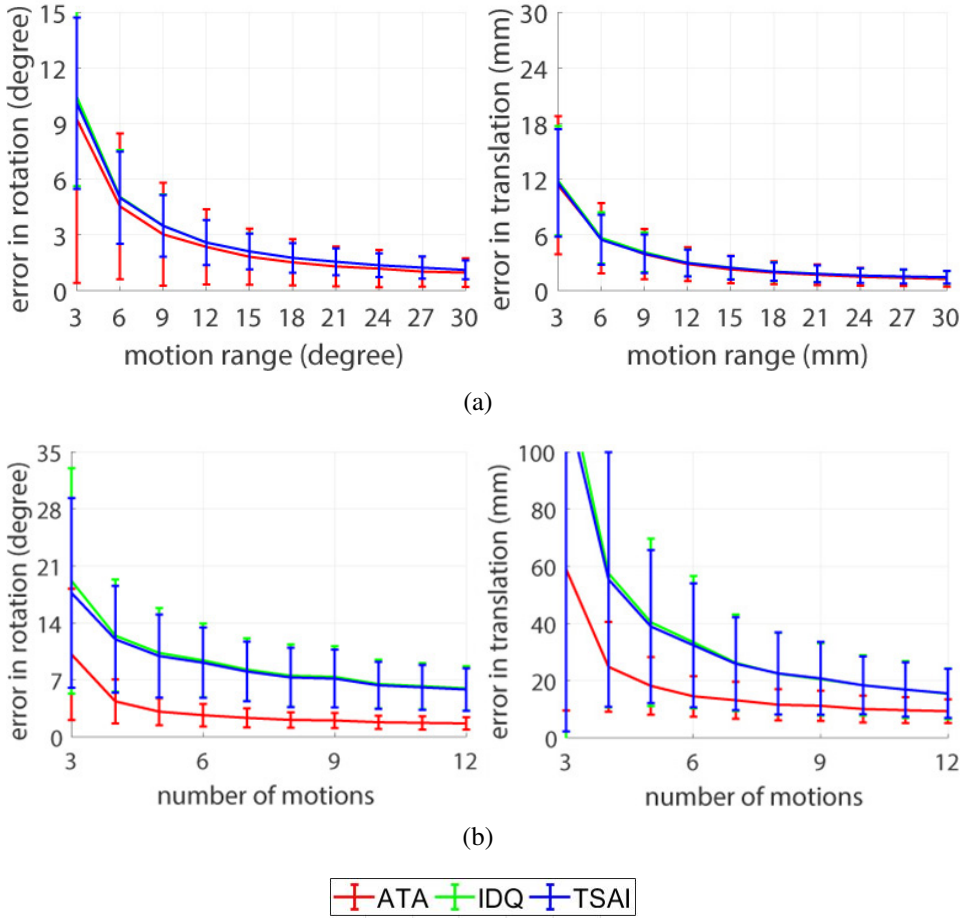


Figure 3.9: Stereoscopic hand-eye calibration using synthetic data: (a) Increasing motions range. (b) Increasing number of motions.

of the comparisons are statistically significant, except for the DQ part, which has a very high standard deviation, and hence unreliable results. This shows that the method is not stable when noise is added into the system, as the method uses a quadratic equation (which is not stable as previously mentioned) to calibrate the matrix. Therefore, due to this instability, the DQ method will be omitted from the plots so that the differences in more stable algorithms can be focused so as to increase the statistical significance of the comparison.

3.4.4 Increasing motion range

It was shown in [41] that calibration accuracy can be improved with a wider motion range. The proof is shown in [44] which formulate the case of three motions and the authors further deduce that the RMS error of the rotation component is

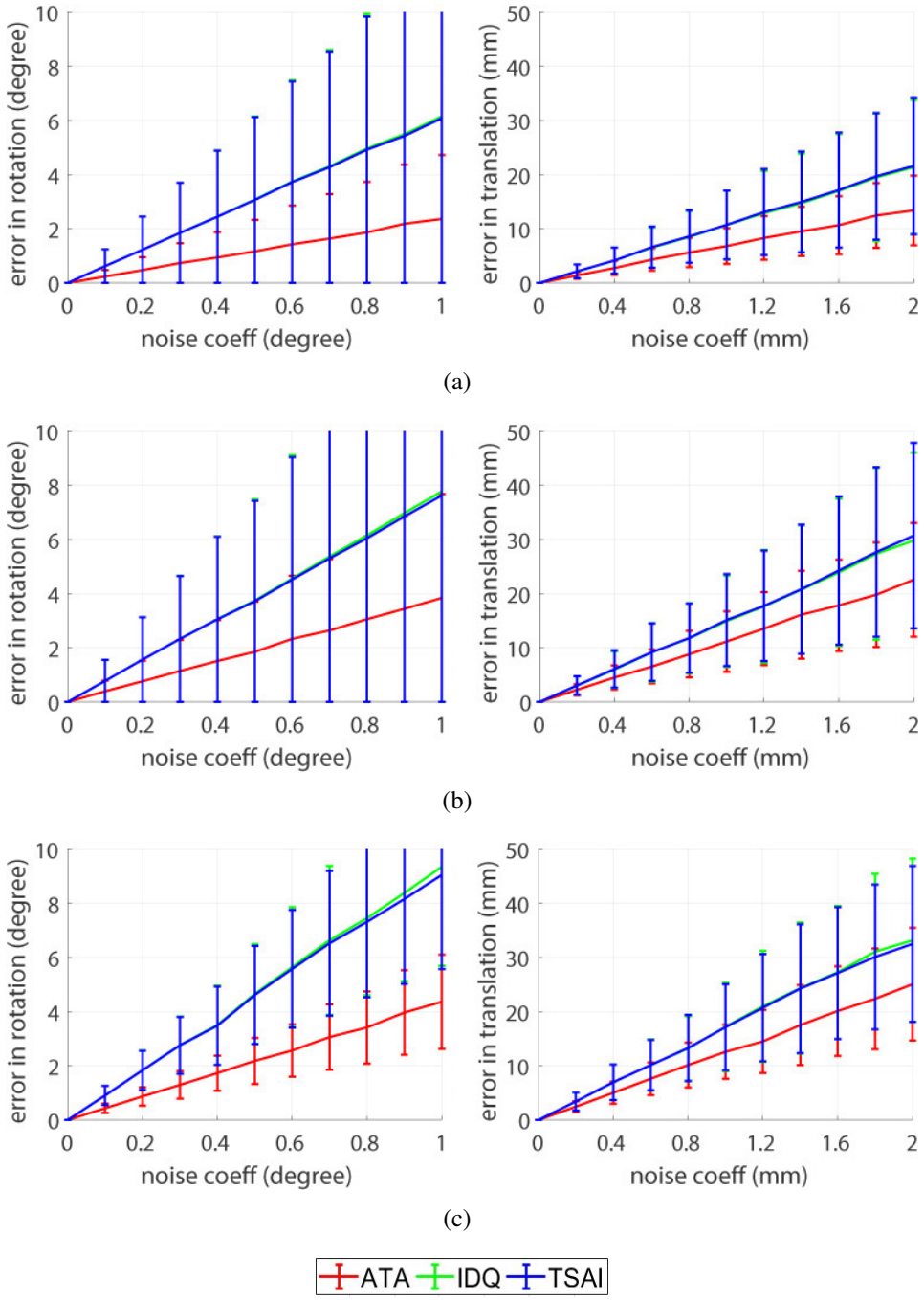


Figure 3.10: Stereoscopic hand-eye calibration using synthetic data: (a) Increasing noise in robot motions. (b) Increasing noise in both robot and camera motions. (c) Increasing noise in robot motions, camera motions and stereo calibration. ANOVA is applied to the raw results from these experiments and the p-value for each simulated parameter in every experiment is less than 0.01.

inversely proportional to the sine of the angle between the rotation axes. Therefore, to appropriately select the data for the hand-eye problem, we should select pairs

of rigid transformations containing wide rotation motions to increase calibration accuracy [57]. Examples of wide and small motion ranges are shown in Figure 3.11(a) and 3.11(c), respectively.

In this experiment, we generated transformations that have a translation magnitude of 3 mm to 30 mm with an increment of 3 mm, whereas the rotation had a magnitude of 3 degrees to 30 degrees with an increment of 3 degrees. These small motion ranges were in line with the restricted motions expected from robotic surgical instruments. The results are displayed in Figure 3.9(a).

All cases were evaluated with eight input motions, 1 mm of Gaussian noise in translation and 1 degree of Gaussian noise in rotation in both the end-effector and camera motions. Figure 3.9(a) shows that motion range has an impact on calibration performance. The calibration error is large when the motion range is small and decreases as the motion range is wide. This becomes more evident in the results in the experiments with real data where we compared the calibration performance of the two different robots.

3.4.5 Increasing number of motions

The number of motions is also one of the main criteria for increasing the calibration accuracy. This has been extensively proved in the literature [41, 44, 57]. As stated in the literature, the noise in the rotation component can be suppressed by a factor of $\sqrt{N}$ where N is the number of motions. However, included motions also have to conform to a wide motion range.

Figure 3.9(b) shows the comparison of the calibration performance with an increasing number of motions. Similarly to the previous experiment, constant noise of 1 mm in the translation component and 1 degree in the rotation component was applied in both the robot and camera motions. Motion range in this setup was kept fixed at 20 mm in the translation component and 5 degrees in the rotation component. Fig 3.9(b) shows that the calibration error decreased as the number of motions increases and that ATA outperformed the other methods for any number of motions.

3.4.6 Increasing Gaussian noise

This experiment compared the robustness of each algorithm against increasing Gaussian noise. Figures 3.10(a)-3.10(c) display the results when increasing zero-mean Gaussian noise is added to the camera, the robot, and stereo information. Noise in the translation component was increased in 0.2 mm steps from 0 to 2 mm, while a noise in the rotation component is increased in a step of 0.2 degrees from 0 to 2 degrees. In each simulation, eight motions and the same motion range as previous experiments were used.

Given that a picture is taken with clear edges and features in the context of robotics, cameras are often considered as noise-free sensors. There may exist a sub-pixel error caused by inaccurate intrinsic calibration, but such error is typically considered negligible in comparison to the errors propagated through a robot's kinematic chain. The robot pose is obtained through the forward kinematics whose input is from noisy position readings and inexact nominal kinematic parameters that usually do not consider stress in strings and cables in the presence of a gravitational force. Figure 3.10(a) displays the error when noise is added to end-effector's movement. The error increases with the noise as expected, however, ATA's performance degrades at a slower rate than the other tested algorithms.

Figure 3.10(b) shows the result when noise is added to both camera and end-effector motions. The noise intensity as shown in Eq. 3.20 is the same for both data. The calibration error rises faster than Figure 3.10(a). However, the comparison shows no difference from the previous experiment: the calibration error of the ATA algorithm increases at a slower rate than the other approaches for both the rotation and translation components.

Finally, the result presented in Figure 3.10(c) is the closest to the scenario with real data. In the case of stereo vision, not only are the camera and robot kinematic parameters noisy, but an error also exists from the stereo camera calibration. Since the formulation uses this information to take into account the additional constraints, the error has to be considered. However, as in the previous cases, although the calibration error is higher than Figure 3.10(b), ATA is still the best performer for

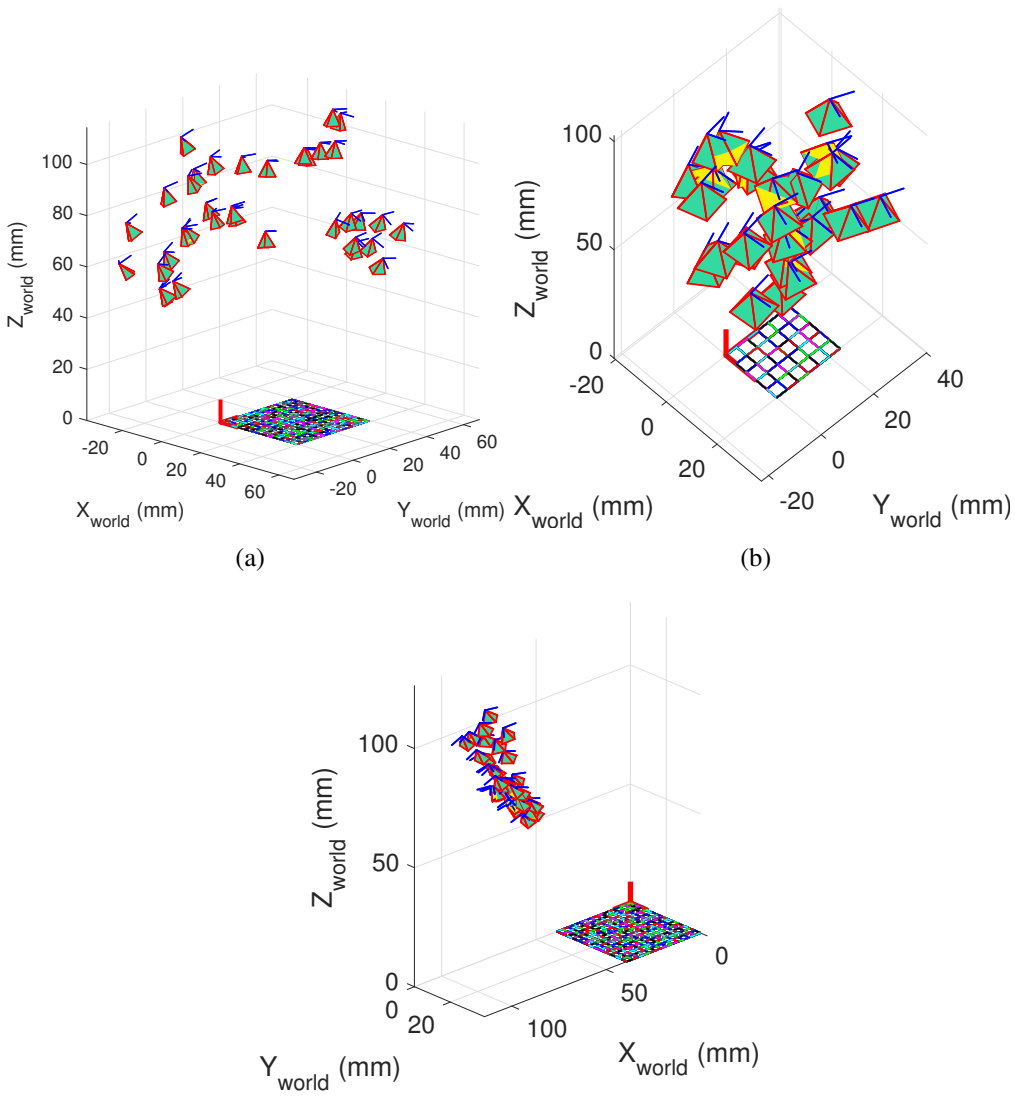
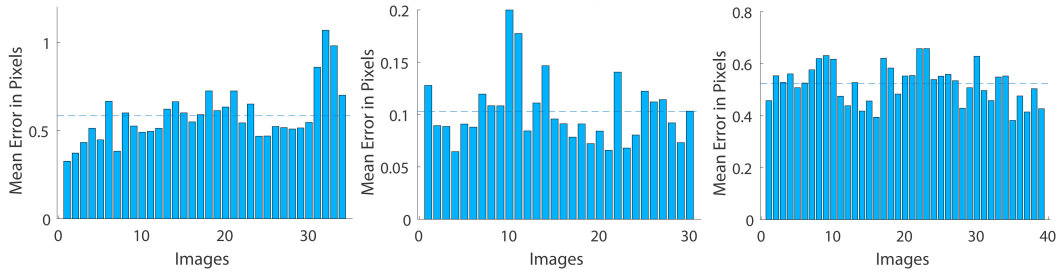


Figure 3.11: (a) Camera poses with respect to the calibration grid obtained with the KUKA robot. Images can be acquired for a wide camera motion range, resulting in hand-eye calibrations with a higher calibration accuracy. (b) Camera poses with respect to the calibration grid obtained using the KUKA arm with a NanEye stereo camera. The motion range and the calibration grid are much smaller than the case of the da Vinci camera as the NanEye camera has a smaller field of view. Therefore, not only the image quality is poor, the motion range included in the calibration is also not as wide as the da Vinci camera. (c) Camera poses with respect to the calibration grid obtained with the da Vinci robot. The camera motion is constrained to a smaller range of rotations and translations around the RCM and thus solving the hand-eye problem to get an accurate hand-eye matrix is even more challenging than the previous cases.



(a) (b) (c)

Figure 3.12: The re-projection error computed from the intrinsic and extrinsic parameters estimated by the MATLAB calibration toolbox. The extrinsic parameters are used in the hand-eye calibration. The dashed line indicates the overall mean error. (a) Re-projection error from the KUKA data with the da Vinci scope (Overall mean error = 0.57 pixels) (b) Re-projection error from the KUKA data with the NanEye scope (Overall mean error = 0.11 pixels) (c) Re-projection error from the da Vinci data (Overall mean error = 0.55 pixels).

increasing noise.

3.5 Experiments with real data

The camera poses used in the experiments with the KUKA data are shown in Figure 3.11(a) and 3.11(b) in the grid coordinate. We collected these poses by manually moving the robot arm around the calibration grid. The camera poses are spread diversely above the grid, although the camera poses in Figure 3.11(b) are also spread above the grid in a smaller range due to the camera specification. This will affect calibration accuracy, according to the results shown in Figure 3.9(a).

As opposed to the camera poses in the KUKA data, Figure 3.11(c) shows that the da Vinci Standard data has a rather small workspace and the camera pose is not as wide as KUKA's. This is because that the configuration of the da Vinci constrains the camera to be close to the RCM. Therefore, the camera motion tends to be around the insertion axis of the robot with limited translational and rotational movements and cannot produce a wide motion ranges. This makes the hand-eye problem in the robot significantly more challenging, as the noise has a greater impact on the calibration as shown in Figure 3.9(a). Furthermore, according to the robot manual, the accuracy of its positioning system is acceptable up to a couple of mm for translation

and a couple of degrees for rotation and also does not account for mechanical compliances. This indicates that the system is not as accurate as the KUKA in terms of its overall motion. These factors show that solving the hand-eye problem in the da Vinci Standard is more challenging than using the KUKA.

In terms of the noise in the camera data, Figure 3.12 shows the re-projection error in pixels using the estimated extrinsic parameters which are used in the hand-eye calibration routine. Since Figure 3.12(a) and 3.12(c) are obtained using the same camera, the errors in the re-projection metric are close. This suggests that the difference in hand-eye calibration performance between these two setups is solely due to the difference in noise resilience in each robot. On the other hand, despite having the lowest re-projection error among the datasets (Figure 3.12(b)), it may not yield the best hand-eye calibration result as a NanEye stereo camera's resolution is three times as low as the da Vinci's. This will be evident in the next section.

3.5.1 Experiments with the KUKA LBR IIWA 7 R800

Figure 3.13(a) shows that all algorithms yield a low calibration error. This is because of the accurate positioning system in the KUKA LBR IIWA 7 R800, a wide motion range in its calibration and a sub-pixel accuracy in the camera calibration algorithm. However, a small error in the input data can produce a significantly higher calibration error in the hand-eye solution, which is in accordance with the simulation results described in Figure 3.9-3.10. In terms of calibration performance, ATA still slightly outperforms existing algorithms and achieves the smallest error in the camera poses estimation. The rotation error is around 1.5 degrees and the translation error is around 3.2 mm.

In contrast, the plots in Figure 3.13(b) contain significantly larger errors in all evaluation metrics than those in Figure 3.13(a). The only difference in the two experiments is the camera which suggests that despite a small re-projection error, the extrinsic parameters can still be inaccurately estimated (an estimation that also depends on the quality of the captured images). The inaccuracy caused by the poor quality of the camera can also be observed in the high standard deviation in the plots, as suggested by Figure 3.10(c) a larger noise in the system induces instability

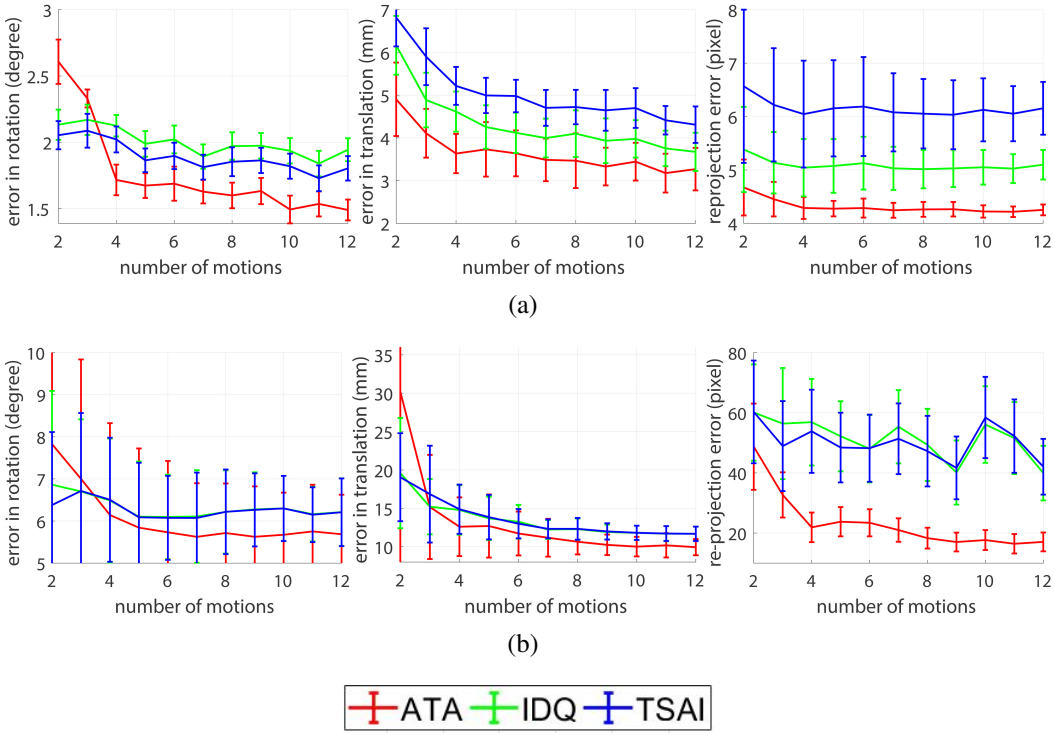


Figure 3.13: Calibration performance of each algorithm when applied to the real data from KUKA with (a) a da Vinci camera and (b) a NanEye stereo camera in terms of the rotation component (left), the translation component (middle) and the re-projection error (right).

in the calibration and hence a higher standard deviation.

From a comparison of the reported standard deviation in Figure 3.13(a) with that of the synthetic data, we can deduce the noise in the KUKA robot. The plots report a standard deviation of 0.5 mm in the translation component and 0.07 degrees in the rotation component when we use eight motions in the calibration, which, by linear interpolation, suggests that the noise inside the system is around 0.0979 mm and 0.04 degrees for the two respective components. Knowledge of this information is useful for the simulation of the robots as the generated noise characteristic can be produced a high degree of fidelity.

The right-hand side plots in both Figures show the re-projection error in pixels when the grid in the world coordinates is projected back onto the image using the predicted camera pose. The result shows that ATA produces the smallest re-projection error among all methods.

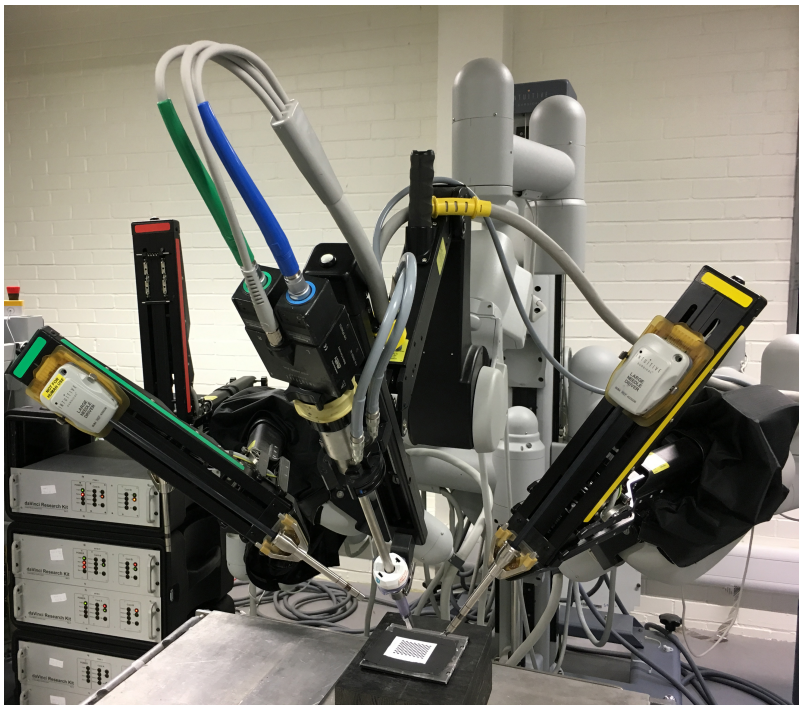


Figure 3.14: Experimental setup for capturing data from the da Vinci Standard.

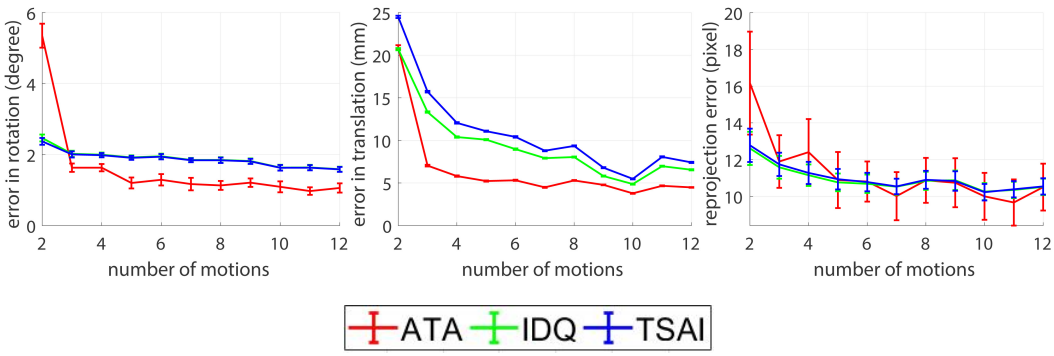


Figure 3.15: Calibration performance of each algorithm when applied to the real data from the da Vinci Standard in terms of the rotation component (left), the translation component (middle) and the re-projection error (right).

3.5.2 Experiment with the da Vinci Standard

Figure 3.15 shows the calibration performance in terms of camera pose prediction and re-projection error, respectively. The comparison shows that even with a small motion range, ATA can still outperform the other algorithms and its robustness increases with the number of motions included in the calibration which agrees with the experiments with synthetic data.

However, the converged calibration result still yields higher errors than the

ones in the experiment with the KUKA. As shown in the results, the calibration result already converges with 12 motions, so one factor that can improve calibration accuracy is motion range. Therefore, an insufficiently wide motion range is the bottleneck parameter in achieving more robust hand-eye estimations with this system.

The other interesting result is a relatively low standard deviation (0.05 mm in the translation component and 0.07 degrees in the rotation component) in comparison to Figure 3.13(a). This suggests that the noise in da Vinci is lower than the KUKA (0.001 mm in the translation component and 0.04 degrees in the rotation component), but its positioning system is not as accurate. This agrees with the manual in suggesting that mechanical compliance and external forces are not accounted for by the dVRK, therefore creating an offset in the pose [50] and requiring robot calibration to minimise the discrepancy.

3.6 Discussion

This chapter presents a hand-eye calibration algorithm that uses the adjoint transformation and stereo information to derive a new formulation to increase calibration accuracy in surgical robots. Based on the results shown in the previous section, implementing the stereo constraints into the original hand-eye equation increases the robustness of the calibration algorithms and the adjoint transformation creates an improvement in solving the hand-eye problem.

First, one of the key findings of the chapter is that the proposed algorithm can deal with the noise in the robot parameters. Although it is not always the case, the compositions of surgical robots cannot usually provide sufficiently accurate positioning for RMIS applications which require a precise positioning system. This inaccuracy in the robot parameters creates a significant drift in the re-projected image as demonstrated in Figure 3.3. On the other hand, the camera motion estimated by the camera calibration [42] provides a more accurate pose which is a better input for the hand-eye problem. Given that the CAI applications in surgery such as visual servoing or augmented reality require accurate calibration, the propagated error

from the robot parameters should not be used. Therefore, one of the reasons why ATA works better than the other approaches is its ability to use a camera pose in the translation estimation instead of using the noisy pose from the robot.

The second key finding is the new formulation of the hand-eye problem using the stereo information to increase calibration accuracy. RMIS usually involves using a stereo camera during the procedure for navigation and localisation purposes, but stereo information has never been used in any calibration algorithm. The proposed algorithm makes use of the pre-calibrated stereo camera to create more constraints on the problem and the results show a clear improvement over the original formulation. Furthermore, this finding is validated with the surgical robots and this confirms that the calibration algorithm can still perform well, even with noisy data that contain inexact and inaccurate parameters, including the stereo information itself, that exist in a surgical environment.

The key advantage of ATA and stereoscopic formulation is a robust-to-noise hand-eye calibration approach that offers better calibration accuracy than the state-of-the-art approaches. With accurate hand-eye transformation, the loop of transformation between the camera-end (surgical instruments, tissues and anatomical structures at the operative site) and the robot-end (kinematics and control) can be completed and introduces the potential for an accurate real-time localisation of the whole surgical environment. Thus, applications such as visual servoing, dynamic virtual fixtures or augmented reality can be integrated with small discrepancy which can provide guidance in the operation and result in a safer and possibly faster operation.

One of the limitations of the proposed algorithm is that it cannot fully suppress the error from an uncalibrated robot. ATA performs better than the other algorithms because it does not use the orientation of the robot pose, but it still requires the translation component. Therefore, the offset in the kinematic parameters and joint compliance is still propagated to the hand-eye matrix. This is evident in Figure 3.15 that even though the noise in the system is small, the calibration error is still high. While robot calibration using external sensors such as an optical tracker or an

electromagnetic tracker can be applied to mitigate the discrepancy, the procedure during an operation will be more complicated as it requires an external tracking system and a slight change can invalidate the calibration result.

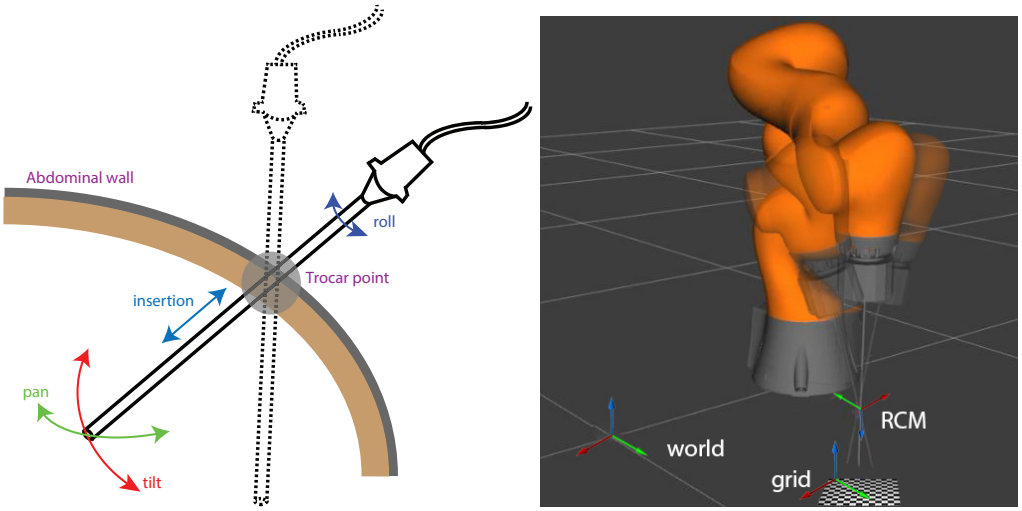
The other limitation of ATA is that it still cannot deal with a small motion range of a camera. This problem has been noted in the literature regarding the hand-eye problem [41, 44, 57]. Specifically, it is essential to obtain a wide motion range in order to accurately estimate the hand-eye matrix. However, this criteria is not always feasible and sometimes can be difficult to achieve in the context of keyhole surgical robots as the camera motion is usually confined in the area around a predefined RCM. The limitation is shown in Figure 3.11(a)-3.11(c) that the da Vinci Standard has a smaller motion range compared to the KUKA, which in turn significantly deteriorates calibration accuracy as shown in Figure 3.15. Nevertheless, this problem can be overcome by using the other object in the surgical environment as a calibration object [67] or imposing further constraints by using RCM to simplify the hand-eye problem. These concepts are introduced in the following chapters.

Chapter 4

Hand-eye calibration using the remote centre of motion

As shown in the previous chapter, one of the main conditions for an accurate hand-eye calibration is to acquire images of a calibration target with a wide range of camera motions that fully explore all six DoF of possible movement, which is also the same criterion for obtaining an accurate camera calibration result [42]. The cameras mounted on surgical robots, however, are mechanically constrained to move around the RCM to ensure that instruments are confined to a motion in the vicinity of the trocar entry ports [47] as shown in Figure 4.1(a). This limits the camera motion to only four DoF (three in rotation and one in translation) and results in ill-conditioned hand-eye constraints. Even though the three DoF in rotation can be sufficient to provide an accurate calibration, the degree of rotation in each axis is small. While a possible solution would be to allow a surgical robot to move freely in six DoF during the calibration phase, this is neither practical when there is a change in the robot setup nor possible for mechanisms with mechanical RCM implementations [49].

In this chapter, we show that it is possible to develop a constraint on the RCM configuration to improve the accuracy of hand-eye calibration. We introduce a new formulation to determine the hand-eye transformation in the RCM constrained setting, by first estimating the RCM location in the camera reference frame and then using this information to simplify the classic hand-eye formulation. The advantages



(a) (b)

Figure 4.1: (a) The schematic shows a magnified version of the type of movement of the camera when being used in RMIS. RCM is denoted at the trocar point to minimise a chance of a robot arm damaging the surrounding tissues [48, 49]. The camera motion is restricted around the RCM and this provides a very small motion range which is not sufficient for a decent calibration. (b) Example setup for hand-eye calibration shown in simulation using RViz to illustrate the type of the motion around the pre-defined RCM. The coordinate frames in the simulated environment are denoted as shown in the figure. The frame "grid" is usually assigned at the calibration grid, and the frame "world" is assigned as the reference point for the robot pose and sometimes is defined at the robot base using the notation "base".

and contributions of this approach are the following:

- Incorporating the RCM position allows the characterisation of robot motions in four DoF and this changes the hand-eye formulation such that constraints are no longer ill-posed.

- A known RCM position enables the formulation of the hand-eye calibration as an absolute pose problem with a stable solution.

- An RCM constrained hand-eye calibration can be more accurate than a classic calibration using free motion, due to the formulation's simplifications. This suggests that a more convenient path towards accurate hand-eye calibration in robotic surgery is through correctly modelling its motion constraints rather than allowing the robot to move freely in a calibration phase.

4.1 Formulation of the RCM

Consider the relative motion equations (Eq. 2.19 and 2.20). Although the motion around RCM has three DoF in rotation, the range of it is rather small which makes the rotation components approach the identity, i.e. $\mathbf{R}_A, \mathbf{R}_B \to \mathbf{I}_3$. Therefore, the rotation component (Eq. 3.1) of the hand-eye equation becomes $\mathbf{R}_X = \mathbf{R}_X$.

Hence, Eq. 3.1 always holds regardless of the relative motions and cannot be used to solve for rotation. This is shown in the experimental section where the conventional algorithms always fail at determining the rotation component. This RCM-constrained motion however, does not affect on Eq. 3.2 as the equality between $\mathbf{R}_A\vec{t}_X$ and $\vec{t}_X$ depends on the magnitude of $\vec{t}_X$. This effect can be easily proved by a rotation motion in the two-dimensional system; a point $p(x,y)$ is rotating around the origin by an angle θ, so the motion traversed by the point is calculated by $\theta\sqrt{x^2+y^2}$, where $\sqrt{x^2+y^2}$ is the magnitude of the translation. Hence, as long as the magnitude of $\vec{t}$ does not zero, the equality between $\mathbf{R}_A\vec{t}_X$ and $\vec{t}_X$ cannot be assumed, even though $\mathbf{R}_A \to \mathbf{I}_3$.

Given that the scope can be freely moved along the insertion axis with a small rotation, the translation component is apparently a non-zero vector. Therefore, the only equation that can be used to solve the hand-eye problem is Eq. 3.2 which is not affected by the identity rotation matrix. However, the hand-eye problem is a six DoF problem and cannot be solved using only the translation component. To mitigate this motion range problem, a constraint on the RCM position is introduced into the hand-eye problem.

$$^{\text{cam},i}\mathbf{T}_{\text{grid}} \begin{bmatrix} \vec{p}_{\text{grid}} \\ 1 \end{bmatrix} = \mathbf{X}^{\text{robot},i}\mathbf{T}_{\text{base}} \begin{bmatrix} \vec{p}_{\text{base}} \\ 1 \end{bmatrix} \tag{4.1}$$

where the frame cam, i and robot, i are defined at the camera tip and the robot end-effector at the pose i, respectively and $\vec{p}_{\text{base}}$ and $\vec{p}_{\text{grid}}$ are the same RCM position, but are represented in different coordinate systems. Note that the notation $\vec{p}$ in this chapter denotes the RCM position in the coordinate system in the subscription. The transformations $^{\text{cam},i}\mathbf{T}_{\text{grid}}$ and $^{\text{robot},i}\mathbf{T}_{\text{base}}$ can be calculated by calibrating the camera

and using forward kinematics, respectively.

The position $\vec{p}_{\text{base}}$ is usually pre-defined along the scope as shown in Figure 4.1(a). In Eq. 4.1, there are currently two unknowns to be solved in the next section; $\vec{p}_{\text{grid}}$ and the hand-eye matrix $\mathbf{X}$. Determining the RCM in the grid coordinate $\vec{p}_{\text{grid}}$ will provide three more constraints to the ill-posed hand-eye problem and make the problem in the RCM setup solvable.

4.2 Hand-eye calibration with the RCM

4.2.1 Calculating the RCM

To find $\vec{p}_{\text{grid}}$, we have to make use of the results demonstrated in [115]; in the RCM setup in which the scope is confined such that it has to go through and pivot around the RCM, one of the axes defining the camera pose intersects the RCM. Although this assumption is not verified in practice as it does not account for the radial distortion and a stereo-scope creates ambiguity about where the RCM is, the pose estimation result nevertheless outperforms the classic solution. Therefore, it is safe to assume that the axes (usually the z-axis) along the scope defining each camera pose intersects at RCM $\vec{p}_{\text{grid}}$ as shown in Figure 4.2. Hence, the first step to find $\vec{p}_{\text{grid}}$ is to find the camera pose that corresponds to each robot pose by solving the homography problem.

To determine the point $\vec{p}_{\text{grid}}$ of intersection in the grid coordinate, we have to find the point that minimises the Euclidean distance between itself and its projection on each line. The distance is defined by,

$$D(\vec{p}) = ||(\vec{o}_{\text{grid}} - \vec{p}) - ((\vec{o}_{\text{grid}} - \vec{p})^T \vec{d}_{\text{grid}})\vec{d}_{\text{grid}}|| \tag{4.2}$$

where $\vec{o}_{\text{grid}}$ is an arbitrary point along the z-axis defined by the frame "cam, i" and $\vec{d}_{\text{grid}}$ is a unit vector describing the direction of the axis. Note that every component in Eq. 4.2 is in the grid coordinate.

In a monocular case, the third and the fourth column of the camera pose in the grid coordinate ${}^{\text{grid}}\mathbf{T}_{\text{cam},i}$ can be directly used as $\vec{d}_{\text{grid}}$ and $\vec{o}_{\text{grid}}$, respectively because

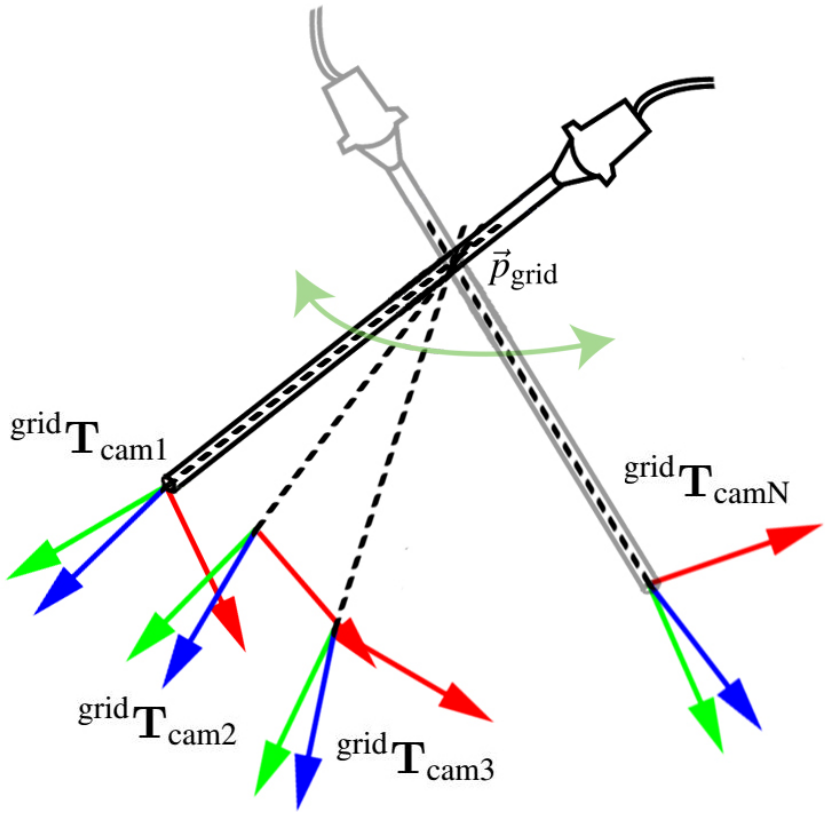


Figure 4.2: The schematic shows that the poses of the camera in each frame with respect to the calibration grid coordinate. The blue arrow of each frame denotes the z-axis. The z-axes of the camera poses intersect at the RCM and we can use this information to determine $\vec{p}_{\text{grid}}$.

of the aforementioned assumption. In a stereoscopic case, there are two sets of camera poses; one for the left camera and one for the right camera, and neither of the poses can be used directly as neither is located at the centre of the scope tip. Therefore, the average transformation between the two has to be determined to find the pose at the scope tip. The average of the two transformations is defined as the transformation that satisfies,

$$\text{logm}\left(\mathbf{M}^{\text{grid}}\mathbf{T}_{\text{camL},i}\right) + \text{logm}\left(\mathbf{M}^{\text{grid}}\mathbf{T}_{\text{camR},i}\right) = \mathbf{0}_4 \tag{4.3}$$

where $\mathbf{M}$ is defined as the average transformation and logm is the matrix logarithm function defined in Eq. 2.13. Eq. 4.3 can be solved using the Levenberg-Marquardt algorithm. It should be noted that this equation only works with a 0 degree scope.

If another type of scope such as a 30 degrees scope (Figure 1.1(b)) is used, i.e. the angle between the shaft and the z-axis is 30 degrees, the component $\vec{d}_{\text{grid}}$ must be transformed into the correct frame by rotating along the x-axis (or y-axis depending on the frame assignment) for 30 degrees before any calculation, because the orientation of the axis is changed.

Therefore, the total distance from a point $\vec{p}$ to a set of lines $(\vec{o}_{\text{grid},i}, \vec{d}_{\text{grid},i})$ can be represented as,

$$\begin{aligned} D(\vec{p}) &= \sum_{i=1}^{N} w_i ||(\vec{o}_{\text{grid},i} - \vec{p}) - ((\vec{o}_{\text{grid},i} - \vec{p})^T \vec{d}_{\text{grid},i})\vec{d}_{\text{grid},i}||^2 \\ &= \sum_{i=1}^{N} w_i (\vec{o}_{\text{grid},i} - \vec{p})^T (\mathbf{I}_3 - \vec{d}_{\text{grid},i}\vec{d}^T_{\text{grid},i})(\vec{o}_{\text{grid},i} - \vec{p}) \end{aligned} \tag{4.4}$$

The parameter w_i is introduced into the equation to penalise some distances more than others, because the estimation of the camera poses always has a re-projection error and the accuracy of the estimation for each pose is not the same. Since the algorithm relies heavily on the RCM position, assigning a confidence score on a more accurate pose is crucial to the process.

To minimise this cost function, we have to find the point $\vec{p}$ such that $\frac{\partial D}{\partial \vec{p}} = \vec{0}$. After taking the derivative and re-arranging the equation, we have,

$$[\sum_{i=1}^{N} w_i (\mathbf{I}_3 - \vec{d}_{\text{grid},i}\vec{d}^T_{\text{grid},i})]\vec{p} = \sum_{i=1}^{N} w_i (\mathbf{I}_3 - \vec{d}_{\text{grid},i}\vec{d}^T_{\text{grid},i})\vec{o}_{\text{grid},i} \tag{4.5}$$

By substituting the z-axes and the positions of the camera to $\vec{d}_{\text{grid},i}$ and $\vec{o}_{\text{grid},i}$, respectively, we can solve Eq. 4.5 for the RCM in the grid coordinate $\vec{p}_{\text{grid}}$ using the least square method.

Furthermore, because of the assumption of the intersection, the frame at the robot end-effector can be assigned such that the z-axis almost aligns with the scope as well as with the z-axis of each camera pose to simplify the hand-eye problem; Let us write R_X in the general representation of an arbitrary rotation matrix with the

angle of rotation θ and the axis of rotation $[r_x, r_y, r_z]^T, ||r_x^2 + r_y^2 + r_z^2|| = 1$ [35].

$$\mathbf{R}_X = \begin{bmatrix} \cos\theta + r_x^2 v_\theta & r_x r_y v_\theta - r_z \sin\theta & r_x r_z v_\theta + r_y \sin\theta \\ r_x r_y v_\theta + r_z \sin\theta & \cos\theta + r_y^2 v_\theta & r_y r_z v_\theta - r_x \sin\theta \\ r_x r_z v_\theta - r_y \sin\theta & r_y r_z v_\theta + r_x \sin\theta & \cos\theta + r_z^2 v_\theta \end{bmatrix} \tag{4.6}$$

where v_θ is $1 - \cos\theta$. The rough alignment of the two z-axes of the camera pose and the robot pose creates an approximation of the last row (as well as the last column) that approaches $[0, 0, 1]^T$, i.e.

$$\begin{bmatrix} r_x r_z v_\theta - r_y \sin\theta \\ r_y r_z v_\theta + r_x \sin\theta \\ \cos\theta + r_z^2 v_\theta \end{bmatrix}^T = \begin{bmatrix} r_{31} \\ r_{32} \\ r_{33} \end{bmatrix}^T \rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \tag{4.7}$$

Solving the last element yields the condition $\theta \rightarrow 0$ or $r_z \rightarrow \pm 1$. Substituting the condition shown in Eq. 4.7 into Eq. 3.2 gives

$$\begin{aligned} \mathbf{R}_A \vec{t}_X + \vec{t}_A &= \begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} t_{Bx} \\ t_{By} \\ t_{Bz} \end{bmatrix} + \vec{t}_X \\ \mathbf{R}_A \vec{t}_X + \vec{t}_A &= \begin{bmatrix} \bullet \\ \bullet \\ r_{31} t_{Bx} + r_{32} t_{By} + r_{33} t_{Bz} \end{bmatrix} + \vec{t}_X \end{aligned} \tag{4.8}$$

The $\bullet$ in Eq. 4.8 indicate that the values are not a part of this analysis. As r_{31}, r_{32} approach 0 and r_{33} approaches 1, the estimation of the translation component in the z-axis is less affected by the error from the rotation estimation. In contrast, the non-surgical robot setups may have a full six DoF problem, and the error from the rotation component subsequently worsens the translation estimation [45]. This behaviour is demonstrated in Section 4.2.3 that the translation estimation is not consistent and is less accurate in comparison to motion with the RCM.

4.2.2 Hand-eye calibration using the remote centre of motion

After computing the RCM position $\vec{p}_{\text{grid}}$, we can substitute into Eq. 4.1 and multiply the transformations ${}^{\text{cam},i}\mathbf{T}_{\text{grid}}$ and ${}^{\text{robot},i}\mathbf{T}_{\text{base}}$ to the points $\vec{p}_{\text{grid}}$ and $\vec{p}_{\text{base}}$ as follows,

$$\begin{bmatrix} \vec{p}_{\text{cam},i} \\ 1 \end{bmatrix} = \mathbf{X} \begin{bmatrix} \vec{p}_{\text{robot},i} \\ 1 \end{bmatrix} \tag{4.9}$$

In the noise-free setup, there is no difference between the points $\{\vec{p}_{\text{cam},1}, \vec{p}_{\text{cam},2}, \ldots, \vec{p}_{\text{cam},N}\}$, because the RCM and the camera frames are both located on a scope which is a rigid body and the relative position of the two robot poses is therefore constant regardless of the camera pose. This also applies to $\{\vec{p}_{\text{robot},1}, \vec{p}_{\text{robot},2}, \ldots, \vec{p}_{\text{robot},N}\}$. Although in the real situation, each pair of $(\vec{p}_{\text{cam},i}, \vec{p}_{\text{robot},i})$ is perturbed by the error in the camera calibration and forward kinematics, respectively, which results in the two distributed point clouds containing the noise characteristic from the mentioned methods, the two sets cannot be considered as two concentrated point clouds. Therefore, to model this constraint, we have to formulate Eq. 4.9 in a linear equation and average them for every pose,

$$\begin{aligned} \frac{1}{N}\sum_{i=1}^{N} \vec{p}_{\text{cam},i} &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{R}_X \vec{p}_{\text{robot},i} + \frac{1}{N}\sum_{i=1}^{N} \vec{t}_X \\ \vec{p}_{\text{cam}} &= \mathbf{R}_X \vec{p}_{\text{robot}} + \vec{t}_X \\ \vec{t}_X &= \vec{p}_{\text{cam}} - \mathbf{R}_X \vec{p}_{\text{robot}} \end{aligned} \tag{4.10}$$

Then, we can substitute the value of $\vec{t}_X$ into Eq. 3.2.

$$\begin{aligned} \mathbf{R}_A \vec{p}_{\text{cam}} - \mathbf{R}_A \mathbf{R}_X \vec{p}_{\text{robot}} + \vec{t}_A &= \mathbf{R}_X \vec{t}_B + \vec{p}_{\text{cam}} - \mathbf{R}_X \vec{p}_{\text{robot}} \\ \mathbf{R}_A \vec{p}_{\text{cam}} + \vec{t}_A - \vec{p}_{\text{cam}} &= \mathbf{R}_X \vec{t}_B - \mathbf{R}_X \vec{p}_{\text{robot}} + \mathbf{R}_A \mathbf{R}_X \vec{p}_{\text{robot}} \end{aligned} \tag{4.11}$$

The only unknown left in the translation equation is R_X which can be solved using the Levenberg-Marquardt algorithm over $so(3)$ as follows,

$$\vec{\mathfrak{r}} = \arg\min_{\vec{\mathfrak{r}}} \left\| \mathbf{R}_A \vec{p}_{\text{cam}} + \vec{t}_A - \vec{p}_{\text{cam}} - \mathbf{R}_X \vec{t}_B + \mathbf{R}_X \vec{p}_{\text{robot}} - \mathbf{R}_A \mathbf{R}_X \vec{p}_{\text{robot}} \right\| \tag{4.12}$$

where $\vec{\mathfrak{r}}$ is the Rodrigues representation of the rotation matrix $\mathbf{R}_X$.

Moreover, according to the assumption in the previous section, the initial guess of vector $\vec{\mathfrak{r}}$ can be chosen such that the z-axis of ${}^{\text{base}}\mathbf{T}_{\text{cam},i}$ is parallel or anti-parallel to that of ${}^{\text{base}}\mathbf{T}_{\text{robot},i}$, i.e. $\vec{\mathfrak{r}}_{\text{init}} = [0,0,\delta]$ where δ is a small degree of rotation ($\approx 1^\circ$). The problem can then be optimised by a bounded non-linear optimisation that allows some deviation from the alignment of the two axes. In our setup, we set the threshold at 5°. Note that it is not necessarily the z-axis that is pointing out of the end-effector. This depends on the frame assignment at the end-effector which can be arbitrary [35] and results in a different initial value $\vec{\mathfrak{r}}$ and optimisation interval.

To solve for the translation component, we simply stack up the matrices from Eq. 3.2 and 4.10 and solve the equation in the least squares method,

$$\begin{bmatrix} \mathbf{R}_{A1} - \mathbf{I}_3 \\ \mathbf{R}_{A2} - \mathbf{I}_3 \\ \vdots \\ \mathbf{R}_{AN} - \mathbf{I}_3 \\ \mathbf{I}_3 \end{bmatrix} \vec{t}_X = \begin{bmatrix} \mathbf{R}_X \vec{t}_{B1} - \vec{t}_{A1} \\ \mathbf{R}_X \vec{t}_{B2} - \vec{t}_{A2} \\ \vdots \\ \mathbf{R}_X \vec{t}_{BN} - \vec{t}_{AN} \\ \vec{p}_{\text{cam}} - R_X \vec{p}_{\text{robot}} \end{bmatrix} \tag{4.13}$$

It may be suggested that we can alternatively derive Eq. 4.11 further by using Eq. 3.1 and represent the problem as the absolute orientation problem as shown in Eq. 4.14.

$$\begin{aligned} \mathbf{R}_A \vec{p}_{\text{cam}} + \vec{t}_A - \vec{p}_{\text{cam}} &= \mathbf{R}_X \vec{t}_B - \mathbf{R}_X \vec{p}_{\text{robot}} + \mathbf{R}_X \mathbf{R}_B \vec{p}_{\text{robot}} \\ \mathbf{R}_A \vec{p}_{\text{cam}} + \vec{t}_A - \vec{p}_{\text{cam}} &= \mathbf{R}_X (\mathbf{R}_B \vec{p}_{\text{robot}} + \vec{t}_B - \vec{p}_{\text{robot}}) \\ \begin{bmatrix} \mathbf{R}_A & \vec{t}_A \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} \vec{p}_{\text{cam}} \\ 1 \end{bmatrix} - \begin{bmatrix} \vec{p}_{\text{cam}} \\ 1 \end{bmatrix} &= \mathbf{R}_X \left(\begin{bmatrix} \mathbf{R}_B & \vec{t}_B \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} \vec{p}_{\text{robot}} \\ 1 \end{bmatrix} - \begin{bmatrix} \vec{p}_{\text{robot}} \\ 1 \end{bmatrix} \right) \end{aligned} \tag{4.14}$$

Since we have N sets of Eq. 4.14, we can solve the equation independently for the rotation component using the absolute orientation algorithm [116] at a smaller computational cost.

However, the equation is not stable because this equation makes use of Eq. 3.1

when changing $\mathbf{R}_A\mathbf{R}_X$ into $\mathbf{R}_X\mathbf{R}_B$ which is invalidated by the small motion range. Moreover, the terms $\mathbf{A}[\vec{p}^T_{\text{cam}}, 1]^T$ and $\mathbf{B}[\vec{p}^T_{\text{robot}}, 1]^T$ contradicts the assumption that the RCM should be constant in the "cam" and "robot" frames regardless of the pose of the camera and robot as they are all connected by a rigid body: the scope. In other words, the terms $\mathbf{A}[\vec{p}^T_{\text{cam}}, 1]^T$ and $\mathbf{B}[\vec{p}^T_{\text{robot}}, 1]^T$ express the same RCM, but changing pose i to pose j causes the differences between $\mathbf{A}[\vec{p}^T_{\text{cam}}, 1]^T - [\vec{p}^T_{\text{cam}}, 1]^T$ and $\mathbf{B}[\vec{p}^T_{\text{robot}}, 1]^T - [\vec{p}^T_{\text{robot}}, 1]^T$ to approach the zero vectors and the problem becomes ill-posed and lacks DoF as follows,

$$\begin{bmatrix} \vec{0} \\ 1 \end{bmatrix} + \begin{bmatrix} \mathcal{N}_{3\times1}(0, \sigma_A) \\ 1 \end{bmatrix} = \mathbf{X} \begin{bmatrix} \vec{0} \\ 1 \end{bmatrix} + \begin{bmatrix} \mathcal{N}_{3\times1}(0, \sigma_B) \\ 1 \end{bmatrix} \tag{4.15}$$

where $\mathcal{N}(0, \sigma)$ stands for the noise with a mean of zero and a standard deviation of σ. To solve Eq. 4.15, we have to find a 4×4 skew-symmetric matrix, consisting of the centred points in each coordinate system. The final matrix used to calculate the rotation element is similar to the form below,

$$\begin{bmatrix} 0 & -\mathcal{N}_x & -\mathcal{N}_y & -\mathcal{N}_z \\ \mathcal{N}_x & 0 & -\mathcal{N}_z & \mathcal{N}_y \\ \mathcal{N}_y & \mathcal{N}_z & 0 & -\mathcal{N}_x \\ \mathcal{N}_z & -\mathcal{N}_y & \mathcal{N}_x & 0 \end{bmatrix}$$

which consists of only values from the noise and hence the formulation of absolute orientation problem is considered to be unstable.

However, although the naive implementation of the absolute orientation does not work, this can be mitigated by incorporating the non-linear optimisation method that uses the initialisation from the classic hand-eye solution which suggests that the RCM constraint formulated in Eq. 4.14 can only be used to refine the hand-eye solution. This solution is considered as part of the validation in the next section.

4.2.3 Experiments with synthetic data

The proposed method was validated in this section by comparing its calibration performance with the existing algorithms in [44], [45] and the absolute orientation algorithm [116], noted as TSAI, IDQ and ABSOR respectively. For TSAI and IDQ, we also added the non-linear optimisation at the end of these methods to refine the calibration using dual quaternion parametrisation (Eq. 3.12). For ABSOR, we first initialised the solution using TSAI method, and solve Eq. 4.14 with the Levenberg-Marquardt algorithm.

In the literature on hand-eye calibration, validation was performed by checking how the algorithm performs under varying noise, motion range and number of motions. Since the effect of increasing the number of input motions has already been extensively described in the previous chapter (that is, it increases calibration accuracy) and Eq. 4.12 is partially derived from the hand-eye matrix, the effect on the RCM formulation is the same: i.e., a higher number of input motions improves the calibration and thus the criterion is not included in the chapter. However, it is still interesting to see the robustness of the RCM formulation and how well the algorithm performs when the motion range is extremely limited.

Similar to the results shown in Sections 3.4 and 3.5, we used Eq. 3.21-3.23 to compute the calibration error when experimenting with the synthetic data.

The simulated data was generated by creating a loop of transformations between the calibration grid, the camera frame, the robot arm, and the base frame. Unlike the free-motion case, the robot motion was not completely random; the robot was commanded to move around the pre-defined RCM in the simulation in a spiral motion as shown in Figure 4.1(b). To generate the camera motion, we made use of the spherical coordinate to create a camera pose such that the z-axis passes through the RCM. Each z-axis can be expressed in the following form,

$$ {}^{\text{grid}}\vec{z}_{\text{cam},i} = \begin{bmatrix} \sin\theta_{z,i}\cos\psi_{z,i} \\ \sin\theta_{z,i}\sin\psi_{z,i} \\ \cos\theta_{z,i} \end{bmatrix} \tag{4.16} $$

where the parameters $\theta_{z,i}$ denotes the angle between the z-axis of the pose i and the z-axis of the world frame and ψ_z denote how the axis is rotated in the XY plane. In the experiment, we denoted the maximum value of θ_z and ψ_z and generated the values randomly from 0 to the maximum value to simulate the type of motions in the RCM-constrained configuration. The direction of the y-axis was then chosen arbitrarily at random. The first two elements and the x-axis can be computed from the cross product so that the generated coordinate conforms to the right-hand rule. The calculation is shown in Eq. 4.17 and Eq. 4.18 where y_1 and y_2 were randomly generated.

$$
{}^{\text{grid}}\vec{y}_{\text{cam},i} = \begin{bmatrix} y_1 \\ y_2 \\ \frac{y_1 \sin\theta_{z,i}\cos\psi_{z,i} + y_2 \sin\theta_{z,i}\sin\psi_{z,i}}{-\cos\theta_{z,i}} \end{bmatrix} \tag{4.17}
$$

$$
{}^{\text{grid}}\vec{x}_{\text{cam},i} = {}^{\text{grid}}\vec{y}_{\text{cam},i} \times {}^{\text{grid}}\vec{z}_{\text{cam},i} \tag{4.18}
$$

We used this method to generate 200 robot motions around the RCM. The hand-eye matrix was assumed to be a 180 degrees of rotation in the x-axis and a translation along the z-axis and the grid was simulated so as to be right under the scope. The loop was completed by chaining all the transformations together. Gaussian noise was then added to the transformations before they were fed into the hand-eye algorithm. Noise was not added to the RCM position directly, as the noise already worsened the RCM position estimation from the noisy orientation.

The comparison was run between the RCM formulation, the classic hand-eye solution [44], the dual quaternion solution [45] and the absolute orientation method in different experimental setups by varying the intensity of Gaussian noise, and the input motion range. The accuracy was determined by comparing the estimated transformation with the ground truth of the hand-eye matrix. For each set of simulation parameters, the experiment was run 1000 times. All the p-values calculated from every experimental setups were less than 0.05 which suggests that the difference in the calibration performance was statistically significant.

4.2.4 Gaussian noise

The noise in this experiment was increased from 0 mm to 0.2 mm in the translation component and from 0 degree to 1 degree in rotation, while the parameters θ_z and ψ_z were both kept constant at 15 degrees. The noise added in the transformations did not only perturb the relative transformations **A** and **B**. According to Eq. 4.4 and 4.3, it also created an uncertainty in the RCM position $\vec{p}_{\text{grid}}$ as the point was calculated from the z-axis of the camera pose.

Nevertheless, Figure 4.3(a) shows that the RCM method still outperforms the other three methods even when noise is present in the motions and the RCM position. This is because that Eq. 4.12 uses the average value of the RCM projected by ${}^{\text{grid}}\mathbf{T}_{\text{cam},i}$ and ${}^{\text{robot},i}\mathbf{T}_{\text{base}}$ which can potentially eliminate the noise. If the noise in the poses is increased further, the proposed method will likely yield a higher calibration error, although this will not significantly affect the calibration performance of TSAI and IDQ because they do not depend on the RCM. However, we are not interested in the larger ranges of noise intensity because the noise characteristic of the currently available robotic system as suggested in Chapter 3, is significantly smaller in magnitude.

The result suggests that using Eq. 4.12 and 4.13 to solve the calibration problem increases calibration accuracy and is better than solving the problem using the conventional calibration method. Although ABSOR method slightly outperforms the classic solutions in estimating the translation component, the plot still shows that Eq. 4.12 is a better constraint for solving the rotation, which subsequently creates a better estimation of the translation component.

4.2.5 Motion range

The noise intensity in the motions was kept at 1 mm for translation and 0.5 degrees for rotation, but the maximum values of the parameters θ_z and ψ_z were increased from 4 degrees to 44 degrees in steps of 4 degrees, while the da Vinci data had the motion range around 20 degrees and the KUKA with the configured RCM had the motion range of 5 to 10 degrees.

As shown in Figure 4.3(b), the RCM method can estimate the hand-eye matrix

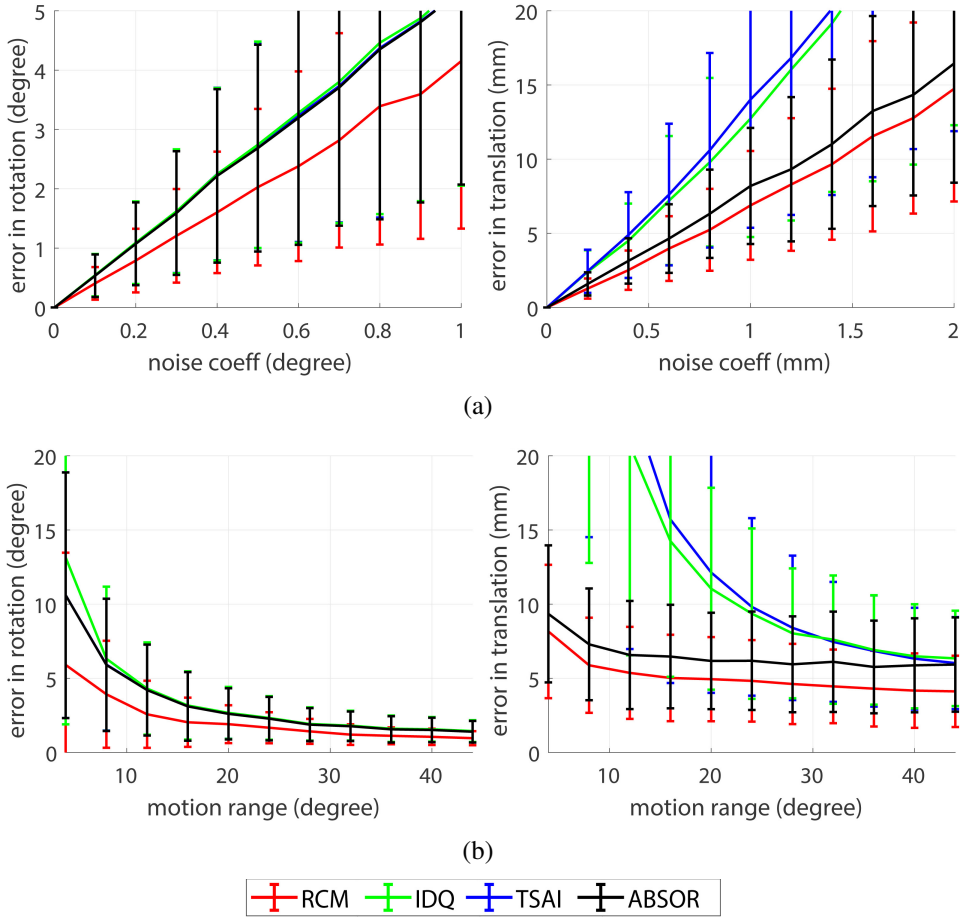


Figure 4.3: The comparison of the calibration performance with different experimental setups (a) Increasing noise in both robot and camera motions (b) Increasing motion range.

well when the motion range is around $\theta_z = \psi_z = 12$ degrees whereas the other algorithms fail in an experimental setting. The RCM method still outperforms the others even with an increasing motion range. This indicates that the RCM constraint can overcome the problem of restricted motion and is also applicable to the hand-eye calibration, regardless of the motion range.

The result also suggests that the proposed RCM method achieves accurate calibrations in the cases where the motion range is ill-posed for other classic hand-eye methods. Although the calibration error still does not satisfy calibration accuracy that medical robots can operate on [50], the results show the potential of solving the hand-eye problem in such setups in which the original methods and formulations cannot perform.

4.3 Experiments with real data

For the real data from da Vinci and KUKA, we validated the proposed method by comparing the re-projection error, the rotation error from Eq. 3.27 and the translation error from Eq. 3.28 as we now programmatically denoted the RCM for the KUKA robot.

The experiments were set up similarly to the ones in the previous chapter: we used a KUKA LBR iiwa 14 R820 mounted with a NanEye stereo camera (Figure 3.5) and a da Vinci robot with its own camera. The RCM position was then configured by the controller and the robot was commanded to move around the RCM above the calibration grid to collect several images of a checkerboard.

For both setups, we collected 39 different camera and robot poses and randomly chose 12 poses as input to each calibration method. As shown in the previous chapter, the hand-eye algorithms converge after 12 motions are used. To evaluate calibration performance, we cannot use the hand-eye equation itself (Eq. 2.17) since the rotation component of the modelled equation always holds for the case of a restricted motion range and the calibrated hand-eye matrix must satisfy the relationship between every coordinate frame in the setup, and not only for the camera and the robot arm. Therefore, we calculated the error using the three metrics defined in the previous chapter: error in the rotation (Eq. 3.27), error in the translation (Eq. 3.28), and the re-projection error. This process was repeated 200 times to obtain a distribution of the error.

4.3.1 Experiments with the KUKA LBR IIWA 14 R820

Figures 4.4(a)-4.4(c) show the calibration performance of each algorithm in terms of the rotation component, the translation component (using the RCM position), and the re-projection error. When the motion was constrained around the RCM position, the other three algorithms failed in calibrating the matrix and gave a very high error in comparison to the proposed method. The very high error in rotation shown in Figure 4.4(a) indicates that the rotation component of the estimated hand-eye matrix is not correct and cannot be used to recover the correct transformation ${}^{\text{base}}\vec{z}_{\text{grid}}$, even though the estimated rotation matrix satisfies Eq. 3.1. Although

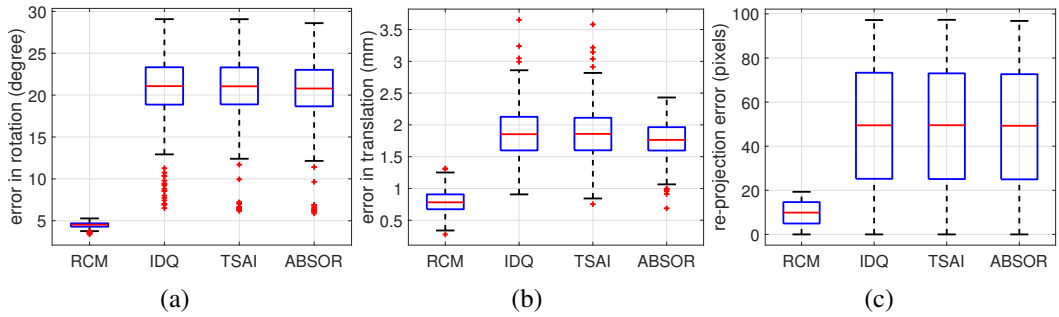
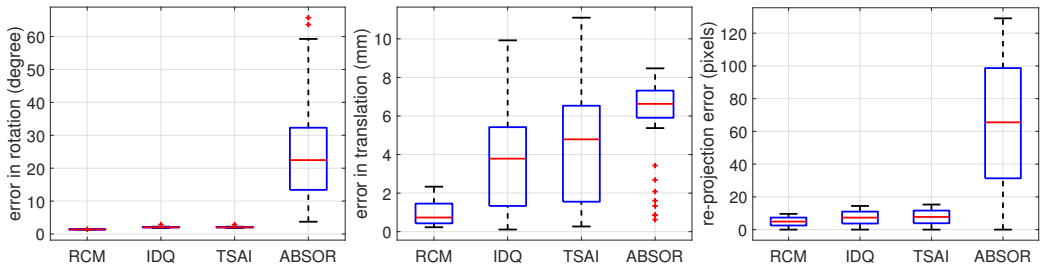


Figure 4.4: The comparison of the calibration performances from each algorithm when tested with the real data from the KUKA arm and a NanEye stereo camera. The red line represents the median of the distribution and the outliers are marked by a red cross; (a) Error in the rotation estimation (b) Error in the translation component (c) Re-projection error.

the conventional hand-eye algorithms may yield a similar rotation matrix to the RCM method in some trials, the translation component is severely worsened by the error in the rotation component which is not the case in RCM method in which the z-component is less affected.

Despite having a comparable error in the translation component, the consistency in re-projecting the RCM position using the conventional methods is not satisfactory. Figure 4.4(b) shows that the RCM position are not constant based on the calculated transformation ${}^{\text{grid}}\mathbf{T}_{\text{base}}$. This indicates that the RCM constrained motions are indeed ill-posed, producing very different transformations with accordingly lower equation residues. One of the main reasons why the conventional algorithms fail in the experiment is because the restricted rotational motion are extremely small and cannot satisfy the uniqueness criteria for solving the hand-eye problem. This confirms the deduction in Eq. 3.1 that the conventional formulation cannot be used for calibrating the hand-eye matrix in this situation.

Figure 4.4(c) shows the distribution of the re-projection error from every trial in the experiment. The re-projected grid is calculated from the intrinsic and the predicted extrinsic parameters using Eq. 3.24. Since the conventional methods cannot produce the correct rotation component of the matrix, the re-projected grid is not completely aligned with the calibration grid. On the other hand, the RCM method yields a lower re-projection error in comparison as the predicted pose is



(a) (b) (c)

Figure 4.5: The comparison of the calibration performances from each algorithm when tested with the real data from the da Vinci Standard. The red line represents the median of the distribution and the outliers are marked by a red cross. (a) Error in the rotation estimation (b) Error in the translation component (c) Re-projection error.

closer to the real camera pose.

Despite the difference in the setup of both experiments, in comparison to Figure 3.13(b) (the free-motion case), RCM still outperforms the RCM-free hand-eye calibration. The result clearly demonstrates that introducing the RCM constraint creates a simpler version of the hand-eye problem than the original one and can yield a more accurate calibration result, although the restricted motion invalidates the RCM-free formulation as demonstrated by IDQ, TSAI and ABSOR methods.

4.3.2 Experiments with the da Vinci Standard

Unlike the KUKA arm setup, the camera motion of the da Vinci Standard is always constrained by the pre-defined RCM. Hence, the outcomes of the calibration by IDQ and TSAI are similar to the ones shown in Figure 3.15. IDQ and TSAI can still perform adequately in the RCM-constrained configuration is because the motion range of the da Vinci is not sufficiently small for it to invalidate Eq. 3.1. This is not the case in the motion range of the RCM-constrained movement of the KUKA arm set in the experiment in which the uniqueness criteria could not be satisfied as shown in Figure 4.4(a).

ABSOR method, in constrast, yields very high errors in all evaluation metrics. Such errors demonstrate that the absolute orientation algorithm alone cannot solve the hand-eye problem as the RCM in Eq. 4.1 contains only one constant

point, whereas the absolute orientation algorithm requires at least three non-colinear points in each coordinate to solve the problem [116].

Similarly to Figure 3.15, despite the non-zero error, the calibration results have a low standard deviation. The RCM method produces around 1.3 degrees in the rotation component, 1.2 mm in the translation component and eight pixels in the re-projection error. We observe that while the RCM motion creates an ill-posed hand-eye problem, it can also be used as a constraint and produces a considerable improvement over the conventional formulation.

4.4 Discussion

This chapter presents a new formulation of the hand-eye problem using RCM-constrained motion. The developed formulation incorporates the pre-defined RCM position as one of the constraints to increase calibration accuracy and practical implementation in such configurations. The algorithm assumes that the z-axis of the camera poses intersect each other at the RCM, thus defining it, and uses a geometrical solution to find the RCM position which allows the construction of an absolute orientation problem with an additional constraint on the conventional general hand-eye equations.

In numerical experiments on synthetic data, the algorithm outperforms the classic hand-eye approaches according to every evaluation criterion. When working with real data captured from both the KUKA robot equipped with a NanEye stereo camera and the da Vinci Standard, the algorithm yields the lowest error with the sensible (correct orientation) hand-eye matrix while the other methods fail to calibrate the transformation. Furthermore, the plots shown in the previous section also suggest that the new method even outperforms the RCM-free setup according to every evaluation criterion.

Despite the more accurate calibration result, calibration error still exists because of the error inherent in robot and camera motions. According to the proof in [44], the influence of noise on the calibrated hand-eye matrix can be reduced by a factor of θ_{ij} in the rotation component and $2\sin\frac{\theta_{ij}}{2}$ in the translation component,

where θ_{ij} is the angle of rotation between two measurements. Given that the rotational motion is small and that the part of the original formulation is still used in the formulation, the effect of noise cannot be eliminated completely and is propagated to the estimated hand-eye matrix as shown in the calibration result. However, the result clearly indicates that the world-grid transformation ${}^{\text{base}}\vec{z}_{\text{grid}}$ can be recovered using the calibrated hand-eye matrix which means that the RCM algorithm can yield a sensible value for the hand-eye transformation with a restricted motion range and the result agrees with experimental and with the simulated data. This agreement demonstrates that the proposed algorithm has potential for applications where practical considerations limit the calibration process and do not permit the collection of calibration data points with a wide motion range.

Although the calibration error is still high and may not yet be applicable to real systems, the result shows that introducing the RCM position is a solution for hand-eye calibration in the RMIS environment. Further, enforcing this constraint avoids the requirement for surgical robots to have non-RCM compliant motion during calibration, which is currently not available in any hospital setting. Free-motion hand-eye techniques would require the re-design of surgical robot mechanisms and result in a more complex workflow and more time-consuming calibration. Therefore, the proposed method is well-suited to robotic systems that are mechanically constrained around the RCM position, such as current keyhole surgery robotic telemanipulators. Since it has been shown that correctly modelling motion constraints improves the overall calibration performance, future directions of research, in addition to the minimal solver contribution of the RCM constraint, include investigating the feasibility of adding the constraints based on the kinematic structure of the robot, in order to simplify the hand-eye problem even further.

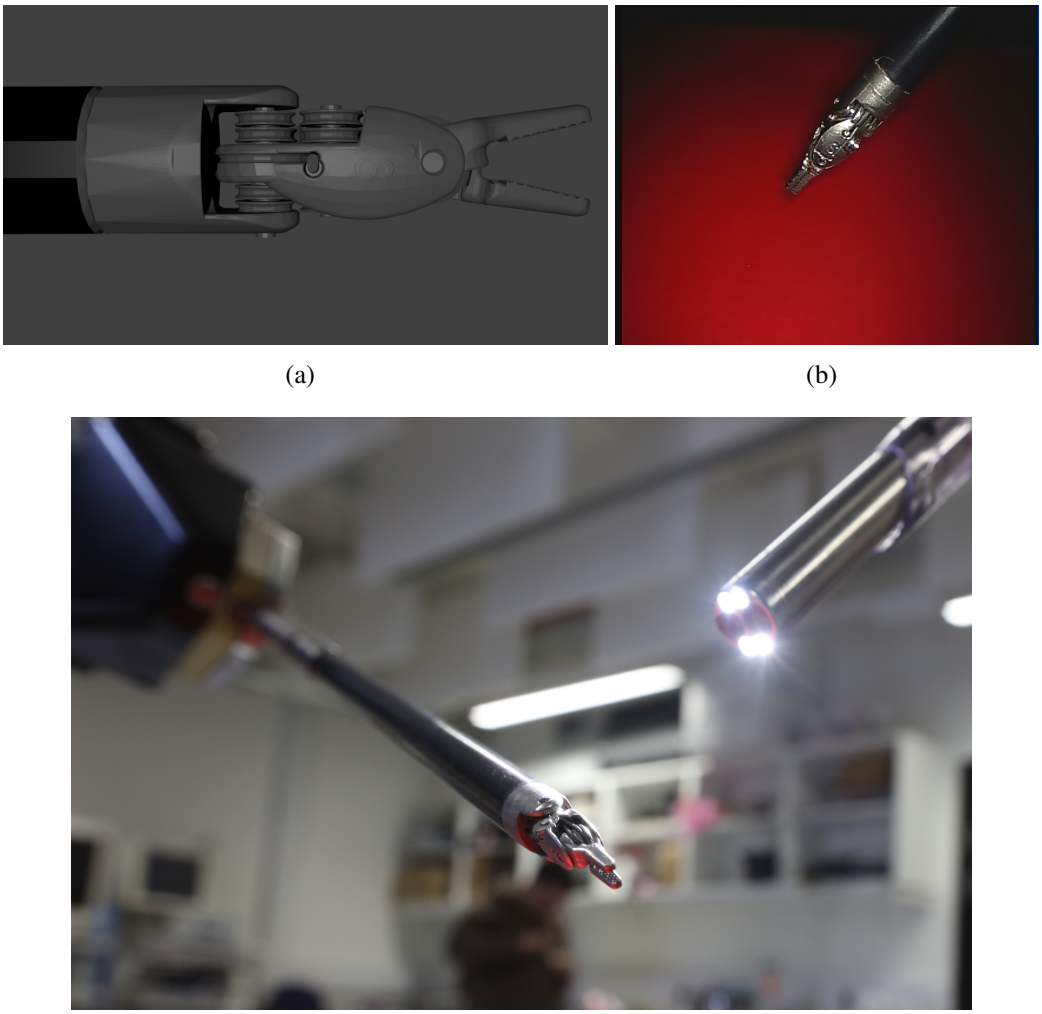
Chapter 5

Hand-eye calibration without a calibration grid

So far we have proposed modifications to the calibration pipeline aiming to tackle the problem of noisy data and input motion range. One of the problems that cannot be solved by the previously proposed algorithms is the complexity of the calibration procedure and the disruption of the surgical workflow.

According to the literature on the hand-eye problem, there are several criteria in the data collection [57] and the results from the previous chapters suggest that no algorithm yields a perfect calibration result in the RMIS environment which may present a complication in real systems. Therefore, the calibration procedure can be challenging for medical staff that are not properly trained to solve a computer vision or robotics problem. Moreover, the calibration procedure may require an additional apparatus for the calibration which needs to be sterilised beforehand. This also poses a problem in an operation as the workflow can be disrupted if there is a need for re-calibration.

One of the solutions to reduce the complexity of the calibration procedure is to use an alternative object as the calibration target that does not need re-sterilisation and that the users are already familiar with. If the geometry of the calibration target is known, the object can be used as a constraint for the calibration of the imaging sensor, even if the geometry of the target is not simple such as in needle procedures [117, 118].



(a) (b)

(c)

Figure 5.1: (a) The CAD model used in the instrument tracking algorithm. (b) An image of the instrument captured by the camera. (c) An image of the instrument moving within the field of view of the camera while it is being tracked.

In the RMIS environment, the camera and instruments are controlled and asynchronously moved around to avoid intractable mutual motion. Since the physical geometry of the instruments is known from manufacturing and computer-aided design (CAD) models (Figure 5.1(a)), the same calibration principle can be applied to the surgical instruments. This requires a surgical instrument tracking method to determine the pose of the instrument such that the 3D model of the instrument aligns with images. The tracking method typically make uses of edge features [119], color features [72, 120, 121], local gradient descriptors [122] and a combination of

shape-based features [72, 123]. Alternatively, a computer vision marker can also be printed on a surgical tool to help to localise the tool in the camera coordinate [124]. However, the pose of the tool estimated through this method may require a further calibration to ascertain the transformation between the marker and the tool.

This chapter presents the formulation for the hand-eye problem using a surgical tool as a calibration target. The method does not require an additional calibration object, which is a step towards providing CAI that automatically updates the calibration [40, 125] and can potentially yield a more accurate calibration result as a surgical instrument usually has a wider motion range than a camera's. The algorithm is validated through extensive experiments with both synthetic and real data which shows that using a surgical instrument as a calibration target has the potential to apply to a real-time application.

5.1 Hand-eye calibration using a surgical tool

5.1.1 Formulation

Recall the original equation (Eq. 2.17), the relative transformations are defined as the camera motion and the end-effector motion as shown in Eq. 2.19 and 2.20. In order to use a surgical tool as the calibration target, the calculation of the relative motions have to be re-formulated as follows,

$$\mathbf{A} = {}^{\text{psm}}\mathbf{T}_{\text{tool}}(\tau)\left({}^{\text{psm}}\mathbf{T}_{\text{tool}}(\tau')\right)^{-1} \tag{5.1}$$

$$\mathbf{B} = {}^{\text{tool}}\mathbf{T}_{\text{cam}}(\tau)\left({}^{\text{tool}}\mathbf{T}_{\text{cam}}(\tau')\right)^{-1} \tag{5.2}$$

In the normal setup, hand-eye calibration determines the unknown fixed transformation $\mathbf{X}$ that connects the robot arm coordinate frame F_{robot} and the camera coordinate frame F_{cam} (Fig. 3.1(b)). This setup, on the other hand, determines the transformation $\mathbf{X} = {}^{\text{psm}}\mathbf{T}_{\text{tool}}$ that connects the camera coordinate and the instrument coordinate. The frame "psm" and "tool" are close to each other but due to the uncertainty in the tool tracking algorithm and the difference between the frame as-

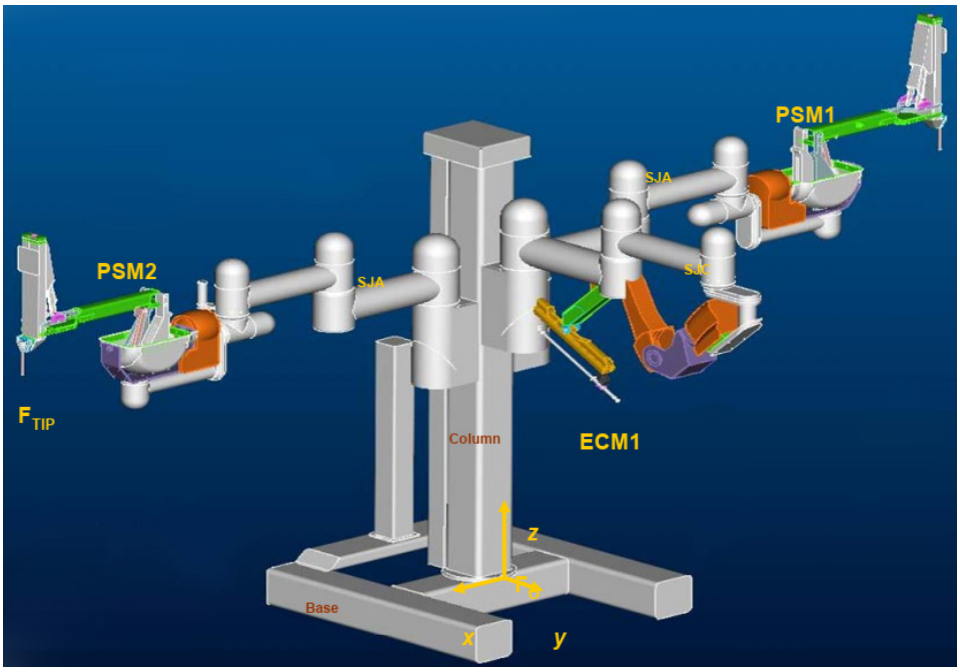


Figure 5.2: The frame assignment in the robot da Vinci standard. The ”psm” frame is assigned at F_{tip} as at the tip of the instrument assuming that the needle drive instrument is installed [50].

signments in the robot kinematics and the tool tracking, the frames are not identical. Furthermore, the “psm” frame is determined by the DH parameters in [50] which are not tool-dependent and the real frame may change with the instruments. The translation component of the calculated transformation is therefore small in magnitude and the rotation component may require 90 or 180 degrees to match the frame arrangement.

5.1.2 3D instrument tracking

Normally, in the conventional hand-eye calibration approach, the camera pose with respect to the calibration grid coordinate is estimated by solving Eq. 2.21 for a transformation ${}^{\text{cam}}\mathbf{T}_{\text{grid}}$ such that the re-projection error is minimised. While a calibration grid contains image features that are easily detected, a complex model such as a surgical instrument (Fig. 5.1(b)) is a composition of less perceptible features which makes applying the same principle to the object a highly challenging problem.

We use the instrument tracking method described in [72] to estimate the 3D

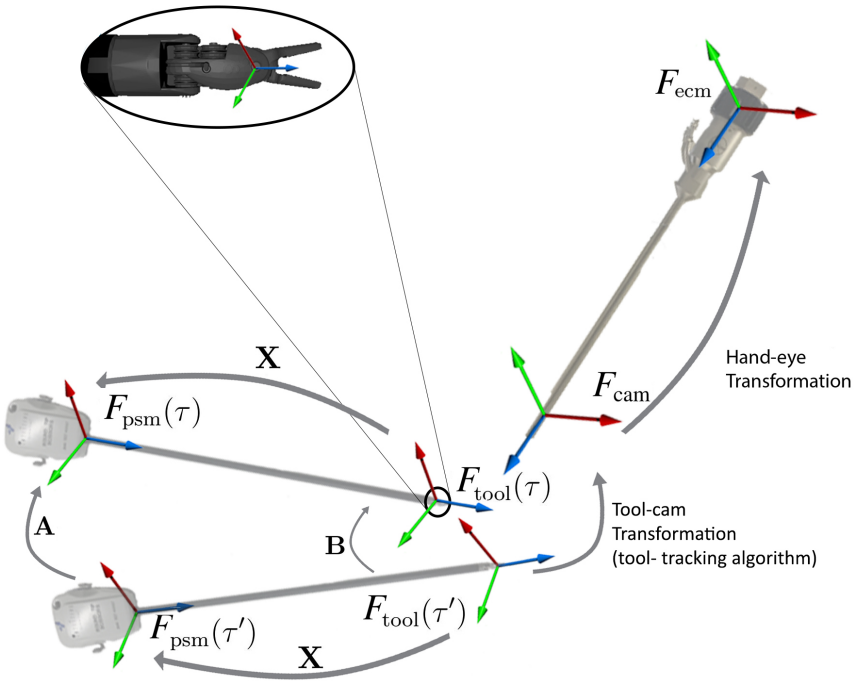


Figure 5.3: The schematic for the proposed hand-eye calibration incorporating a tool tracking algorithm [72] as mathematically represented by Eq. 5.1 and 5.2. No calibration grid is required and the camera does not move. Instead the instruments are moved by the robotic system.

pose of the surgical instrument. The algorithm solves the joint cost of aligning a CAD model of the instrument (Fig. 5.1(a)) using colour-based segmentation and local optical flow point tracking. The cost function is solved simultaneously using gradient descent across both camera in the stereo system which effectively exploits the stereo constraints and uses a linear Kalman filter for temporal consistency in frame-to-frame tracking.

5.1.3 Calibration

The hand-eye calibration approach described in Chapter 3 is used to solve the hand-eye problem once we obtain ${}^{\text{cam}}\mathbf{T}_{\text{tool}}$ and ${}^{\text{psm}}\mathbf{T}_{\text{ecm}}$ through instrument tracking and the kinematic chain, respectively. However, the estimated transformation is not the hand-eye matrix itself. Rather, it is the mapping between the tool frame and the "psm" frame (as defined by the robot kinematic chain) which is one of the missing components in the whole chain of transformations. Note that the "psm" frame

Algorithm 3 Hand-Eye Calibration without a calibration grid

```
procedure HANDEYE
    ^base T_psm,i ← forward kinematics
    A ← construct tool's relative motions
    ^tool T_cam ← tool tracking
    B ← construct tracking target's relative motions
    ^psm T_tool ← solve the hand-eye problem using Algorithm 1.
    {^ecm T_cam}_{i=1}^N ← solve Eq. 5.3 for N frames.
    X ← average transformations {^ecm T_cam}_{i=1}^N
    return ^ecm T_cam                    ▷ Hand to eye transformation
end procedure
```

is very close to the tool frame, but they are not necessarily the same because the kinematic chain described in the manual [50] only applies to certain surgical tools. Therefore, we have to estimate the true hand-eye transformation by completing the chain of transformations.

As there are typically more than two poses in the calibration process, a set of the hand-eye matrix can be calculated by,

$$^{\text{ecm}}\mathbf{T}_{\text{cam}} = (^{\text{base}}\mathbf{T}_{\text{ecm}})^{-1}(^{\text{base}}\mathbf{T}_{\text{psm},i})(^{\text{psm}}\mathbf{T}_{\text{tool}})(^{\text{tool}}\mathbf{T}_{\text{cam}}) \tag{5.3}$$

Therefore, more than one pose will complete the loop. The last step in estimating the hand-eye matrix involves calculating a per-frame hand-eye matrix $^{\text{ecm}}\mathbf{T}_{\text{cam}}$ before averaging the transformations across each frame by finding the transformation that satisfies Eq. 3.26.

The average of the transformations is calculated in the Lie algebra domain to preserve the geometrical property of the rotation matrices. The rotation components of $\{\mathbf{X}_i\}_{i=1}^N$ are converted to its Lie group $so(3)$ using Eq. 2.15. All the skew symmetric matrices are then averaged across the number of data points and converted back to $SE(3)$, while the translation components are averaged directly in the $SE(3)$ domain. The algorithm is summarised in Algorithm 3.

5.2 Experiments and results

This section shows the calibration performance of the hand-eye calibration algorithm without using a calibration grid. The purpose of the validation is to check the robustness and the effectiveness of the new formulation. Since no method in the literature describes hand-eye calibration working with the new formulation, and considering that we show in Chapter 3 that the algorithm can outperform the other state-of-the-art calibration methods, the comparison of the calibration performance is not shown in this chapter.

We tested the algorithm with the synthetic data to understand how robust the modelled equation is when the data are not gathered from the extrinsic parameters. Since Figure 3.4 shows that the error from the camera parameters resembles the shape of 2-D Gaussian noise and considering too that the tracking of a calibration grid and of a surgical tool rely on the camera parameters, we can infer that the error from the tool-tracking method is similar to Gaussian error. Therefore, we added noise to the synthetic image to simulate distortion. We also added noise to the kinematic data to simulate the noise in the position reading in robotic systems. For example, da Vinci uses tendon actuation for the joints, which is sensitive to noise, and the noise can be propagated to the end-effector pose through the robot kinematic chain [46]. As a result, a small discrepancy is created between a pose in the camera coordinates and an instrument pose estimated by the forward kinematic, which will, in turn, induce an error in the calibrated matrix.

The same experiment as Figure 3.10(b) with a relevant range of noise coefficient is also included in the validation process in addition to the synthetic tracking result. According to the results in the tool tracking paper [72], the recorded error fluctuates between less than 1 mm to 6 mm in the translation component, and from 3 to 20 degrees in the rotation component, due to drift in the tracking results. Since the error in the input data is significantly larger than the errors in Chapter 3 and considering that the proposed algorithm uses the tracking result as one of the main inputs, the error in calibration with tool tracking should be somewhat similar to the error with increasing noise.

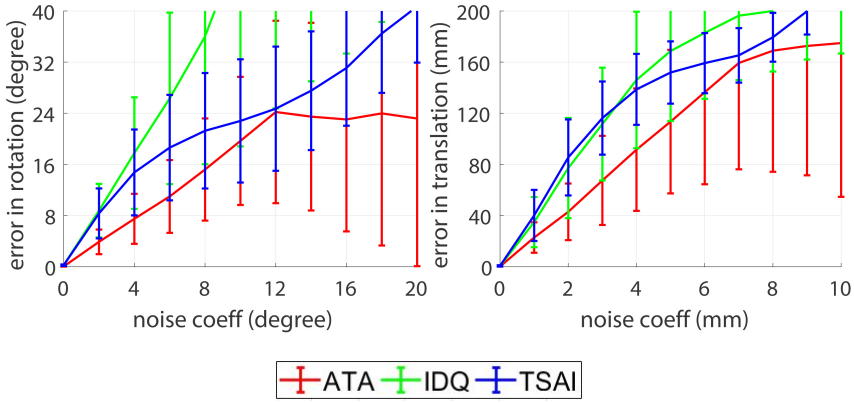


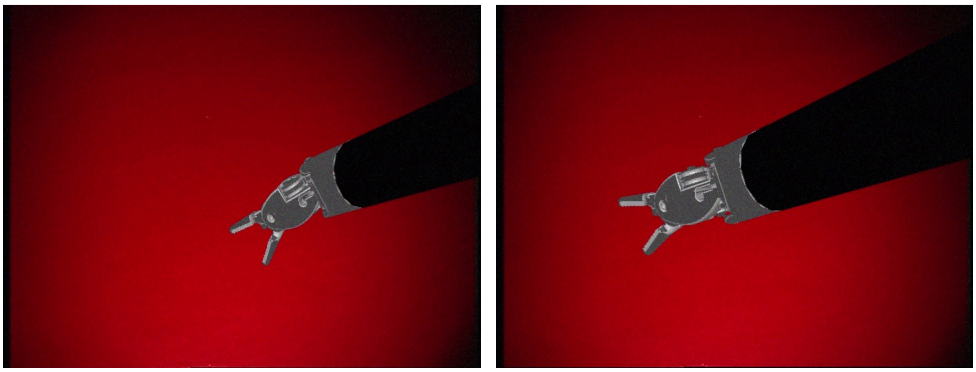
Figure 5.4: Performance of the algorithm when it is tested with synthetic data with increasing intensity of Gaussian noise in the camera parameters. This replicates the relevant range of noise in the camera data obtained from the tool tracking algorithm; the error in the rotation estimation (left) and the error in the translation estimation (right). The noise from the camera data is originated from unsuccessful tracking results in terms of its orientation which in turn creates inaccuracy in the translation component as shown in [72].

The calibration error for the experiments with synthetic data was calculated from Eq. 3.21 - 3.23. For the experiment with real data, we calibrated the hand-eye matrix using Algorithm 3 and we used data shown in Figure 3.11(c) to validate the calibrated matrix because the same camera was used and the data were collected using the same camera setup. Furthermore, we also used prior knowledge of the frame assignment and RCM position to calculate the error in the rotation (Eq. 3.27) and translation components (Eq. 3.28) which reflected the real setup.

5.2.1 Experiment with increasing Gaussian noise

The noise in this experiment was increased from 0 to 10 degrees in the rotation component and 0 to 10 mm for the translation component, whilst keeping the noise from the robot kinematic at 0.001 mm and 0.04 degrees for the translation and rotation components, respectively, to simulate the noise in the positioning system of the da Vinci Standard (see Figure 3.15).

Figure 5.4 shows that the calibration errors still increase linearly with a similar rate to the errors shown in Figure 3.10(b). With the extended range of the noise coefficient, the plots suggest that the calibration error should be in the range of 10 to 20 degrees for the rotation component and 20 to 30 mm for the translation



(a) (b)

Figure 5.5: Synthetic data generated from projecting an instrument CAD tool at poses generated from kinematic data and an artificial hand-eye transform. The images are corrupted with the Gaussian noise but the kinematic data is noise-free.

component.

5.2.2 Experiment with noise in the imaging data

Unlike the other experiments, the synthetic data in this experiment was generated by rendering a CAD model of a surgical instrument (Fig. 5.1(a)) onto a virtual camera using the OpenGL library. The robot motion was collected from the joint encoder data using the da Vinci Research Kit (dVRK) [126] in order to obtain a realistic prediction of motion for a surgical robot system. The sensible ground truth hand-eye matrix was combined with a forward kinematics to calculate the pose of the camera with respect to the robot base. Therefore, for each set of joint values, the new set of transformations could be computed and used it to render the instrument in the appropriate pose. Example images from the synthetic data are shown in Fig. 5.5(a) and 5.5(b).

In this experiment, we increased the noise intensity in the kinematic component from 0 to 2 degrees for the rotation component and from 0 to 2 mm for the translation component. The noise simulated the inaccuracy in the positioning system of the robot which can cause a greater error when using the pose estimation algorithm.

Fig. 5.6 shows the error trend when the algorithm was tested with increasing Gaussian noise. As shown in the literature, as well as in the results in Chapter 3,

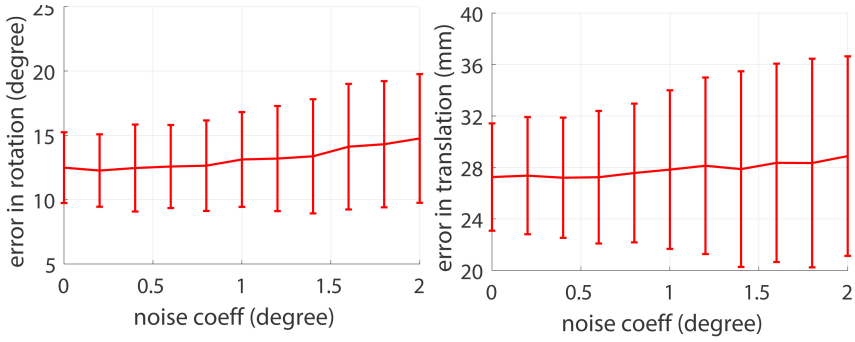
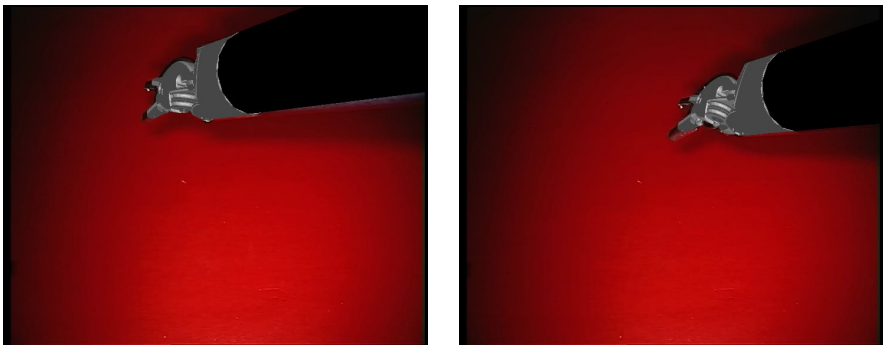


Figure 5.6: Performance of the algorithm when it is tested with synthetic data with an increasing intensity of Gaussian noise in the kinematics. This replicates the real noise that is seen when estimating poses from tendon-driven robots such as the da Vinci Standard; error in the rotation estimation (left) the error in the translation estimation (right).

the relationship between noise intensity and error in both rotation estimation and translation estimation is linear. However, the results show that the linear behaviour is not as evident as the observations in the literature, for which a calibration grid was used as the calibration target.

One of the reasons for this difference is that the noise in the kinematic parameter is significantly smaller than the noise from the tool tracking result; this makes the linear characteristic in the calibration error less evident as the noise in the kinematics is increased. However, the errors agree with the plots in Figure 5.4 such that both the errors in the rotation and translation fall in the expected ranges.

Furthermore, the tracking error in estimating ${}^{\text{tool}}\mathbf{T}_{\text{cam}}$ using the tool tracking algorithm changes the noise behaviour in the formulation significantly. The main source of error in the camera motion in the conventional method is the sub-pixel error from the camera calibration algorithm [42] whereas in this case it is from the tool tracking method. The performance of the tool tracking algorithm is sensitive to drift which results in increasingly noisy estimation of the poses and creates a larger noise in the relative transformation $\mathbf{B}$ in Eq. 5.2. The error is even clearer when there is zero noise added to the kinematic data as the algorithm fails to yield the ground truth of the hand-eye matrix with the noise-free data. This shows that the error from the tool tracking algorithm has a significant impact on the estimation of the hand-eye matrix.



(a) (b)

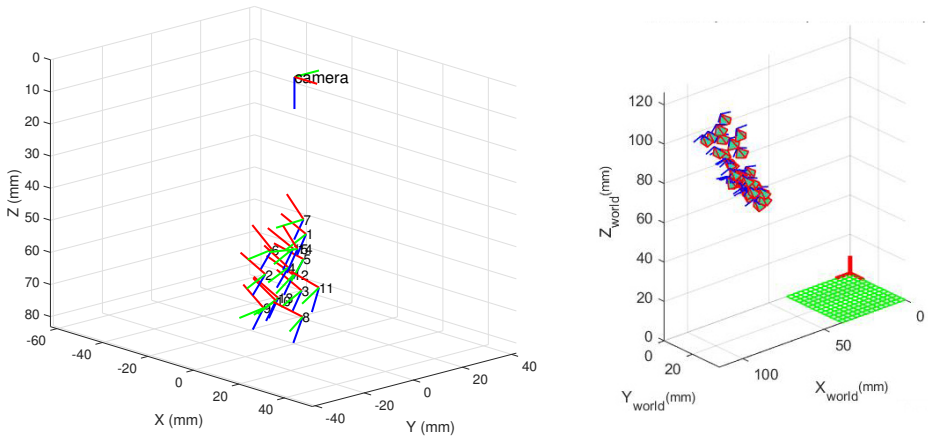
Figure 5.7: (a)-(b) shows instrument tracking result from the algorithm of [72] when it is re-projected onto images. We use the estimates from this algorithm to compute the camera data for the input of the calibration routine.

5.2.3 Experiments with real data

We performed the experiment using data from the da Vinci Standard. The camera attached to Endoscope control manipulator (ECM) was fixed while the surgical instrument controlled by Patient side manipulator (PSM) was moved within the camera frame to obtain a frame-by-frame pose of the surgical tool. The kinematic of the PSM arm is described in Table 7.4 in the Appendix section. In the same ECM setup, we also moved the camera around the calibration grid to collect several images for the purpose of validating the calibration result.

Tool tracking was then applied to determine the optimal pose of the PSM in each frame such that the CAD model can be re-projected and overlay the tool image. Note that the intrinsic parameters of the camera are pre-determined. This process resulted in the transformations in Eq. 5.2. The examples of the re-projected CAD model are shown in Figure 5.7(a) and 5.7(b). To obtain the relative transformations in Eq. 5.1, we used forward kinematics to determine the pose of the PSM with respect to the ECM by simply chaining the two transformations. After collecting the data, we applied Algorithm 3 to find the hand-eye matrix.

The experiment was run on a 2.90GHz Intel Core i9-8950HK laptop. It took the machine around 10 milliseconds to finish one calibration routine as the noise in the data is considerably higher than in the conventional data. As explained in Chapter 3, the solution is converged more slowly with the noisy data.



(a) (b)

Figure 5.8: (a) The surgical instrument poses with respect to camera. The surgical tool has a wider motion range than the camera and can therefore be used to create much more varied poses for the calibration. The tool's position ranges from $[-20,-10,45]^T$ to $[10,15,80]^T$ in mm in the camera coordinate system. (b) The camera poses with respect to a calibration grid frame [42]. Due to the RCM constraint of surgical robotic systems, the range of poses is confined within a very restricted space. The camera's position ranges from $[90,-15,130]^T$ to $[110,10,145]^T$ in mm in the calibration grid coordinate system.

Fig. 5.8(a) and 5.8(b) show that the motion range of the surgical tool is much wider than the camera. With the conventional hand-eye algorithm, the camera motion is confined to the area surrounding the RCM, which cannot offer a sufficient motion range for suppression the effect of noise. In contrast, the motion of a surgical tool is freer and can be used as an input for Eq. 2.17 in which the same property of motion range holds similarly to the conventional setup.

The distributions of the calibration errors in rotation and translation when a different number of motions were used are shown in Fig. 5.9. Although the errors are rather high, the plots demonstrate the same trend as Fig. 3.15 since it converges when more motions are included (albeit more slowly than when the conventional camera data is used). This indicates that using a surgical tool as a calibration target to approach the hand-eye problem in surgical robots shows the same characteristics as the modelled hand-eye equation and is feasible.

We also note that the CAD model is slightly misaligned with the image of the surgical tool, which severely affects the calibration error for the hand-eye matrix.

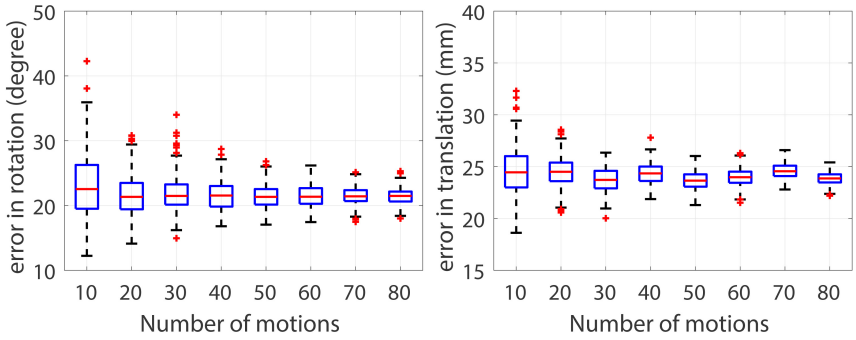


Figure 5.9: The calibration performance of the proposed hand-eye algorithm. The number on the horizontal axis represents the number of included motions in the calibration. The error in both rotation (left) and translation estimation (right) are considerably higher than the conventional method as shown in Fig. 3.15, since the error from the tool tracking algorithm is propagated to the main calibration.

Although the CAD model overlays the shaft of the tool very well, which indicates better accuracy in translation estimation, the tip of the targeted surgical tool is not completely covered. Such misalignment at the tip of the tool suggests that the orientation at the tip (the last three DoF of the PSM arm) is not properly estimated. Indeed, it yields a very high error rotation error in [72]. This error then significantly worsens the rotation estimation for the calibration result and subsequently yields an inaccurate estimation of the translation component, whereas according to the literature [44, 45], the rotation component typically has a small error.

Nevertheless, the results agree with the experiment with the synthetic data; the calibration error is around 20 degrees in the rotation component and 25 mm in the translation component, even though the noise intensity in the robot is quite small. This suggests that the factors that play an important role in the proposed algorithm are not only the image and kinematic data or the hand-eye calibration algorithm but also the accuracy of the tool tracking algorithm.

5.3 Discussion

This chapter demonstrates the potential for using a surgical instrument as the calibration target when calibrating the hand-eye matrix. Our approach tackles the hand-eye problem with a practical method of performing the calibration during robotic-assisted surgery that does not require any additional calibration objects. The

potential advantages of the approach include improvements in the ergonomics of the system, the workflow and potentially the accuracy of the calibration itself.

First, the calibration can be performed online and updated during the procedure. Typically, the calibration of any systems is an offline procedure, and has to be performed before starting the operation, and requires an update, if there is a change in the camera or surgical tool setup. This poses a problem in an operation room as surgical procedures may require time (a crucial factor in surgery) to set up the calibration environment, sterilise the calibration object and perform the calibration. Replacing the conventional calibration object with a surgical tool simplifies the procedure significantly as the calibration environment is easy to set up and a surgical tool does not need to be sterilised since it should already have been sterilised beforehand. Moreover, this results in a simpler workflow for any surgical applications that require multiple calibrations throughout a procedure.

The second advantage relates to the work surgeons have to perform. Using the conventional calibration target require more work and training in addition to a complex surgical procedure because certain steps must be followed and some criteria satisfied in order to obtain an accurate calibration. This could pose a problem for surgeons as they might not be aware of the calibration concept which could lead to an inaccurate calibration result and the need for re-calibration. This difficulty can be mitigated by utilising the object in a surgical environment and in the context of processes that surgeons are already familiar with.

Finally, the use of a surgical tool could improve calibration accuracy. As demonstrated in the literature that a wide motion range can suppress calibration error, and with a surgical tool, and the calibration problem can be modelled similarly to the classic hand-eye problem. The approach makes this criterion more achievable since a surgical tool has a wider motion range than a camera. Therefore, a sufficiently wide range of poses can be collected and used to obtain an accurate calibration.

However, the limitation of the proposed algorithm is shown in the results that the improvement over the conventional calibration algorithm is not evident. The

calibration algorithm relies heavily on the performance of the tool tracking. As the error from the tool tracking method is still considerably high and creates a set of noisy transformations which invalidate the hand-eye equation, the algorithm yields a very high calibration error whereas the conventional methods applied on the same robot results in far more accurate calibration results (Figure 3.15).

One possible source of the error is the misalignment of the projection of the CAD model, as the optimisation is applied on the cost function derived from the model alignment which is also a result of colour-based segmentation and local optical flow. Therefore, if the projection of the tool in the camera frame deviates from the CAD model or is not clear due to it being out of focus or distortion, it is likely that the resultant transformations will not be accurate. To mitigate this, we could add a computer vision marker or a calibration grid to the shaft [124] as they are easy to detect and it will give prior knowledge of the pose of the tool's shaft. This can simplify the optimisation problem as the pose of the shaft corresponds to most of the da Vinci's kinematic; only the last four joints correspond to the orientation, opening and closing of the tool's tip, while the others are related to the position and orientation of the shaft (see Table 7.4).

To summarise, we present a potential solution for the hand-eye problem in RMIS applications by using a surgical instrument as the calibration target. Although the final results are still in need of improvement, there is potential for the development of a calibration algorithm that uses a more robust model such that the tracking results can be utilised in the calibration routine.

Chapter 6

Data synchronisation for hand-eye calibration

Apart from the data selection criteria required to obtain an accurate hand-eye matrix explained in [41,44,57], there is one more step that is not commonly considered when it comes to solving the hand-eye problem: the synchronisation problem. The parameters τ and τ' in Eq. 2.17 indicate that the two transformations on the two sides of the equation must be extracted from synchronised data streams. However, this condition is not always fulfilled as different sensors (camera, motors position reading, or external trackers) may have different specifications (sampling rate, data loss, or activation time) which can result in a temporal signal misalignment as shown in Figure 6.1. The asynchronicity between the data streams invalidates the well-posed hand-eye formulation, i.e. Eq. 2.17 becomes invalid if the data streams are not fully synchronised. Therefore, most hand-eye approaches in the literature cannot be applied here as they all require a full synchronisation between the two data streams.

It can be argued that the issue can be simply solved by discretely capturing sets of images and corresponding kinematic data, i.e. users could stop the motion, capture the data and repeat until the sufficient amount of data is collected for the calibration. In practice, this strategy should eliminate the necessity of any synchronisation algorithms for the calibration since the temporal misalignment does not affect the data. However, although the hand-eye problem is solved, the transfor-

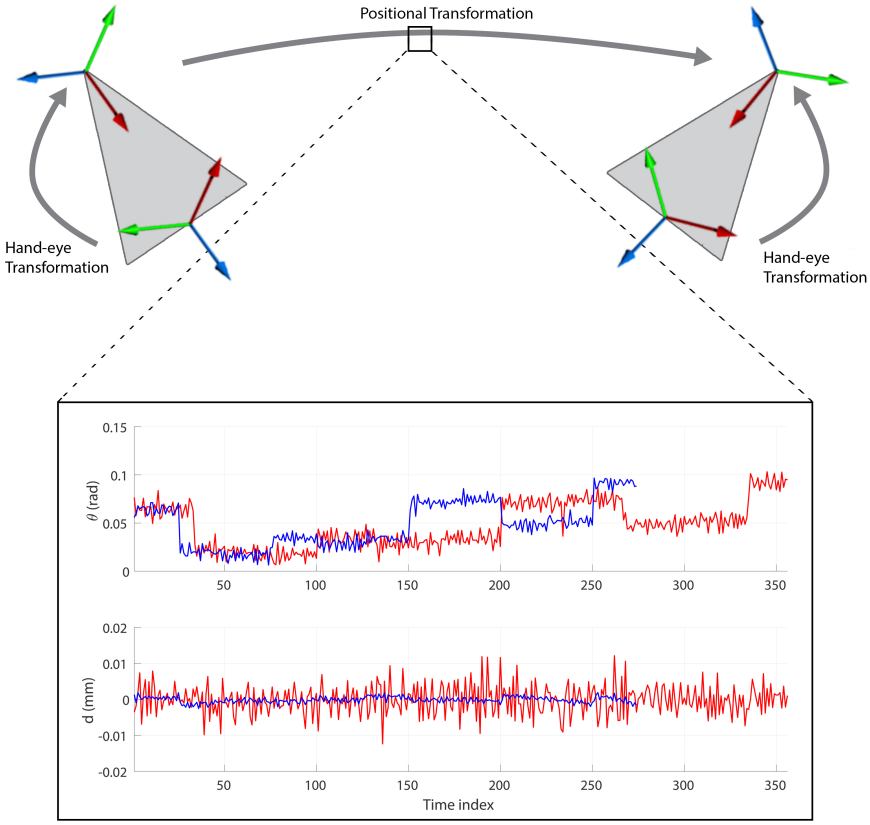


Figure 6.1: An example of data capture with different types of sensors. One is the camera and the other is a set of markers detected by a motion tracking system rigidly attached to the camera. Cameras used in robotic systems usually have capture rate around 15 – 60 fps, while many tracking systems can publish data at the rate higher than 100 fps. As we use the two equipments at the same time, the two data streams will be severely asynchronous and some data will be unusable since the rate on the other end is not fast enough, e.g. the red signal and the blue signal are similar but they are asynchronous which creates differences in their shape.

mations in each coordinate frame are not yet synchronised which makes the setup impossible to employ CAIs in surgeries. Therefore, synchronising the data streams is still required as a data pre-processing step before the calibration.

To solve the asynchronicity problem, we propose a new approach to the synchronisation of the data streams for calibrating the hand-eye matrix when the two data streams have different sampling rates and activation times. The method is part of the data preprocessing steps before the main calibration routine. The method uses the cross-correlation technique to find the time delay between the two data streams and the screw theory to recover the missing transformations for the lower-sampling

rate data. The use of screw theory has several advantages:

- The higher number of motions recovered by the algorithm based on screw motion can increase calibration accuracy [41];
- The recovery of respective hand-hand transformations for a given eye-eye pair, and vice versa;
- Both rotation and translation components are taken into account in the time-delay estimation;
- The compensation for the time-delay and sampling rate both conform to the screw constraints.

The proposed method consists of data synchronisation using cross-correlation with normalisation and resampling, followed by recovering the missing data using the screw constraints. Our experiments with both synthetic data and real data demonstrate that our method outperforms the state-of-the-art probabilistic algorithm described in [61] which shows that solving the hand-eye problem using the synchronised data can achieve a more accurate calibration result than previous efforts. First, let us recall Eq. 2.19 and 2.20 and extend the equations to the two-motions case (Figure 6.2). The equations can be extended as follows,

$$\mathbf{A}_1 = {}^{\text{cam}}\mathbf{T}_{\text{grid}}(\tau)({}^{\text{cam}}\mathbf{T}_{\text{grid}}(\tau'))^{-1} \tag{6.1}$$

$$\mathbf{B}_1 = {}^{\text{robot}}\mathbf{T}_{\text{base}}(\tau)({}^{\text{robot}}\mathbf{T}_{\text{base}}(\tau'))^{-1} \tag{6.2}$$

$$\mathbf{A}_2 = {}^{\text{cam}}\mathbf{T}_{\text{grid}}(\tau)({}^{\text{cam}}\mathbf{T}_{\text{grid}}(\tau''))^{-1} \tag{6.3}$$

$$\mathbf{B}_2 = {}^{\text{robot}}\mathbf{T}_{\text{base}}(\tau)({}^{\text{robot}}\mathbf{T}_{\text{base}}(\tau''))^{-1} \tag{6.4}$$

where τ, τ' and τ'' are different time instances in cases in which the robot configurations are not the same. Eq. 6.1 - 6.4 are all linked by the original hand-eye equation (Eq. 2.17), $\mathbf{A}_1\mathbf{X} = \mathbf{X}\mathbf{B}_1$ and $\mathbf{A}_2\mathbf{X} = \mathbf{X}\mathbf{B}_2$; this is not the case when the two relative motions are not fully synchronised, i.e. the time parameters τ and τ' in Eq. 6.1 do not represent the same common time as in Eq. 6.2.

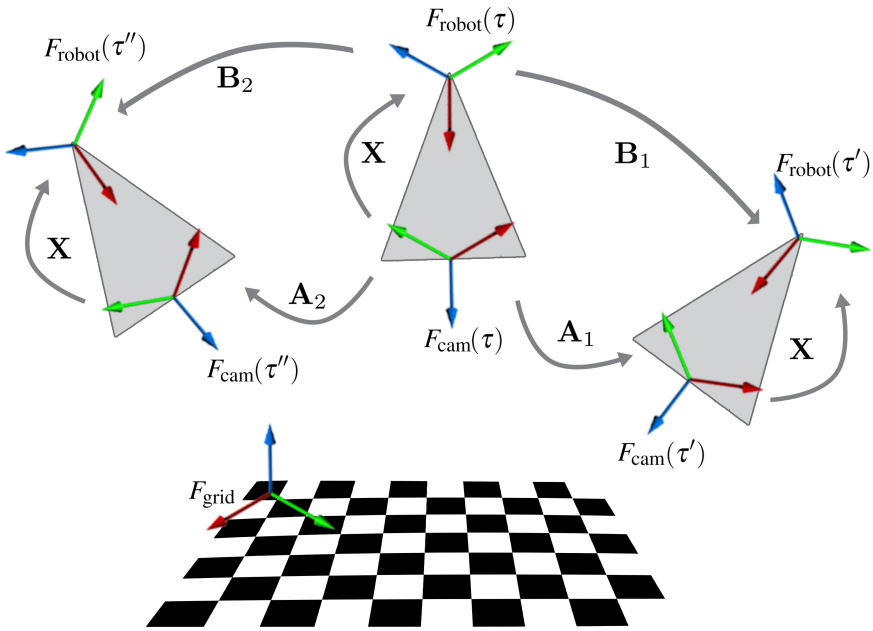


Figure 6.2: The schematic shows the transformation loop for hand-eye calibration as shown in Eq. 2.17. The camera is rigidly mounted onto the robot and moved to different poses to capture images of the calibration object corresponding to each robot pose.

6.1 Data synchronisation using cross-correlation

In the hand-eye problem (Eq. 2.17), since the two coordinate systems are rigidly connected to each other, the relative motions **A** and **B** have constraints on rotation and translation components that are independent of the calibration parameters.

$$\theta_A = \theta_B \tag{6.5}$$

$$d_A = d_B \tag{6.6}$$

where θ_A, θ_B are the angles of rotation in both reference frames, and d_A, d_B denote the amount of translation along the axes of rotation. These parameters are defined by the following equations,

$$\theta = \arccos\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \tag{6.7}$$

$$d = \vec{t}^{\,T}\vec{l} \tag{6.8}$$

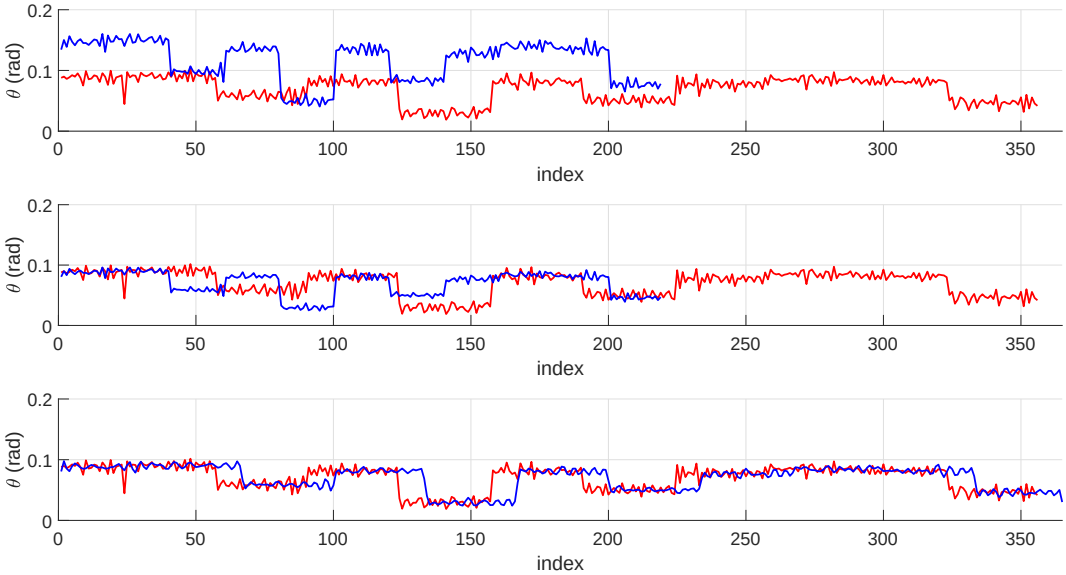


Figure 6.3: Degree of rotation that occurs in each motion. The three graphs show the screw parameter in the rotation component θ_B (blue) in comparison with its corresponding signal θ_A (red) after normalising with sampling frequency and upsampling. The top graph shows the signal before normalisation and upsampling. The middle graph shows the signal after normalisation and the bottom graph shows the signal after upsampling which is the real input for the cross-correlation function.

where r_{ij} is an element in a rotation matrix R, $\vec{t}$ represents a translation vector in a transformation and $\vec{l}$ is a unit vector denoting the principal axis of rotation [35]. This geometric relationship is called the screw motion and it is invariant to any rigid transformation, and thus is the same in any reference frames. The complete proof of this relationship is outlined in [69]. In the case of a small motion range (small θ), the constraints are also valid as the screw constraint on the translation component $d = \vec{t}^T \vec{l}$ does not consider the angle of rotation.

In the eye-in-hand setup, a camera is rigidly attached to a manipulator, so every camera motion and robot motion must conform to the screw constraints. Therefore, the parameters in Eq. 6.5 and 6.6 should be somewhat similar with some uncertainty due to the noise and difference in sampling rate as can be seen in the uppermost plot in Figure 6.3.

In signal processing, the cross-correlation technique for any two signals is a commonly known method to find the point at which the similarity score between the two signals is the highest. However, to apply the cross-correlation, the data

requires normalisation and resampling so that the two signals have similar shapes and lengths.

6.1.1 Data normalisation and resampling

The relative transformations **A** and **B** as well as the screw parameters d and θ are derived from the geometric difference between two rigid transformations. Given that the two data streams are not sampled at the same rate, the calculated difference between both data streams will not be the same: e.g., if an arbitrary signal $f(t)$ is sampled at different rates a_1 and a_2, the differentiation of the two sampled signals will not be the same but rather proportional to each other. Therefore, the data has to be normalised before applying resampling and cross-correlation so that the screw parameters for both data streams carry the same weight. The normalisation can be performed as follows,

$$\widehat{\theta_B} = \frac{f_B}{f_A}\theta_B \tag{6.9}$$

$$\widehat{d_B} = \frac{f_B}{f_A}d_B \tag{6.10}$$

where f_A and f_B are the sampling rates of the data streams $\{\mathbf{A}_i\}$ and $\{\mathbf{B}_i\}$ respectively. The normalised result is shown in the middle plot in Figure 6.3 that the blue data now has the same magnitude as the red data.

The next step is to resample the data such that the blue data has the same length as the red data before applying the cross-correlation technique. In signal processing, resampling the data consists of three procedures which are: upsampling, applying a low-pass filter and downsampling [127]. To upsample the data θ_B by a factor of q, zeros are added into the signal such that,

$$\widehat{\theta_B}[n] = \begin{cases} \widehat{\theta_B}[\frac{n}{q}] & , \frac{n}{q} \in \mathbb{I}^+ \\ 0 & , \text{otherwise} \end{cases} \tag{6.11}$$

Then, an anti-aliasing finite impulse response (FIR) filter is applied to the upsampled data to smooth the original data over the added zero. Finally, the data

is downsampled by the factor p which can be performed by discarding the data when the index is not a multiple of p. The result after resampling is shown in the lowermost plot in Figure 6.3. The plot shows that the only difference between the red and blue data is the constant time-delay which is to be estimated by using the cross-correlation method.

6.1.2 Generalised cross-correlation

The generalised cross-correlation method (GCC) is a common method to find the point in time at which two signals have the highest similarity score. This works well for estimating the time delay between two signals: both for complex-valued and real signals. The method is fully detailed in [128]. Therefore, this method has gives the advantage of exploiting both screw parameters in estimating the time delay: the parameter θ is added to the real component and the translation parameter d is added to in the imaginary component. FFT pruning and the Hilbert transform are also applied here, in addition to the classic cross-correlation, to increase the precision of the estimated time delay, whereas the estimated time delay is otherwise only the multiplication of the data's sampling rate. FFT pruning and the Hilbert transform are described in detail in [129].

6.2 Data recovery using screw constraints

After compensating for the time-delay in the data streams, the misalignment between them is at a minimum which creates adequate inputs for calibrating the hand-eye matrix. However, the stream with the faster capture rate still has more motion data than the other stream: i.e., some data produced in this stream has no correspondence in the other and therefore cannot be readily used. Furthermore, according to [41], calibration accuracy can be improved with relative motions ($\mathbf{A}, \mathbf{B}$ in Eq. 2.19 and Eq. 2.20) with a wide motion range. This condition is rather difficult to meet in the RMIS setup because the readily synchronised data may not provide a sufficiently wide motion range. Therefore, we should use as many motions as possible to maximise the motion range from the captured poses which in turn maximises calibration accuracy, instead of using only the motion that has correspon-

dence which depends closely on the hardware specifications of the two systems.

Therefore, the interpolation between two sampled points is required to fill in the missing transformations. To recover the transformations, we use dual quaternions to represents the relative transformations as such representation allows the use of the screw constraint unlike other representations. We then deduce that every missing transformation must conform to the screw constraints due to the camera-robot setup. Although the simple linear interpolation method could be used to recover the missing data as currently available vision systems and sensors have a very fast capture rate which makes the motions between two sampled points very close to linear, linear interpolation does not guarantee that the recovered motions conform to the screw constraints (Eq. 6.5-6.6) which may also invalidate the original hand-eye equation.

To formulate the equations for recovering the data, we first have to convert the transformations into a dual quaternion representation [130]. For any rigid transformation $\mathbf{T} \in SE(3)$, there is a dual quaternion $\hat{\mathbf{q}} = \mathbf{q} + \varepsilon \mathbf{q}'$ such that,

$$\mathbf{q} = \begin{bmatrix} \sin\frac{\theta}{2}\vec{l} \\ \cos\frac{\theta}{2} \end{bmatrix} \quad \mathbf{q}' = \begin{bmatrix} \frac{1}{2}(\cos\frac{\theta}{2}\vec{t} + \sin\frac{\theta}{2}(\vec{t} \times \vec{l})) \\ -\frac{1}{2}d\sin\frac{\theta}{2} \end{bmatrix} \tag{6.12}$$

where the parameters are already defined in Eq. 6.7-6.8. Without loss of generality, let us define the data stream $\{\mathbf{B}_i\}$ as the lower sampling rate data, $\tau_1, \tau_2, \ldots$ as the time instances that the data streams $\{\mathbf{A}_i\}$ and $\{\mathbf{B}_i\}$ are synchronised, and τ_b as the time instance of the missing transformation. We will have

$$\hat{\mathbf{q}}_{A,i} = \hat{\mathbf{q}}_a(\tau_i)\hat{\mathbf{q}}_a(\tau_b)^{-1} \tag{6.13}$$

$$\hat{\mathbf{q}}_{B,i} = \hat{\mathbf{q}}_b(\tau_i)\hat{\mathbf{q}}_b(\tau_b)^{-1} \tag{6.14}$$

Eq. 6.13 and 6.14 show the relative transformations between arbitrary and synchronised frames i and b that are missing in the data stream $\{\mathbf{B}_i\}$ due to a lower sampling rate. According to the screw constraints, the definitions of dual quaternion (Eq. 2.8, 2.12) and the definitions of the screw parameters (Eq. 6.7, 6.8), it is clear

that the scalar components of the rotation and translation quaternions of $\hat{\mathbf{q}}_{A,i}$ and $\hat{\mathbf{q}}_{A,i}$ must be the same.

Therefore, we can equate the two scalar components of both quaternions, and simplify the problem by using matrix multiplication to multiply the quaternions [131]. Note that the scalar component of the rotation quaternion has to be squared because the sign of $\frac{\theta}{2}$ can either be positive or negative and still have the equivalent rotation as the rotation quaternion $\mathbf{q}$ and $-\mathbf{q}$ are equivalent(i.e. they represent the same rotation matrix).

Let the elements in the rotation quaternions $\mathbf{q}_b(\tau_i)$ be $[b_{xi}, b_{yi}, b_{zi}, b_{wi}]$, and the optimised rotation quaternion $\mathbf{q}_b(\tau_b)$ be $[q_x, q_y, q_z, q_w]$. The real component in Eq. 6.14 is calculated by,

$$\mathbf{q}_{B,i} = \begin{bmatrix} b_{wi} & -b_{xi} & -b_{yi} & -b_{zi} \\ b_{xi} & b_{wi} & -b_{zi} & b_{yi} \\ b_{yi} & b_{zi} & b_{wi} & -b_{xi} \\ b_{zi} & -b_{yi} & b_{xi} & b_{wi} \end{bmatrix} \begin{bmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{bmatrix} \tag{6.15}$$

The screw constraint on the rotation component creates equality between the squared of the scalar component of Eq. 6.15 and that of the data stream $\{\mathbf{A}_i\}$. Therefore, the optimised equation for the rotation component is,

$$\phi_r(\mathbf{q}) = \sum_{i=1}^{N} ||(b_{xi}q_x + b_{yi}q_y + b_{zi}q_z + b_{wi}q_w)^2 - (a_{ri})^2|| \\ \text{subject to} \sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2} = 1 \tag{6.16}$$

where a_{ri} is the scalar component of the rotation quaternion of $\mathbf{q}_{A,i}$ and the counter i runs through every synchronised time index. For the translation component, we can apply the same technique as Eq. 6.15 and derive the scalar component of the translation quaternion in Eq. 6.14 as follows,

$$q_{bi,t} = \frac{1}{2}(\vec{t}_b - \vec{t}_i)^T (\cos\frac{\theta_b}{2}\sin\frac{\theta_i}{2}\vec{l}_i - \sin\frac{\theta_b}{2}\cos\frac{\theta_i}{2}\vec{l}_b) = \frac{1}{2}(\vec{t}_b - \vec{t}_i)^T \vec{l}_{bi} \tag{6.17}$$

and arrive at the optimisation equation,

$$\phi_t(\vec{t}_b) = \sum_{i=1}^{N} ||q_{bi,t}^2 - a_{ti}^2|| \tag{6.18}$$

where $\vec{t}_b, \theta_b$ and $\vec{l}_b$ are the translation component, degree of rotation and principal axis of rotation of the recovered transformation, $\vec{t}_i, \theta_i, \vec{l}_i$ are those of the synchronised transformations and a_{ti} is the scalar component of the translation quaternion in Eq. 6.12.

Both of the derived objective functions are non-linear functions. Since non-linear functions are generally not easy to solve or may not have a solution [132], we will show that Eq. 6.16 and 6.18 have at least one minimum. Let us start with the rotation equation, Eq. 6.16 can be re-written in the form of matrix function with a quaternion input as follows,

$$\phi_r(\mathbf{q}) = \text{Trace}((\mathbf{B}_q\mathbf{q}\mathbf{q}^T\mathbf{B}_q^T - \vec{a}_q\vec{a}_q{}^T) \odot (\mathbf{B}_q\mathbf{q}\mathbf{q}^T\mathbf{B}_q^T - \vec{a}_q\vec{a}_q{}^T)) \tag{6.19}$$

where $\odot$ defines the Hadamard product (an element-wise multiplication), and B_q and $\vec{a}_q$ are defined as follows,

$$\mathbf{B}_q = \begin{bmatrix} b_{x1} & b_{y1} & b_{z1} & b_{w1} \\ b_{x2} & b_{y2} & b_{z2} & b_{w2} \\ \vdots & \vdots & \vdots & \vdots \\ b_{xN} & b_{yN} & b_{zN} & b_{wN} \end{bmatrix}, \qquad \vec{a}_q = \begin{bmatrix} a_{r1} \\ a_{r2} \\ \vdots \\ a_{rN} \end{bmatrix} \tag{6.20}$$

To find the critical point and analyse the Hessian matrix, we compute the first and second-order derivatives of $\phi_r(\mathbf{q})$.

$$\frac{\partial \phi_r}{\partial \mathbf{q}} = 4\mathbf{B}_q^T(\mathbf{I}_N \odot (\mathbf{B}_q\mathbf{q}\mathbf{q}^T\mathbf{B}_q^T - \vec{a}_q\vec{a}_q^T))\mathbf{B}_q\mathbf{q} \tag{6.21}$$

$$\frac{\partial^2 \phi_r}{\partial \mathbf{q}^2} = 4\mathbf{B}_q^T(\mathbf{I}_N \odot (3\mathbf{B}_q\mathbf{q}\mathbf{q}^T\mathbf{B}_q^T - \vec{a}_q\vec{a}_q^T))\mathbf{B}_q \tag{6.22}$$

Let us take a look at the Hessian matrix. The term $3\mathbf{B}_q\mathbf{q}\mathbf{q}^T\mathbf{B}_q^T - \vec{a}_q\vec{a}_q^T$ in Eq.

6.22 creates a condition for positive semi-definiteness of the problem which implies convexity of the problem that for each diagonal value,

$$3(b_{xi}q_x + b_{yi}q_y + b_{zi}q_z + b_{wi}q_w)^2 - (a_{ri})^2 \geq 0, \quad i = 1,2,...,N \tag{6.23}$$

The inequality can be simplified further into an inequality of the difference between two cosine functions as they both represent rotation quaternions which formulates the condition for positive semi-definiteness in $\phi_r(\mathbf{q})$ as follows,

$$\cos\frac{\theta_B(\mathbf{q})}{2} \geq \frac{1}{\sqrt{3}}\cos\frac{\theta_A}{2}, \qquad \cos\frac{\theta_B(\mathbf{q})}{2} \leq -\frac{1}{\sqrt{3}}\cos\frac{\theta_A}{2} \tag{6.24}$$

The condition derived above helps to identify whether an extremum is a maximum or a minimum. To find the extremum of the function, we have to solve the cubic function derived from the gradient $\frac{\partial\phi_r}{\partial\mathbf{q}} = \vec{0}$. The most trivial solution for the equation is $\mathbf{q} = \vec{0}$, but the zero vector does not represent a rotation quaternion and thus the zero vector is not a valid solution. Let us consider the term in the parentheses $\mathbf{B}_q\mathbf{q}\mathbf{q}^T\mathbf{B}_q^T - \vec{a}_q\vec{a}_q^T$. The term derives from the equating of the square of the two scalar values in the two quaternions, which are squared to avoid the ambiguity of the signs in the quaternion. Therefore, regardless of the combinations of signs in both quaternions and according to the screw constraints (Eq. 6.5), there exists an optimal solution $\mathbf{q}^*$ and its counterpart $-\mathbf{q}^*$ for the equation $\mathbf{B}_q\mathbf{q}\mathbf{q}^T\mathbf{B}_q^T - \vec{a}_q\vec{a}_q^T$. Substituting $\mathbf{q}^*$ into Eq. 6.22 yields,

$$\frac{\partial^2\phi_r}{\partial\mathbf{q}^2}|_{\mathbf{q}=\mathbf{q}^*,-\mathbf{q}^*} = 8\mathbf{B}_q^T(\mathbf{I}_n \odot (\mathbf{B}_q\mathbf{q}\mathbf{q}^T\mathbf{B}_q^T))\mathbf{B}_q \tag{6.25}$$

It is obvious that $\mathbf{q}^*$ and $-\mathbf{q}^*$ yield a positive semi-definite Hessian matrix as $\mathbf{q}^*$ and $-\mathbf{q}^*$ already satisfy Eq. 6.24 and for any $\vec{x} \in \mathbb{R}^n$ the inner product $\vec{x}^T\frac{\partial^2\phi_r}{\partial\mathbf{q}^2}|_{\mathbf{q}=\mathbf{q}^*,-\mathbf{q}^*}\vec{x}$ is always greater than or equal to 0. This confirms that the function is convex in this region and according to the formulation, $\mathbf{q}^*$ and $-\mathbf{q}^*$ yield the same minimum value.

For the translation function $\phi_t(\vec{t})$, the objective function can be re-formulated

into the matrix equation as well. The form is similar to that of the rotation objective function.

$$\phi_t(\vec{t}) = \text{Trace}((\mathbf{B}_{qd}\vec{t}-\vec{b})(\mathbf{B}_{qd}\vec{t}-\vec{b})^T - \vec{a}_t\vec{a}_t^T) \odot ((\mathbf{B}_{qd}\vec{t}-\vec{b})(\mathbf{B}_{qd}\vec{t}-\vec{b})^T - \vec{a}_t\vec{a}_t^T)) \tag{6.26}$$

and the parameters in the function are defined as follows,

$$\mathbf{B}_{qd} = \frac{1}{2}\begin{bmatrix} \vec{l}_{b1}^T \\ \vec{l}_{b2}^T \\ \vdots \\ \vec{l}_{bN}^T \end{bmatrix}, \qquad \vec{b} = \frac{1}{2}\begin{bmatrix} \vec{l}_{b1}^T\vec{t}_1 \\ \vec{l}_{b2}^T\vec{t}_2 \\ \vdots \\ \vec{l}_{bN}^T\vec{t}_N \end{bmatrix}, \qquad \vec{a}_t = \begin{bmatrix} a_{t1} \\ a_{t2} \\ \vdots \\ a_{tN} \end{bmatrix} \tag{6.27}$$

Similarly to Eq. 6.20, we can derive the gradient and the Hessian of the function as follows,

$$\frac{\partial \phi_t}{\partial \vec{t}} = 4\mathbf{B}_{qd}^T(\mathbf{I}_n \odot ((\mathbf{B}_{qd}\vec{t}-\vec{b})(\mathbf{B}_{qd}\vec{t}-\vec{b})^T - \vec{a}_t\vec{a}_t^T))(\mathbf{B}_{qd}\vec{t}-\vec{b}) \tag{6.28}$$

$$\frac{\partial^2 \phi_t}{\partial \vec{t}^2} = 4\mathbf{B}_{qd}^T(\mathbf{I}_n \odot (3(\mathbf{B}_{qd}\vec{t}-\vec{b})(\mathbf{B}_{qd}\vec{t}-\vec{b})^T - \vec{a}_t\vec{a}_t^T))\mathbf{B}_{qd} \tag{6.29}$$

The only difference between this equation from Eq. 6.21 is that the normality constraint does not exist on the three roots for Eq. 6.28. By applying the same principle as for the rotation function, the three roots are the solution to these linear equations,

$$\mathbf{B}_{qd}\vec{t}_1^* - \vec{b} = \vec{0} \tag{6.30}$$

$$\mathbf{B}_{qd}\vec{t}_2^* - \vec{b} = 2\vec{a}_t \tag{6.31}$$

$$-\mathbf{B}_{qd}\vec{t}_3^* + \vec{b} = 2\vec{a}_t \tag{6.32}$$

By substituting each solution to the Hessian matrix, we can see that the solution from Eq. 6.30 yields a negative diagonal value which implies that the Hessian is not positive definite and that it yields a local maximum. On the contrary, the solution from Eq. 6.31 and 6.32 gives a positive definite Hessian and the function ϕ_t at

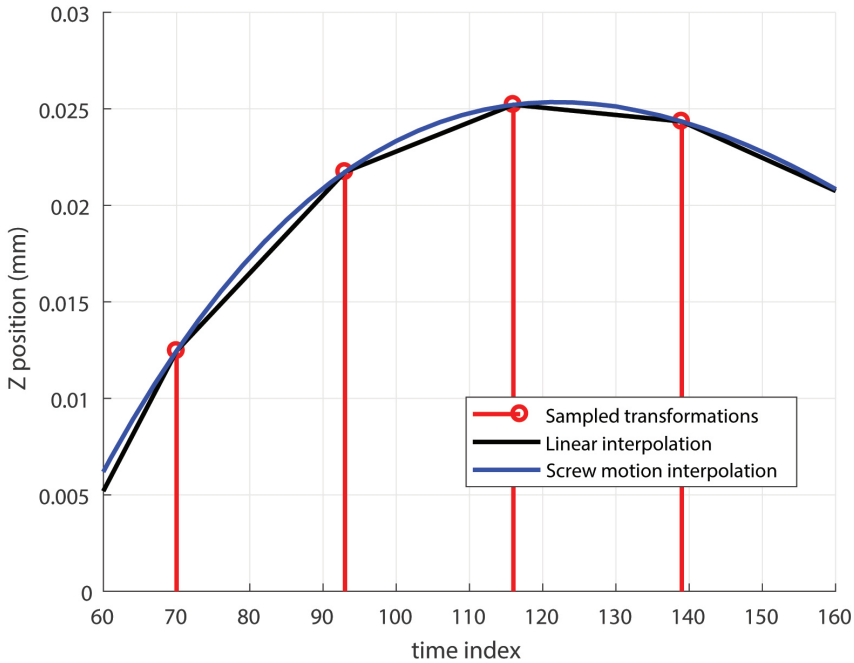


Figure 6.4: An example of the recovered transformations in between each sample. The trajectory recovered using Eq. 6.16 and 6.18 is clearly different from the linear interpolation which assumes the linearity in the trajectory between any two samples which is not always the case.

$\vec{t} = \vec{t}_2^*, \vec{t}_3^*$ gives local minima.

Therefore, it is clear that both objective functions $\phi_r(\mathbf{q})$ and $\phi_t(\vec{t})$ are convex functions in the regions of interest as their Hessian matrices are positive semi-definite, and therefore minimum values exist. Although the solutions in the analysis are derived from the linear function, the original function can be optimised by using a Lagrangian multiplier for the normality constraint in ϕ_r together with the Levenberg-Marquardt algorithm to locate the extrema.

However, the screw constraints on both rotation and translation components contain ambiguity on their signs, i.e. a degree of rotation θ around the rotation axis $\vec{l}$ is equivalent to a degree of rotation $-\theta$ around the axis $-\vec{l}$ and this implies an opposite direction for the translation component too. Therefore, the recovery transformations using Eq. 6.16 and 6.18 must be compared with the adjacent transformations to ascertain whether or not the motion is continuous. This check can be performed in the Lie Algebra domain, i.e. if the motion is not continuous in $se(3)$, the sign of the component must be changed.

Algorithm 4 Synchronising and calibrating hand-eye

1: **procedure** CHESS$({}^{\text{CAM}}\mathbf{T}_{\text{GRID}}, {}^{\text{ROBOT}}\mathbf{T}_{\text{BASE}})$
2: $\mathbf{A} \leftarrow$ construct camera's relative motions
3: $\mathbf{B} \leftarrow$ construct robot's relative motions
4: $\theta_A, \theta_B, d_A, d_B \leftarrow$ compute screw constraints
5: Estimating time-delay parameter τ and synchronise the data streams
6: Recover the lost data using Eq. 6.16 and 6.18.
7: Use Algorithm 1 or 2 to calibrate the hand-eye matrix
8: **end procedure**

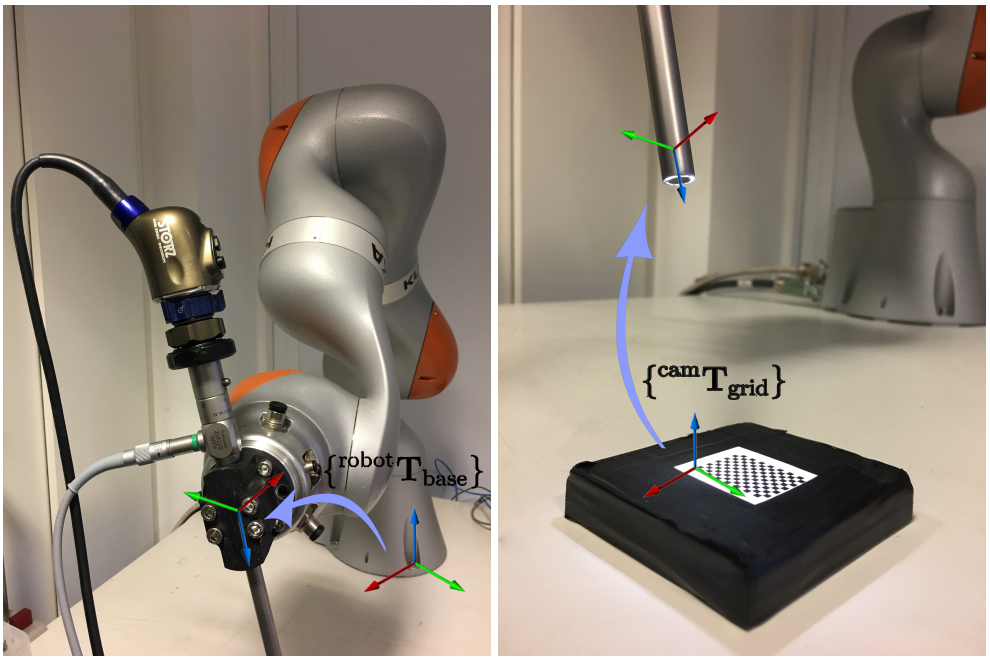
By applying the recovery method described above, an example of interpolated translation in the z-axis can be shown (Figure 6.4), together with the result from using the linear interpolation technique. The interpolated motions from the screw motions are different from the motion recovered by the linear interpolation. Although the motion between two sampled is small and the difference between screw motion and linear interpolation is also small, all the motions used in the hand-eye calibration equations must conform to the screw constraints. Otherwise, this small discrepancy can be propagated to the calibrated hand-eye matrix and yields a significant error in the estimation as shown in Figure 3.10.

For the hand-eye calibration itself, let us apply the algorithm from Chapter 3 to solve the problem as it outperforms the other methods and the translation component can be optimised using either one of the data streams. A summary of the whole process is written in Algorithm 4.

6.3 Experiments and results

This section shows the results of the experiments and calibration after preprocessing the data using the synchronisation and recovery technique explained in the previous section.

To generate the synthetic data for this section, we generated several 6×1 control points (three for the rotation component and three for the translation component) and 10 points describing a polynomial trajectory connecting each control point. This trajectory was defined as the robot's motion. We then used the method in Section 3.3 to complete the loop of transformations.



(a) (b)

Figure 6.5: The setup for the experiment with the KUKA robot. (a) Endoscope attached to the flange of the robot. (b) Endoscope capturing a video of the calibration grid. The data streams $\{^{\text{cam}}\mathbf{T}_{\text{grid}}\}$ and $\{^{\text{robot}}\mathbf{T}_{\text{base}}\}$ define the relative motions of the camera and the robot arm, respectively.

We then compared the performance of the algorithm with that of the linear interpolation and state-of-the-art probabilistic methods [61]. However, the original probabilistic method described in [61] does not include a case in which the capture rate of the two data streams are not the same. Further, the results in [58] show that the method does not work well as it requires correspondences in between both data streams. Therefore, the probabilistic algorithm used in this section is similar to Algorithm 4 and replaces the ATA method with a probabilistic method. The experimental setups are simulated by varying the intensity of Gaussian noise, the time delay and the difference in the sampling interval. For each set of parameters, the experiment was run 100 times.

For the experiment with real data, we attached an endoscope to the flange of the KUKA LBR iiwa 7 R800 and move the arm around the calibration grid to obtain the data for the calibration. The capture rate of the robot and the endoscope are 70 and 30 data points per second, respectively. Note that this capture rate may vary

in different robots, cameras and sensors. The hand motion is computed by forward kinematics and the grid detector and camera calibration algorithm in the MATLAB Camera Calibration Toolbox are used to detect the calibration grid and compute the camera pose, respectively (similar to Figure 3.2(a) and 3.2(b). The camera motions and robot motions are then used to computed the eye $\{\mathbf{A}\}$ and hand motions $\{\mathbf{B}\}$ using Eq. 6.1-6.4, respectively.

All experiments were run on a 2.9 GHz Intel Core i9-8950HK laptop. For the real data, the synchronisation and recovery steps took the processor around 760 seconds to complete the whole dataset of $7,000$ transformations (around 0.1 seconds per pose). The algorithm has a high computational cost with a long set of motions as it takes into account the transformations from the whole dataset to recover the missing poses. However, not all $7,000$ transformations were required to obtain the optimal recovered pose in each frame.

Similarly to the experiments in the previous chapter, the method in [73] was used to predict the camera's extrinsic parameters and evaluate the calibration performance of the selected hand-eye algorithm.

6.3.1 Error in time-delay estimation

First, we evaluated the performance of the time delay estimation component with varying the experimental setups; we did not consider the accuracy of the hand-eye calibration to check which factors contribute to the error in time delay estimation. The constant noise in the system is set at 0.5 degrees in the rotation component and 0.5 mm in the translation component, whereas the constant activation time is 30 seconds and the constant difference in the sampling interval is 5. These values are changed accordingly to the factor of interest in each experiment.

The result shown in Figure 6.6(a) was obtained when the noise was increased from 0 to 1 mm in the translation component and 0 to 1 degree in the rotation component with a step of 0.1. As expected, the algorithm could estimate the time delay without any error for the data streams with a low noise intensity and the error increased with increasing noise.

Figure 6.6(b) shows that the synchronisation algorithm can accurately estimate

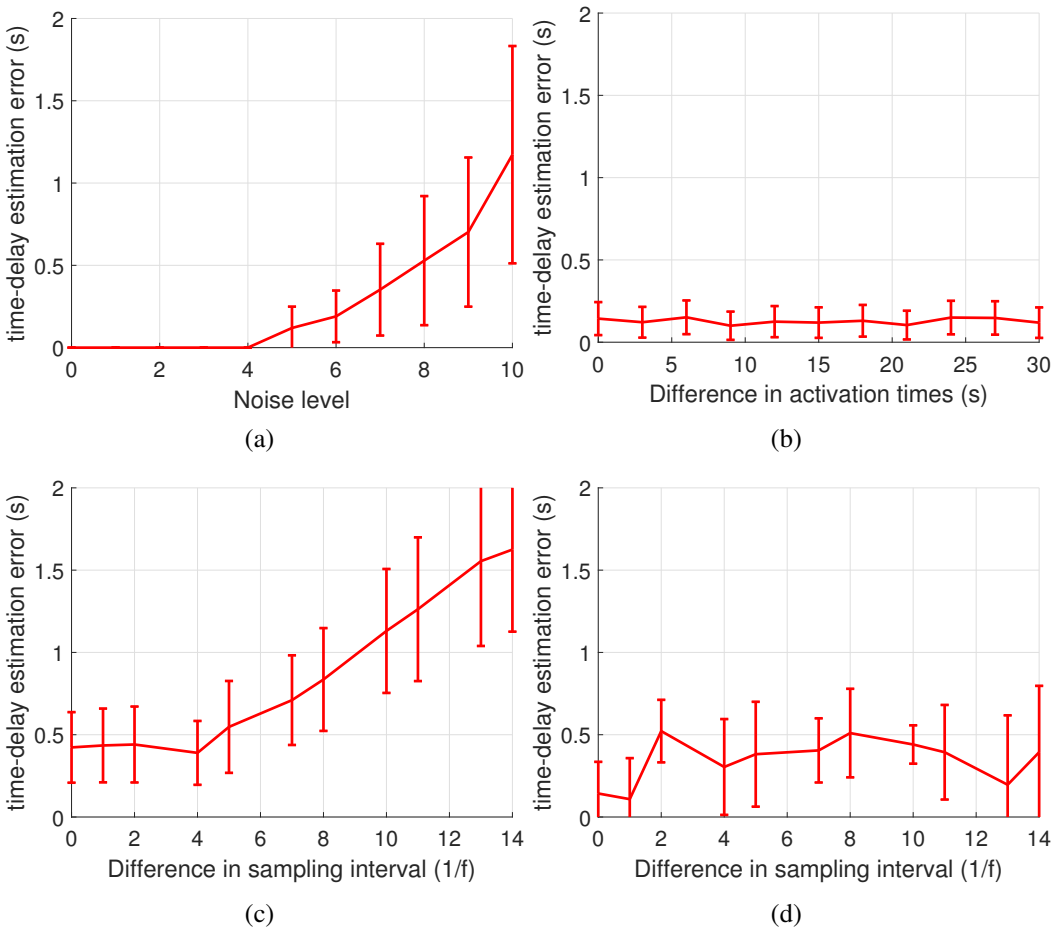


Figure 6.6: Performance of the time-delay estimation algorithm when it is tested with different experimental parameters. (a) Increasing noise in the transformations (b) Increasing in the time-delay (c) Increasing in the difference between sampling rate when noise is set at 0.5 degrees in rotation and 0.5 mm in translation (d) Increasing in the difference between sampling rate when noise is set at 0.25 degrees in rotation and 0.25 mm in translation.

the time delay regardless of its increase, albeit with a small error due to the constant noise in the data streams. This shows that the increasing time delay between each system does not affect the algorithm.

The difference in capture rates is varied to simulate a setup in which two tracking systems with different specifications are used together. Figure 6.6(c) shows increasing error in the time delay estimation. This increasing error is caused by the combination of noise and the difference in the sampling interval. According to Eq. 6.9 and 6.10, the data stream with the lower sampling rate has to be normalised before time delay estimation. However, with the noise in the system, the normalised

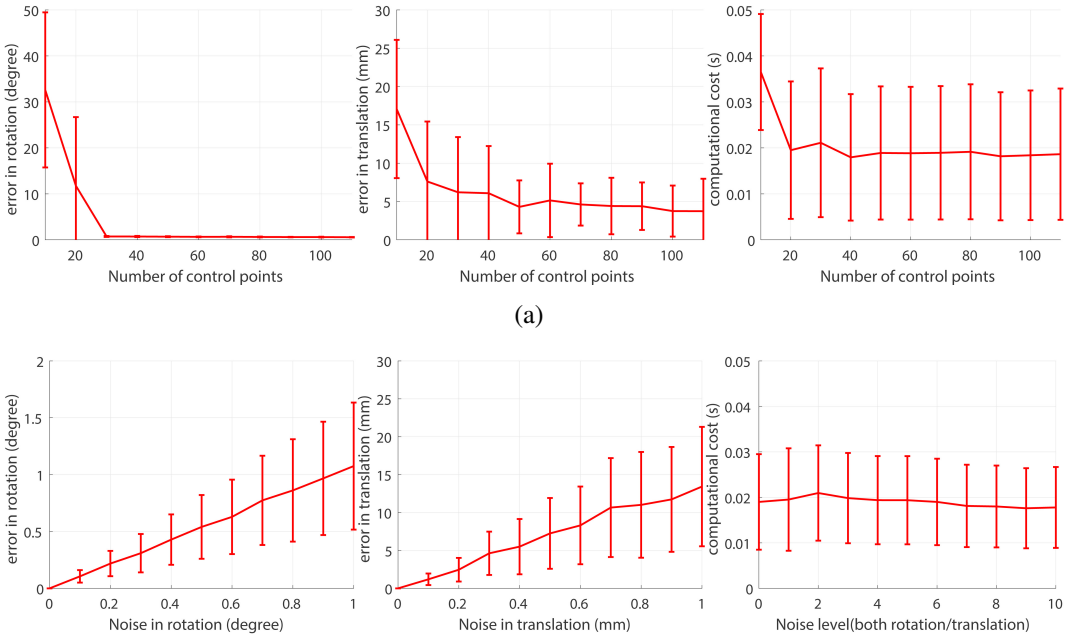


Figure 6.7: Performance of the data recovery algorithm when it is tested with different experimental parameters (a) Variation in the input motion and its computational cost (b) Increasing noise in the transformations .

data is not smooth and this creates suboptimal temporal alignment. On the contrary, the results shown in Figure 6.6(d), which simulate the time-delay estimation error when a smaller noise intensity is applied, are quite similar to those in Figure 6.6(b). The results from the two plots indicate that when the noise is small enough, the increase in the difference between two sampling intervals does not affect the time-delay estimation.

6.3.2 Error in data recovery

We evaluated the performance of the data recovery algorithm by varying the input noise intensity and the number of control points which defines the input motion range for the data recovery. The computational cost for each pose was also recorded along with the accuracy of the estimation calculated from the comparison with the ground truth.

Figure 6.7(a) shows how the variation of input motion in terms of the number of control points affects data recovery performance. The error in the recovered poses is decreased by the number of control points. This effect is similar to that

shown in the plot in Figure 3.9(b) as both are derived from the screw constraints. The computational cost is also reduced as the number of control points increase because the variety of motions can make the algorithm converge faster. However, this can also increase the cost once the convergence is reached as shown in in the plot. The results suggest that around 700 poses are sufficient to obtain the optimal recovered pose without considerably increasing the computational cost.

The plots in Figure 6.7(b) show that the error in the recovered poses increases as the noise in the transformations increases. The plots are similar to the simulation experiments in Chapter 3 as the objective functions are directly derived from the original hand-eye equation. The computational cost is not affected by noise as it is in the last plots because the noise intensity is still so small that it does not destabilise the gradient of the defined objective functions.

Table 6.1: P-value from using ANOVA on the raw results from the calibration from the synchronised and recovered data as shown in Figure 6.8. The unit of the σ_t is mm.

	$\sigma_r = 0.1^\circ$ $\sigma_t = 0.1$	$\sigma_r = 0.2^\circ$ $\sigma_t = 0.2$	$\sigma_r = 0.3^\circ$ $\sigma_t = 0.3$	$\sigma_r = 0.4^\circ$ $\sigma_t = 0.4$	$\sigma_r = 0.7^\circ$ $\sigma_t = 0.7$	$\sigma_r = 0.8^\circ$ $\sigma_t = 0.8$	$\sigma_r = 0.9^\circ$ $\sigma_t = 0.9$
R	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
$\vec{t}$	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
	$\delta t = 2.5$ s	$\delta t = 5$ s	$\delta t = 7.5$ s	$\delta t = 10$ s	$\delta t = 17.5$ s	$\delta t = 20$ s	$\delta t = 22.5$ s
R	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	0.0712
$\vec{t}$	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
	$\delta \frac{1}{f} = 1$	$\delta \frac{1}{f} = 2$	$\delta \frac{1}{f} = 3$	$\delta \frac{1}{f} = 4$	$\delta \frac{1}{f} = 7$	$\delta \frac{1}{f} = 8$	$\delta \frac{1}{f} = 9$
R	0.3592	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	0.6378
$\vec{t}$	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01

6.3.3 Increasing Gaussian noise in the transformations

The purpose of this experiment is to compare the performance of the data pre-processing algorithm when there is noise in the data streams. The noise in the translation component in this experiment was increased from 0 to 1 mm and the noise in the rotation component was increased from 0 to 1 degree, in steps of 0.1 mm and degrees, respectively. The time delay was kept constant at 30 seconds. The sampling interval of the camera motion is three samples, and the robot motion is recorded every four samples. Figure 6.8(a) demonstrates that our algorithm yields

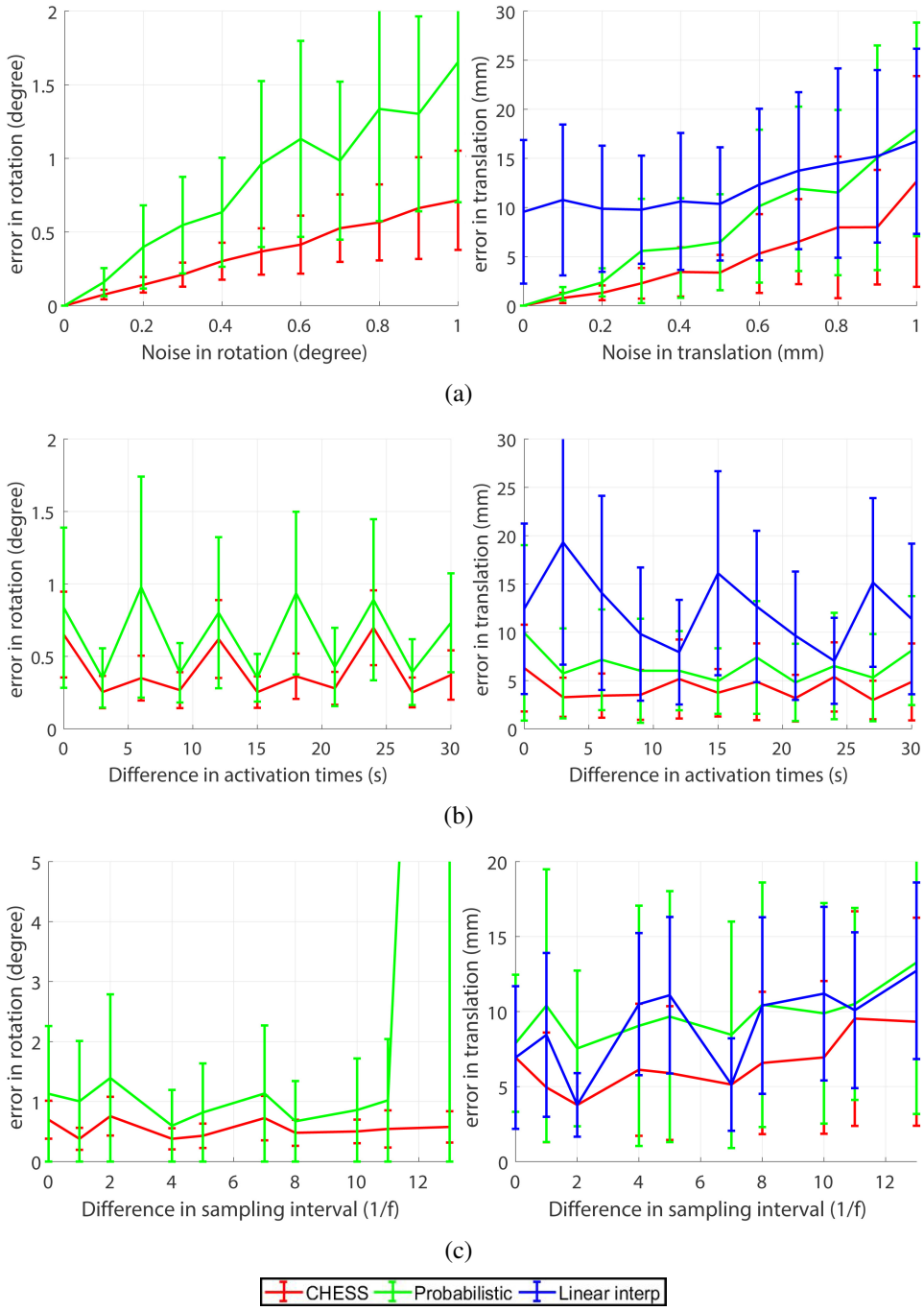


Figure 6.8: Calibration performance of each algorithm when it is tested with (a) Increasing Gaussian noise in the transformations (b) Increasing time-delay imposing on one data stream (c) Different sampling interval. In some of the plots, the blue graph is not shown because the error and the standard deviation are too high to be plotted in the same plot.

better calibration accuracy than the other methods. This not only demonstrates that the proposed method is robust to noise and can be used in the hand-eye calibration

algorithm but also that calibrating the hand-eye matrix using only the average of transformations and the covariance yields a less accurate result than the conventional formulation. The linear data recovery performs the worst among the three algorithms. Although the translational motion between poses is quite small and close to linear, this does not apply to the rotational motion. This is why the error in the rotation component is very high in comparison to the error in the translation component. This demonstrates that the data recovered by linear interpolation cannot be used in the hand-eye problem as it does not conform to the screw constraints which means that it is not valid in Eq. 2.17.

6.3.4 Increasing the time-delay between the two data streams

We varied the time delay in this experiment to see whether a larger temporal misalignment results in an inaccurate synchronisation. The noise in the translation and rotation components in this experiment was kept constant at 0.5 mm and 0.5 degrees, respectively. The sampling rates of the two data streams also remained the same as the previous experiment. Calibration accuracy is demonstrated in Figure 6.8(b) which shows that the difference in activation time does not have a significant effect on the synchronisation algorithm. Despite the added noise in the systems, the algorithm maintains its performance regardless of increasing time delay between the two data streams which confirms the hypothesis that the screw constraints are time-invariant. The p-value computed from the raw experimental result is above 0.05. This is because linear interpolation between poses yields an unstable calibration performance as the interpolated poses are not valid Eq. 2.17.

6.3.5 Increasing the difference in the sampling interval

In this experiment, we varied the sampling interval of one data stream from 3 to 16, i.e. the difference to the other sampling interval is 0 to 13, whilst noise was kept at the same level as in the previous tests and the time delay was fixed at 30 seconds. We show in Figure 6.8(c) that the proposed algorithm can estimate the missing transformations for the lower capture rate data stream and use the recovered information to estimate the hand-eye matrix. Furthermore, the calibration result is

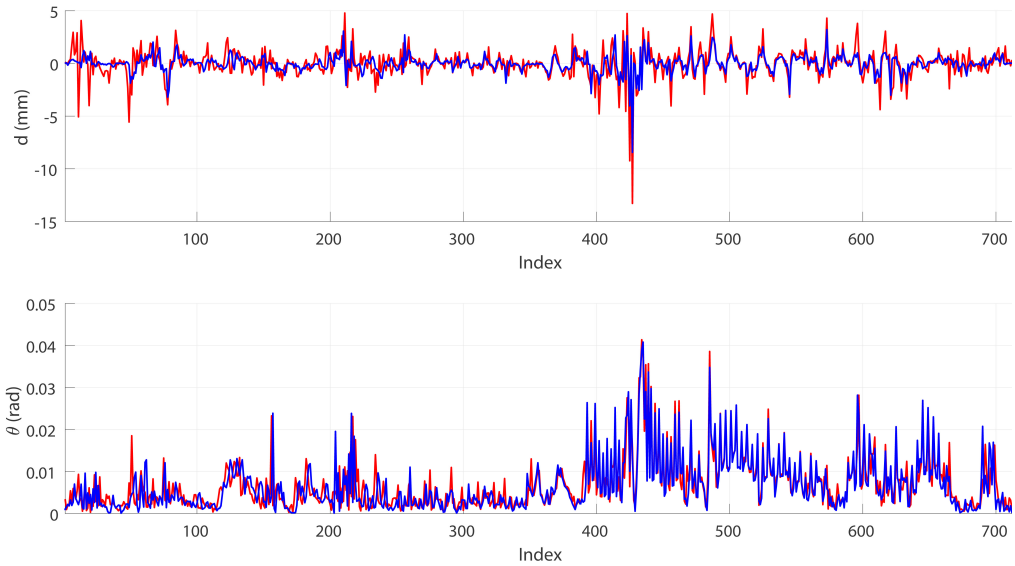
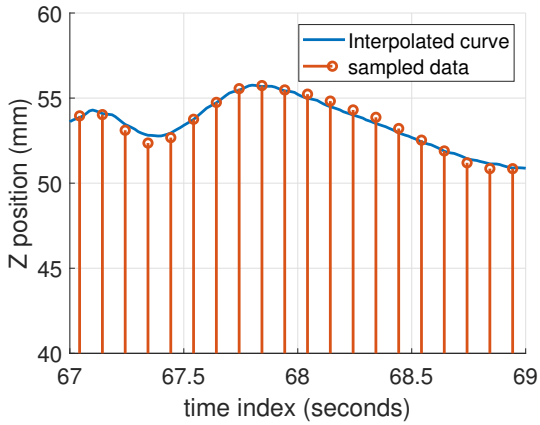


Figure 6.9: Synchronisation of the two signals after resampling and cross-correlation. The upper graph shows the screw constraint on the translation component and the bottom graph show the constraint on the rotation component.

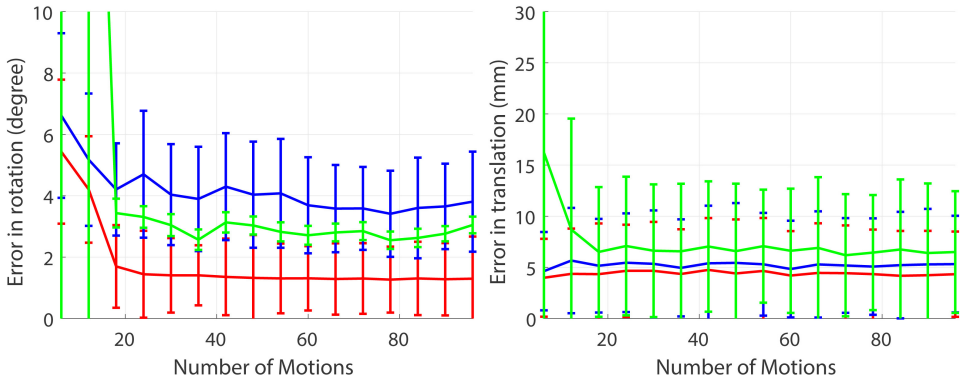
more accurate than that of the linear interpolation and probabilistic methods. The calibration error obtained from the probabilistic method increases exponentially at the end of the plot mainly because the recovered poses cannot provide an accurate estimation of the average transformation and true covariance of each data pool, due to uncertainty during the recovery process. This invalidates the hand-eye equation and the formulation that uses the Dirac delta function [64]. Some of the p-value shown in Table 6.1 are also higher than 0.05; because the tested algorithms, particularly linear and the probabilistic methods, become unstable in some experimental settings.

6.3.6 Experiments with real data

The screw constraints on both the rotation and translation components after data pre-processing are shown in Figure 6.9. The screw parameters d and θ are heavily corrupted by noise, and the patterns in the data are not clearly visible (the translation component in particular), yet the proposed algorithm can still find the point in time where the similarity score is the highest and synchronise the data streams accordingly. The synchronisation is evident over the second half of the data. Then, the data is recovered based on the screw constraints and a segment



(a)



(b)

Figure 6.10: Data recovery result and the calibration performance based on the synchronised and recovered data (a) the segment of the whole motion starting from $t = 67$ seconds to $t = 69$ seconds. The blue curve is the interpolated version of the lower sampling rate data. It interpolates between the two points that synchronise with the higher sampling rate data (b) Performance of the two algorithms when they are tested with the real data captured from the KUKA robot.

of the whole motion is shown in Figure 6.10(a). A small amount of ripples and uncertainties appearing in the recovered data stream is due to the influence of noise. Since the distance the robot arm and camera travel between two sampled points is small in magnitude, the noise in the interval can become more significant and corrupt the recovery process. However, the recovered data can still be used in the main calibration and yield satisfied results as shown in Figure 6.10(b).

Typically, calibration results start to converge when 10 motions are used in the calibration according to Figure 3.9(b), 3.13(a) and 3.15 and we may not need more than 20 motions to achieve an optimally calibrated matrix, but this does not apply

to the motion range used in the calibration. The variation in the transformations can produce a wider motion range as the recovered transformation may be able to yield a more optimal set of relative transformations.

For the hand-eye calibration performance, the errors can be computed by using Eq. 3.27 and 3.28 to validate the rotation and the translation components, respectively. As shown in the plots, the calibration performance of our algorithm is superior to the other two methods in both rotation and translation estimation, despite the convergence of the error at around 4 mm and 1.3 degrees. The results in this plot agree with those in Figure 6.8(c) that the recovered pose does not produce the correct average transformations and covariance and therefore yields a higher error than the other algorithms. On the other hand, with the synchronised and recovered data, we obtain the smallest error and the large quantity of data included in the calibration is not sufficient to deteriorate the calibration result which means that the synchronised and recovered data stream conforms to the screw constraints.

The calibration results using the linearly interpolated data are comparable to the performance of the proposed algorithm. The only difference between the two approaches is that the linear method assumes linear motion between any two consecutive poses. This produces a different result from Figure 6.8 that the linear interpolation method outperforms the probabilistic method in rotation estimation. This is simply because the sampling rate of the KUKA is very high (≈ 70 samples per second) which causes every consecutive motion to be very close to linear motion. However, the linearity of the motions cannot be safely assumed here because linearity does not imply conforming to the hand-eye screw constraints. Therefore, the poses recovered by the linear interpolation approach may not satisfy the hand-eye equation which can create the error as shown in plots. On the contrary, the proposed method interpolates the missing poses using the screw constraints and recover valid data for calibration of the hand-eye matrix which yields a superior calibration result.

6.4 Discussion

This chapter presents a data pre-processing approach to solving the asynchronicity problem for hand-eye calibration and also presents a data recovery algorithm that is able to perform online using the hand-eye screw constraints. The asynchronicity problem exists because robotic platforms usually have sensors and vision systems with different specifications to help to navigate and localise the surroundings. Therefore the data pre-processing step is crucial task to making such systems usable. The designed algorithm exploits the fact that, due to the rigid hand-eye connection, both the camera and robot motions must conform to the screw constraints. The approach consists of two main steps: data synchronisation and data recovery.

The results shown in the previous sections indicate that our method is not only robust against noise in the sensors, time delays and differences in sampling intervals but also that it can outperform the state-of-the-art probabilistic approach. The results also demonstrates that calibrating the hand-eye matrix with the complete set of synchronised data yields better calibration accuracy than working with only the average of the transformations and the covariance. Moreover, the recovered poses can be used as input for calibrating the hand-eye matrix as they all conform to the screw constraints. Therefore, the data synchronisation and recovery algorithms proposed in the chapter can increase calibration accuracy when there is asynchronicity in the input data streams. Furthermore, we also show that the recovery algorithm can be applied online to fill in the missing transformations as it can determine the missing poses at a rate of 0.01 second per pose. This ensures the validity of the data streams and introduces the potential to improve calibration accuracy even further as more motions and a wider motion range can then be included in the calibration.

Although it may be argued that we can programmatically synchronise all the data streams by using the common timestamp available in Robot Operating System (ROS), this capacity depends highly on the availability, integrality and compatibility of the hardware specifications. Our method on the other hand, is applicable to the setup and has been validated in the experiments. The findings in this chapter

creates the potential to perform further work on screw constraints and to develop the hand-eye solution specifically for the designed synchronisation algorithm. This will enable the improvement of calibration accuracy such that any tracking equipment can be used with a robotic system. For instance, an ultrasound transducer may be held by the manipulator and used in the system to track the deformable motion of an organ so the pre-operative scan can be correctly registered. Although a common timestamp can be found in the kinematic and imaging data, this is not the case for data from an ultrasound transducer.

To summarise, the presented algorithm works as a data synchronisation and recovery which is an important data pre-processing and refining step for increasing calibration accuracy of the hand-eye matrix. We demonstrate through extensive experiments that the post-processed data conform to the screw constraints and increase calibration accuracy.

Chapter 7

Conclusion and discussions

In this thesis, solutions to the hand-eye calibration formulation and pipeline are proposed along with extensive validations. The findings contribute to the literature regarding hand-eye calibration and introduce the potential to be developed further as a part of the RMIS applications. The contributions of each chapter are briefly summarised as follows.

In Chapter 3, we introduce a state-of-the-art stereoscopic formulation for the hand-eye problem and an alternating solution to the problem using the adjoint transformation. The stereoscopic formulation yields a better calibration accuracy as it contains more motion constraints than the monocular case, and the alternating solution can guarantee the global minimum due to the nature of the alternating optimisation. The proposed methods are validated by both synthetic and real data from three robots and show superiority in terms of calibration performance when noise is present. However, the methods do not solve the motion range problem in the RMIS environment.

Chapter 4 presents a solution to the motion range problem using the RCM position. While the RCM position in the robot coordinate is typically defined beforehand, the RCM position in the camera coordinate can be deduced from the camera poses. The RCM positions in two different coordinates create another constraint for the hand-eye problem that compensates for the missing DoF when the motion range is limited. Moreover, we also show that prior knowledge on the frame arrangements can simplify the problem even further in terms of the starting point of the optimi-

sation. The solution is validated using synthetic and real data from two robots and the results suggest that the RCM constraint offers better calibration accuracy than its counterpart in Chapter 3;

The potential for using a surgical instrument as the calibration target is presented in Chapter 5. The method makes use of the tool tracking algorithm to determine the camera pose and together with the kinematics of the robot data, construct the hand-eye relationship. We test the method with synthetic data and data from the da Vinci Standard and find that, although still not satisfactory due to the error and the drifting from the tool tracking algorithm, the calibration performance contains a similar noise characteristic to the normal hand-eye problem, which can be mitigated by more accurate input data. Therefore, this chapter shows the potential for using the other object as a calibration target. Moreover, further development of the algorithm may simplify the complexity of the calibration procedure and make the calibration more adaptive such that it causes less disruption to the surgical workflow;

We present a data synchronisation and recovery method based on the screw constraints in Chapter 6. Asynchronicity in the data streams for the hand-eye problem occurs more than one piece of tracking equipment is working in the systems. The method is derived from the fact that the two coordinate systems are rigidly connected and therefore always conform to the screw constraints. By using the cross-correlation method to find temporal misalignment, the synchronisation can be performed by simply compensating for the delay. Then, the data recovery can be applied by using the equality of the screw constraints in a pair of motions. We test the robustness of the synchronisation and recovery method on synthetic data and compare the calibration performance of the proposed method with the state-of-the-art probabilistic hand-eye solution. The results show that the synchronised and recovered data conform to the screw constraints and performs better than the probabilistic method.

Although the contributions in this thesis introduce potential solutions to the problems in hand-eye calibration, there are limitations in the proposed methods that

require further research. First, we will consider the calibration accuracy of a potential online hand-eye calibration. As suggested in Chapter 5, calibrating the hand-eye matrix using a surgical instrument is feasible and will simplify the calibration procedure and surgical workflow. However, the use of a surgical tool introduces more noise sensitivity and inaccuracy to the hand-eye problem due to the drifting in the tracking results and yields high calibration errors. Therefore, in order to combine the tracking technique with hand-eye calibration, we need to investigate how the error in the drift propagates to the calibrated matrix so that a more robust-to-drift formulation can be introduced to mitigate the problem accordingly. Furthermore, the hand-eye calibration algorithms could also be improved by using a non-rigid object as the calibration target. Recently developed SLAM for non-rigid objects [133] could be applied to localise an operative site and simultaneously retrieve respective camera poses which can be used to calibrate the hand-eye matrix.

Second, the conventional hand-eye formulation used throughout the thesis contains a major flaw for RMIS applications. It has been pointed out in the literature that achieving a wide motion range in order to obtain an optimal calibration accuracy is challenging in the RMIS setup. Although Chapter 4 introduces a modification to the hand-eye solution using the RCM position and the absolute orientation constraint, the achieved calibration accuracy may not be sufficient (more than 4 degrees in the rotation and high re-projection error). To solve the motion range problem in RMIS, more kinematic constraints should be added to the problem to further simplify it and compensate for the missing DoF as suggested in the formulation.

Third, the distributions of the calibration errors in Chapter 3 suggest that the main sources of error are noise and inaccuracy (an offset caused by mechanical compliance and external forces) in a positioning system. While the proposed hand-eye algorithm minimises the error from noise, the offset in robot poses is not taken into account by the formulation. Typically, the error from the offset can be mitigated by applying robot calibration, but the procedure can cause a disruption in the workflow as it requires a re-calibration for every change in a system. Therefore, a neural network method could be an alternative solution to the hand-eye problem [134–136]

as the method can generalise the relationship between the kinematic model and the camera using the difference in the relative poses detected in images and the ones calculated by the kinematic model.

In summary, this thesis proposes several modifications to the classic hand-eye problem suited to solving the current challenges in hand-eye calibration in RMIS. Although the accuracy of the potential online hand-eye calibration and a formulation requires further investigation to achieve an optimal and usable calibration algorithm in RMIS, the findings in this thesis contribute to finding an optimal solution to the hand-eye problem in robotic surgery.

Appendix

DH parameters

The DH parameters of each robot arm used in the research are listed in this section. Both of the KUKA arms use the Standard DH convention [35], while the ECM arm and the PSM1 arm of the da Vinci Standard use the modified DH convention [79]. Every robot follows the frames assignment described in the respected table except the PSM1 of the da Vinci Standard which starts after the following constant transformation,

$$\mathbf{T}_0 = \begin{bmatrix} -1 & 0 & 0 & -101.6 \\ 0 & -1 & 0 & -101.6 \\ 0 & 0 & 1 & 430 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Table 7.1: DH parameters of KUKA LBR IIWA 7 R800.

Frame	Types of joints	a_i (mm)	α_i (rad)	d_i (mm)	θ_i (rad)
1	Revolute	0	$-\frac{\pi}{2}$	190	θ_1
2	Revolute	0	$\frac{\pi}{2}$	0	θ_2
3	Revolute	0	$\frac{\pi}{2}$	400	θ_3
4	Revolute	0	$-\frac{\pi}{2}$	0	θ_4
5	Revolute	0	$-\frac{\pi}{2}$	400	θ_5
6	Revolute	0	$\frac{\pi}{2}$	0	θ_6
7	Revolute	0	0	126	θ_7

Table 7.2: DH parameters of KUKA LBR IIWA 14 R820.

Frame	Types of joints	a_i (mm)	α_i (rad)	d_i (mm)	θ_i (rad)
1	Revolute	0	$-\frac{\pi}{2}$	202.5	θ_1
2	Revolute	0	$\frac{\pi}{2}$	0	θ_2
3	Revolute	0	$\frac{\pi}{2}$	420	θ_3
4	Revolute	0	$-\frac{\pi}{2}$	0	θ_4
5	Revolute	0	$-\frac{\pi}{2}$	400	θ_5
6	Revolute	0	$\frac{\pi}{2}$	0	θ_6
7	Revolute	0	0	126	θ_7

Table 7.3: DH parameters of the ECM arm of the da Vinci Standard.

Frame	Types of joints	a_{i-1} (mm)	α_{i-1} (rad)	d_i (mm)	θ_i (rad)
0	Fixed	0	0	430	$\frac{\pi}{2}$
1	Prismatic	89.79	0	d_1	0
2	Revolute	0	0	416.6	θ_2
3	Revolute	431.8	0	142.88	θ_3
4	Revolute	431.8	0	-345.88	$\theta_4 + \frac{\pi}{2}$
5	Fixed	0	$-\frac{\pi}{4}$	0	$\frac{\pi}{2}$
6	Fixed	-66.41	0	0	0
7	Fixed	612.6	0	101.6	$-\frac{\pi}{2}$
8	Revolute	0	$\frac{\pi}{2}$	0	$\theta_8 + \frac{\pi}{2}$
9	Revolute	0	$-\frac{\pi}{2}$	0	$\theta_9 - \frac{\pi}{2}$
10	Prismatic	0	$\frac{\pi}{2}$	$-382.2 + d_{10}$	0
11	Revolute	0	0	382.8	θ_{11}
12	Fixed	0	$-\frac{\pi}{2}$	0	$-\frac{\pi}{2}$
13	Fixed	0	$-\frac{\pi}{2}$	0	$-\frac{\pi}{2}$
14	Fixed	0	$-\frac{\pi}{2}$	0	0

Table 7.4: DH parameters of the PSM1 arm of the da Vinci Standard.

Frame	Types of joints	a_{i-1} (mm)	α_{i-1} (rad)	d_i (mm)	θ_i (rad)
1	Prismatic	89.79	0	d_1	0
2	Revolute	0	0	416.6	θ_2
3	Revolute	431.8	0	142.88	θ_3
4	Revolute	431.8	0	-130.2	$\theta_4 + \frac{\pi}{2}$
5	Revolute	0	$\frac{\pi}{2}$	408.9	θ_5
6	Revolute	0	$-\frac{\pi}{2}$	-102.9	θ_6
7	Fixed	478	0	152.4	$\frac{\pi}{2}$
8	Revolute	0	$\frac{\pi}{2}$	0	$\theta_8 + \frac{\pi}{2}$
9	Revolute	0	$-\frac{\pi}{2}$	0	$\theta_9 - \frac{\pi}{2}$
10	Prismatic	0	$\frac{\pi}{2}$	$-431.8 + d_{10}$	0
11	Revolute	0	0	415.9	θ_{11}
12	Revolute	0	$-\frac{\pi}{2}$	0	$\theta_{12} - \frac{\pi}{2}$
13	Revolute	9	$-\frac{\pi}{2}$	0	$\theta_{13} - \frac{\pi}{2}$
14	Fixed	0	$-\frac{\pi}{2}$	0	0

Bibliography

[1] J. Burgner-Kahrs, D. Rucker, and H. Choset. Continuum robots for medical applications: A survey. *IEEE Robotics and Automation Letters*, 31(6):1261–1280, December 2015.

[2] L. R. Kavoussi, R. G. Moore, J. B. Adams, and A. W. Partin. Comparison of robotic versus human laparoscopic camera control. *The Journal of Urology*, 158(4):2134–2136, Dec 1995.

[3] J. Marescau, M. K. Smith, D. Fölscher, F. Jamali, B. Malassagne, and J. Leroy. Telerobotic laparoscopic cholecystectomy: initial clinical experience with 25 patients. *Annals of Surgery*, 234(1):1–7, July 2001.

[4] M. Hayashibe, N. Suzuki, M. Hashizume, Y. Kakeji, K. Konishi, S. Suzuki, and A. Hattori. Preoperative planning system for surgical robotics setup with kinematics and haptics. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 1(2):76–85, Jan 2005.

[5] R. Negenborn. *Robot Localization and Kalman Filters On finding your position in a noisy world*. PhD thesis, Utrecht University, Domplein 29, 3512 JE Utrecht, the Netherlands, 2003.

[6] A. Tewari, J. Peabody, R. Sarle, G. Balakrishnan, A. Hemal, A. Shrivastava, and M. Menon. Technique of da vinci robot-assisted anatomic radical prostatectomy. *Urology*, 60(4):569 – 572, 2002.

[7] J. Bodner, H. Wykypiel, G. Wetscher, and T. Schmid. First experiences with the da Vinci™ operating robot in thoracic surgery. *European Journal of Cardio-Thoracic Surgery*, 25(5):844–851, May 2004.

[8] Y. Y. Juo, O. Hyder, A. H. Haider, M. Camp, A. Lidor, and N. Ahuja. Is Minimally Invasive Colon Resection Better Than Traditional Approaches? First Comprehensive National Examination With Propensity Score Matching. *JAMA Surgery*, 149(2), 2014.

[9] E. Soto, Y. Lo, K. Friedman, C. Soto, F. Nezhat, L. Chuang, and H. Gretz. Total laparoscopic hysterectomy versus da vinci robotic hysterectomy: is using the robot beneficial? *Journal of Gynecologic Surgery*, 22(4), 2011.

[10] E. R. Kim, C. Lim, D. J. Kim, J. S. Kim, and K. H. Park. Robot-assisted cardiac surgery using the da vinci surgical system: A single center experience. *The Korean Journal of Thoracic and Cardiovascular Surgery*, 48(2), 2015.

[11] S. B. Bhayani. da vinci robotic partial nephrectomy for renal cell carcinoma: an atlas of the four-arm technique. *Journal of Robotic Surgery*, 1(4), 2008.

[12] A. Wedmid, E. Llukani, and D. I. Lee. Future perspectives in robotic surgery. *BJU International*, 108(6b):1028–1036, 2011.

[13] M. Diana and J. Marescaux. Robotic surgery. *British Journal of Surgery*, 102(2):15–28, 2015.

[14] O. Ukimura and I. S. Gill. Image-fusion, augmented reality, and predictive surgical navigation. *Urologic Clinics of North America*, 36(2):115 – 123, 2009. New Technology in Urologic Surgery.

[15] P. Pessaux, M. Diana, L. Soler, T. Piardi, D. Mutter, and J. Marescaux. Towards cybernetic surgery: robotic and augmented reality-assisted liver segmentectomy. *Langenbeck's Archives of Surgery*, 400(3):381–385, 2015.

[16] A. H. Hallett, E. K. Mayer, H. J. Marcus, T. P. Cundy, P. J. Pratt, A. W. Darzi, and J. A. Vale. Augmented reality partial nephrectomy: Examining the current status and future perspectives. *Urology*, 83(2):266 – 273, 2014.

[17] K. W. Sim, B. Baker, K. Amin, A. Chan, K. Patel, and J. Wong. Augmented and virtual reality in surgery—the digital surgical environment: applications, limitations and legal pitfalls. *Annals of Translational Medicine*, 4(23), 2016.

[18] P. Vávra and1 J. Roman, P. Zonča, P. Ihnát, M. Němec, J. Kumar, N. Habib, and A. El-Gendi. Recent development of augmented reality in surgery: A review. *Journal of Healthcare Engineering*, 2017:1 – 9, 2017.

[19] A. Banach, K. Leibrandt, M. Grammatikopoulou, and G. Yang. Active contraints for tool-shaft collision avoidance in minimally invasive surgery. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1556–1562, May 2019.

[20] R. Moccia, M. Selvaggio, L. Villani, B. Siciliano, and F. Ficuciello. Vision-based virtual fixtures generation for robotic-assisted polyp dissection procedures. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7934–7939, Nov 2019.

[21] S. A. Bowyer, B. L. Davies, and F. R. Baena. Active constraints/virtual fixtures: a survey. *IEEE Transactions on Robotics*, 30(1):138–157, Feb 2014.

[22] A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager. Vision-assisted control for manipulation using virtual fixtures. *IEEE Transactions on Robotics*, 20(6):953–966, Dec 2004.

[23] A. M. Okamura. Haptic feedback in robot-assisted minimally invasive surgery. *Current Opinion in Urology*, 19(1):102–107, Jan 2009.

[24] H. Xin, J. S. Zelek, and H. Carnahan. Laparoscopic surgery, perceptual limitations and force: A review. In *First Canadian Student Conference on Biomedical Computing*, 2006.

[25] A. Hernansanz, D. Zerbato, L. Gasperotti, M. Scandola, P. Fiorini, and A. Casals. Improving the development of surgical skills with virtual fixtures in simulation. In *2012 Information Processing in Computer-Assisted Interventions (IPCAI)*, pages 157–166, 2012.

[26] F. Rydén and H. J. Chizeck. Forbidden-region virtual fixtures from streaming point clouds: Remotely touching and protecting a beating heart. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3308–3313, Oct 2012.

[27] M. Li, M. Ishii, and R. H. Taylor. Spatial motion constraints using virtual fixtures generated by anatomy. *IEEE Transactions on Robotics*, 23(1):4–19, Feb 2007.

[28] A. Ruszkowski, C. Schneider, O. Mohareri, and S. Salcudean. Bimanual teleoperation with heart motion compensation on the da vinci®research kit: implementation and preliminary experiments. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4101–4108, 2016.

[29] M. Selvaggio, G. A. Fontanelli, F. Ficuciello, L. Villani, and B. Siciliano. Passive virtual fixtures adaptation in minimally invasive robotic surgery. *IEEE Robotics and Automation Letters*, 3(4):3129–3136, Oct 2018.

[30] J. J. Abbott, G. D. Hager, and A. M. Okamura. Steady-hand teleoperation with virtual fixtures. In *The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003.*, pages 145–151, Nov 2003.

[31] R. Moccia, C. Iacono, B. Siciliano, and F. Ficuciello. Vision-based dynamic virtual fixtures for tools collision avoidance in robotic surgery. *IEEE Robotics and Automation Letters*, 5(2):1650 – 1655, Apr 2020.

[32] M. Marinho, H. Ishida, K. Harada, K. Deie, and M. Mitsuishi. Vision-based dynamic virtual fixtures for tools collision avoidance in robotic surgery. *IEEE Robotics and Automation Letters*, 5(2):524–531, Apr 2020.

[33] T. Osa, C. Staub, and A. Knoll. Framework of automatic robot surgery system using visual servoing. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1837–1842, Oct 2010.

[34] C. Staub, T. Osa, A. Knoll, and R. Bauernschmitt. Automation of tissue piercing using circular needles and vision guidance for computer aided laparoscopic surgery. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4585–4590, May 2010.

[35] S. Hutchinson M. W. Spong and M. Vidyasagar. *Robot modeling and control*. John Wiley & Sons, Ltd., 2006.

[36] B. C. Becker, S. Voros, R. A. MacLachlan, G. D. Hager, and C. N. Riviere. Active guidance of a handheld micromanipulator using visual servoing. In *2009 IEEE In-*

ternational Conference on Robotics and Automation (ICRA), pages 339–344, May 2009.

[37] C. S. Chen, M. S. Hsieh, Y. W. Chiu, C. H. Tsai, S. M. Liu, C. C. Lu, and P. L. Yen. An unconstrained virtual bone clamper for a knee surgical robot using visual servoing technique. *Journal of the Chinese Institute of Engineers*, 33(3):379–386, 2010.

[38] L. Ott, F. Nageotte, P. Zanne, and M. de Mathelin. Physiological motion rejection in flexible endoscopy using visual servoing and repetitive control : Improvements on non-periodic reference tracking and non-periodic disturbance rejection. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4233–4238, May 2009.

[39] M. Azizian, M. Khoshnam, N. Najmaei, and R. V. Patel. Visual servoing in medical robotics: a survey. part i: endoscopic and direct vision imaging – techniques and applications. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 10(3):263–274, 2014.

[40] D. Stoyanov. Surgical vision. *Annals of Biomedical Engineering*, 40(2):332–334, Feb 2012.

[41] R. Y. Tsai and R. K. Lenz. Real time versatile robotics hand/eye calibration using 3d machine vision. In *1988 IEEE International Conference on Robotics and Automation (ICRA)*, pages 554–561 vol.1, Apr 1988.

[42] S. J. D. Prince. *Computer Vision: Models Learning and Inference*. Cambridge University Press, 2012.

[43] J. H. Jang, S. H. Kim, and Y. K. Kwak. Calibration of geometric and non-geometric errors of an industrial robot. *Robotica*, 19(3):311–321, may 2001.

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

[45] A. Malti and J. P. Barreto. Robust hand-eye calibration for computer aided medical endoscopy. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5543–5549, May 2010.

[46] M. Z. Huang and O. Masory. A simple method of accuracy enhancement for industrial manipulators. *International Journal of Advanced Manufacturing Technology*, 8(2):114–122, 1993.

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

[48] S. Aksungur. Remote center of motion (rcm) mechanisms for surgical operations. *International Journal of Applied Mathematics, Electronics and Computers*, 3(2):119–126, Feb 2015.

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

[50] Intuitive Surgical Inc. *ISI API User Guide: da Vinci Research Kit*, 2010.

[51] G. A. Fontanelli, M. Selvaggio, M. Ferro, F. Ficuciello, M. Vendittelli, and B. Siciliano. A v-rep simulator for the da vinci research kit robotic platform. In *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob*, pages 1056–1061, 2018.

[52] B. Rosa, C. Gruijthuijsen, B. van Cleynenbreugel, J. V. Sloten, D. Reynaerts, and E. V. Poorten. Estimation of optimal pivot point for remote center of motion alignment in surgery. *International Journal of Computer Assisted Radiology and Surgery*, 10(2):205–215, 2014.

[53] Hermann Mayer, István Nagy, and Alois Knoll. Kinematics and modelling of a system for robotic surgery. In *On Advances in Robot Kinematics*, pages 181–190. Springer Netherlands, 2004.

[54] F. Schramm, F. Geffard, G. Morel, and A. Micaelli. Calibration free image point path planning simultaneously ensuring visibility and controlling camera path. In *2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2074–2079, April 2007.

[55] A. Filion, A. Joubair, A. S. Tahan, and I. A. Bonev. Robot calibration using a portable photogrammetry system. *Robotics and Computer-Integrated Manufacturing*, 49:77 – 87, 2018.

[56] G. Xiong, Y. Ding, L. Zhu, and C. Su. A product-of-exponential-based robot calibration method with optimal measurement configurations. *International Journal of Advanced Robotic Systems*, 14(6):1–12, 2017.

[57] J. Schmidt and H. Niemann. Data selection for hand-eye calibration: A vector quantization approach. *The International Journal of Robotics Research*, 27(9):1027–1053, 2008.

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

[59] N. Lawrentschuk, N. E. Fleshner, and D. M. Bolton. Laparoscopic lens fogging: A review of etiology and methods to maintain a clear visual field. *Journal of Endourology*, 24(6):905–913, 2010.

[60] T. G. Manning, N. Papa, M. Perera, S. McGrath, D. Christidis, M. Khan, R. O'Beirne, N. Campbell, D. Bolton, and N. Lawrentschuk. Laparoscopic lens fogging: solving a common surgical problem in standard and robotic laparoscopes via a scientific model. *Surgical Endoscopy*, 32(3):1600–1606, 2018.

[61] Q. Ma, H. Li, and G. S. Chirikjian. New probabilistic approaches to the $ax = xb$ hand-eye calibration without correspondence. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4365–4371, May 2016.

[62] H. Li, Q. Ma, T. Wang, 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, Jan 2016.

[63] Q. Ma, Z. Goh, S. Ruan, and G. S. Chirikjian. Probabilistic approaches to the $axb = ycz$ calibration problem in multi-robot systems. *Autonomous Robots*, 42(7):1497–1520, Oct 2018.

[64] M. K. Ackerman and G. S. Chirikjian. A probabilistic solution to the $ax = xb$ problem: Sensor calibration without correspondence. In *1st International Conference on Geometric Science of Information (GSI)*, pages 693–701, 2013.

[65] H. Wang, T. Gao, J. Kinugawa, and K. Kosuge. Finding measurement configurations for accurate robot calibration: Validation with a cable-driven robot. *IEEE Transactions on Robotics*, 33:1156–1169, 2017.

[66] K. Pachtrachai, F. Vasconcelos, F. Chadebecq, M. Allan, S. Hailes, V. Pawar, and D. Stoyanov. Adjoint transformation algorithm for hand–eye calibration with applications in robotic assisted surgery. *Annals of Biomedical Engineering*, 46(10):1606–1620, Oct 2018.

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

[68] C. J. Taylor and J. P. Ostrowski. Robust vision-based pose control. In *2000 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2734–2740, 2000.

[69] H. H. Chen. A screw motion approach to uniqueness analysis of head-eye geometry. In *1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 145–151, Jun 1991.

[70] K. Pachtrachai, F. Vasconcelos, G. Dwyer, S. Hailes, and D. Stoyanov. Hand-eye calibration with a remote centre of motion. *IEEE Robotics and Automation Letters*, 4(4):3121–3128, Oct 2019.

[71] M. Rufli, D. Scaramuzza, and R. Siegwart. Automatic detection of checkerboards on blurred and distorted images. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3121–3126, Sep. 2008.

[72] M. Allan, P. L. Chang, S. Ourselin, D. J. Hawkes, A. Sridhar, J. Kelly, and D. Stoyanov. Image based surgical instrument pose estimation with multi-class labelling and optical flow. In *2015 International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 331–338, 2015.

[73] K. Daniilidis. Hand-eye calibration using dual quaternions. *The International Journal of Robotics Research*, 18(3):286–298, 1999.

[74] H. Zhuang, S. Roth, and R. Sudhakar. Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form ax=yb. *IEEE Transactions on Robotics and Automation*, 10(4):549–554, Aug 1994.

[75] F. Dornaika and R. Horaud. Simultaneous robot-world and hand-eye calibration. *IEEE Transactions on Robotics and Automation*, 14(4):617–622, Aug 1998.

[76] A. Tabb and K. M. Ahmad Yousef. Solving the robot-world hand-eye(s) calibration problem with iterative methods. *Machine Vision and Applications*, 28(5):569–590, Aug 2017.

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

[78] K. Huang and C. Stachniss. On geometric models and their accuracy for extrinsic sensor calibration. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2018.

[79] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson, 2004.

[80] R. M. Murray, Z. X. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[81] 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, Feb 1989.

[82] F. C. Park and B. J. Martin. Robot sensor calibration: solving $\mathbf{AX} = \mathbf{XB}$ on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, Oct 1994.

[83] J. C. K. Chou and M. Kamel. Quaternions approach to solve the kinematic equation of rotation, $\mathbf{A}_a\mathbf{A}_x = \mathbf{A}_x\mathbf{A}_b$, of a sensor-mounted robotic manipulator. In *1988 IEEE International Conference on Robotics and Automation (ICRA)*, pages 656–662 vol.2, Apr 1988.

[84] J. C. K. Chou and M. Kamel. Finding the position and orientation of a sensor on a robot manipulator using quaternions. *The International Journal of Robotics Research*, 10(3):240–254, 1991.

[85] A. Malti. Hand–eye calibration with epipolar constraints: Application to endoscopy. *Robotics and Autonomous Systems*, 61(2):161 – 169, 2013.

[86] J. Schmidt, F. Vogt, and H. Niemann. Robust hand–eye calibration of an endoscopic surgery robot using dual quaternions. In *Pattern Recognition*, pages 548–556, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[87] K. H. Strobl and G. Hirzinger. Optimal hand-eye calibration. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4647–4653, Oct 2006.

[88] F. Mourgues and É. Coste-Maniére. Flexible calibration of actuated stereoscopic endoscope for overlay in robot assisted surgery. In *2002 International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 25–34, 2002.

[89] S. Thompson, D. Stoyanov, C. Schneider, K. Gurusamy, S. Ourselin, B. Davidson, D. Hawkes, and M. J. Clarkson. Hand–eye calibration for rigid laparoscopes using an invariant point. *International Journal of Computer Assisted Radiology and Surgery*, 11(6):1071–1080, 2016.

[90] R. Horaud and F. Dornaika. Hand-eye calibration. *International Journal of Robotics Research*, 14(3):195–210, 1995.

[91] J. Heller, M. Havlena, and T. Pajdla. A branch-and-bound algorithm for globally optimal hand-eye calibration. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1608–1615, June 2012.

[92] J. Schmidt, F. Vogt, and H. Niemann. Calibration–free hand–eye calibration: A structure–from–motion approach. In *The 27th Annual Symposium of the German Association for Pattern Recognition (DAGM)*, pages 67–74, 2005.

[93] W. Li, M. Dong, N. Lu, X. Lou, and P. Sun. Simultaneous robot–world and hand–eye calibration without a calibration object. *Sensors (Basel)*, 18(11), 2018.

[94] J. Heller, M. Havlena, A. Sugimoto, and T. Pajdla. Structure-from-motion based hand-eye calibration using l_∞ minimization. In *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3497–3503, June 2011.

[95] Z. Kukelova, J. Heller, and T. Pajdla. Hand-eye calibration without hand orientation measurement using minimal solution. In *The 12th Asian Conference on Computer Vision (ACCV)*, pages 576–589, 2013.

[96] H. Li, Q. Ma, T. Wang, 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, Jan 2016.

[97] M. K. Ackerman, A. Cheng, and G. Chirikjian. An information-theoretic approach to the correspondence-free $ax = xb$ sensor calibration problem. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4893–4899, May 2014.

[98] K. Erdmann and M. J. Wildon. *Introduction to Lie Algebras*. Springer-Verlag London, 2006.

[99] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics*, 27(4), Nov 2008.

[100] X. Zhi and S. Schwertfeger. Simultaneous hand-eye calibration and reconstruction. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1470–1477, Sep. 2017.

[101] M. Shah. Solving the robot-world/hand-eye calibration problem using the kronecker product. *Journal of Mechanisms and Robotics*, 5(3), 2013.

[102] P. Loustaunau W. W. Adams. *An Introduction to Gröbner Bases*. American Mathematical Society, 1994.

[103] C. Nissler, Z. Márton, H. Kisner, U. Thomas, and R. Triebel. A method for hand-eye and camera-to-camera calibration for limited fields of view. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5868–5873, 2017.

[104] F. Vasconcelos, E. Mazomenos, J. Kelly, and D. Stoyanov. Rcm-slam: Visual localisation and mapping under remote centre of motion constraints. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9278–9284, 2019.

[105] C. J. Taylor and J. P. Ostrowski. Robust vision-based pose control. In *2000 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2734–2740, April 2000.

[106] A. Malti and J. P. Barreto. Hand–eye and radial distortion calibration for rigid endoscopes. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 9(4):441–454, 2013.

[107] J. C. Bezdek and R. J. Hathaway. Some notes on alternating optimization. In *2002 AFSS International Conference on Fuzzy Systems*, pages 288–300, 2002.

[108] J. M. Selig. *Lie Groups and Lie Algebras in Robotics*, pages 101–125. Springer Netherlands, 2004.

[109] P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[110] KUKA Laboratories GMBH. *LBR iiwa 7 R800 Assembly Instructions*, 2014.

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

[112] G. Dwyer, R. J. Colchester, E. J. Alles, E. Maneas, S. Ourselin, T. Vercauteren, J. Deprest, E. V. Poorten, P. D. Coppi, A. E. Desjardins, and D. Stoyanov. Robotic control of a multi-modal rigid endoscope combining optical imaging with all-optical

ultrasound. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3882–3888, 2019.

[113] ams AG. *NanEye Miniature Camera Module - datasheet*, 2020.

[114] J. Y. Bouguet. Matlab camera calibration toolbox. 2000.

[115] F. Vasconcelos, E. Mazomenos, J. Kelly, S. Ourselin, and D. Stoyanov. Relative pose estimation from image correspondences under a remote center of motion constraint. *IEEE Robotics and Automation Letters*, 3(3):2654–2661, July 2018.

[116] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, 1987.

[117] S. Ourselin F. Vasconcelos, D. Peebles and D. Stoyanov. Spatial calibration of a 2d/3d ultrasound using a tracked needle. *International Journal of Computer Assisted Radiology and Surgery*, 11(6):1091–1099, 2016.

[118] F. Vasconcelos, D. Peebles, S. Ourselin, and D. Stoyanov. Similarity registration problems for 2d/3d ultrasound calibration. In *2016 European Conference on Computer Vision (ECCV)*, pages 171–187, 2016.

[119] S. Voros, E. Orvain, P. Cinquin, and J. A. Long. Automatic detection of instruments in laparoscopic images: A first step towards high-level command of robotic endoscopic holders. *The International Journal of Robotics Research*, 26(11-12):1173–1190, Nov 2007.

[120] M. Allan, S. Ourselin, S. Thompson, D. J. Hawkes, J. Kelly, and D. Stoyanov. Towards detection and localization of instruments in minimally invasive surgery. *IEEE Transactions on Biomedical Engineering*, 60(4):1050–1058, 2013.

[121] Z. Pezzementi, S. Voros, and G. D. Hager. Articulated object tracking by rendering consistent appearance parts. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3940–3947, 2009.

[122] A. Reiter, P. K. Allen, and T. Zhao. Feature classification for tracking articulated surgical tools. In *2012 International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 592–600, 2012.

[123] M. Allan, S. Thompson, M. J. Clarkson, S. Ourselin, D. J. Hawkes, J. Kelly, and D. Stoyanov. 2d-3d pose tracking of rigid instruments in minimally invasive surgery. In *2014 International Conference on Information Processing in Computer-Assisted Interventions (IPCAI)*, pages 1–10, 2014.

[124] L. Zhang, M. L. Ye, P. L. Chan, and G. Z. Yang. Real-time surgical tool tracking and pose estimation using a hybrid cylindrical marker. *International Journal of Computer Assisted Radiology and Surgery*, 12:921–930, 2017.

[125] J. Lin, N. T. Clancy, D. Stoyanov, and D. S. Elson. Tissue surface reconstruction aided by local normal information using a self-calibrated endoscopic structured light system. In *2015 International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 405–412, 2015.

[126] P. Kazanzidesf, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. Dimaio. An open-source research kit for the da vinci® surgical system. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6434–6439, 2014.

[127] L. R. Rabiner. *Multirate Digital Signal Processing*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1996.

[128] C. Knapp and G. Carter. The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(4):320–327, Aug 1976.

[129] N. S. M. Tamim and F. Ghani. Hilbert transform of fft pruned cross correlation function for optimization in time delay estimation. In *2009 IEEE 9th Malaysia International Conference on Communications (MICC)*, pages 809–814, Dec 2009.

[130] J. B. Kuipers. *Quaternions and rotation Sequences: a Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 1999.

[131] B. K. P. Horn. Some notes on unit quaternions and rotation, 2001.

[132] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[133] M. Turan, Y. Almalioglu, H. Araujo, E. Konukoglu, and M. Sitti. A non-rigid map fusion-based direct slam method for endoscopic capsule robots. *International Journal of Intelligent Robotics and Applications*, 1:399–409, 2017.

[134] G. Villarrubia, J. F. de Paz, P. Chamoso, and F. D. la Prieta. Artificial neural networks used in optimization problems. *Neurocomputing*, 272:10 – 16, 2018.

[135] H. Y. Wu, W. Tizzano, T. T. Andersen, N. A. Andersen, and O. Ravn. *Hand-Eye Calibration and Inverse Kinematics of Robot Arm Using Neural Network*, pages 581–591. Springer International Publishing, 2014.

[136] C. T. Cao, V. P. Do, and B. R. Lee. A novel indirect calibration approach for robot positioning error compensation based on neural network and hand-eye vision. *Applied Science*, 9(9):1–17, 2019.