Eye Gaze Tracking in Surgical Robotics

by

Irene Go Tong

Bachelor of Applied Science, Simon Fraser University, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES
(Biomedical Engineering)

The University of British Columbia
(Vancouver)

August 2017

© Irene Go Tong, 2017

Abstract

Robot-assisted surgery allows surgeons to have improved control and visualization in minimally invasive procedures. Eye gaze tracking is a valuable tool for studying and improving the surgeon experience during robot-assisted surgery. Eye gaze information gives insight on how surgeons are interacting with surgical systems as well as their intentions during surgical tasks. This thesis describes the development of an eye gaze tracker for the da Vinci Surgical System. The eye gaze tracker is designed to track both the 2D and 3D eye gaze of a surgeon. It interfaces with the da Vinci Surgical System through the da Vinci Research Kit (dVRK) and Robot Operating System (ROS) frameworks. The use of the eye gaze tracker is demonstrated in two applications. Firstly, a motor control framework is designed to aid surgeons in moving surgical tools towards their point of gaze. A haptic force is applied to the da Vinci master manipulators to pull the surgeon's hands towards where they are looking. This framework is demonstrated on a full da Vinci Surgical System on dry lab tasks. Secondly, eye gaze information is collected from 7 surgeons performing realistic clinical tasks with the da Vinci Surgical System. A prediction model using a random forest classifier is built based on the eye gaze information and tool kinematic information in order to predict how and when surgeons move their camera. This behavioural model has applications in both surgeon training and endoscope automation.

Lay Summary

With robot-assisted surgery, surgeons are able to operate on patients using minimally invasive tools. The robotic platform gives surgeons better control and visualization of the surgical scene. In this thesis, we aim to make improvements on surgical robot systems by developing an eye gaze tracker to determine where surgeons are looking. This eye gaze tracker monitors the surgeons eye gaze position in both 2D and 3D. We use this eye gaze tracker for two applications. The first application uses the eye gaze information to help the surgeon move surgical instruments towards where they are looking. A force is applied to the surgeon's hands to guide their motions towards where they are looking within the surgical scene. The second application uses eye gaze information and surgical tool information within a machine learning framework to predict how and when surgeons will move their endoscope cameras.

Preface

I was the lead investigator for the work described in this thesis. I was supervised by Dr. Tim Salcudean, who presented me with the research topic of eye gaze tracking for surgical robotics. Dr. Craig Hennessey also supervised my research throughout my degree, providing guidance for the design of the eye gaze tracker.

I was the main developer of the software implemented for this thesis. I designed the eye gaze tracking algorithm described in Chapter 3 and carried out the study described in Chapter 4. The analysis described in Chapter 4 was carried out under an ethics approval from the UBC Clinical Research Ethics Board. The project title is "Evaluation of eye-gaze tracking and control for the da Vinci console. The certificate number is H15-0153.

Chapter 5 was presented at the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). The title of the work published in the IROS conference proceedings is "A retrofit eye gaze tracker for the da Vinci and its integration in task execution using the da Vinci Research Kit". I was the lead investigator for this work and wrote the software implementation and conducted user studies. Omid Mohareri and Samuel Tatasurya were co-investigators. Omid worked on calibrating with the da Vinci Research Kit and designing the control framework. Samuel Tatasurya completed data analysis on the final system. Dr. Salcudean and Dr. Hennessey were supervisors of this work.

Chapter 6 was carried out during an internship at Intuitive Surgical Inc. This work was carried out under the supervision of Dr. Anthony Jarc. I collected and analyzed the data used in this work.

Table of Contents

List of Tables

List of Figures

Glossary

2D	Two Dimensional
3D	Three Dimensional
AUC	Area Under the Curve
DVRK	da Vinci Research Kit
FNIRS	Functional Near-Infrared Spectroscopy
FPS	Frames Per Second
GPU	Graphics Processing Unit
GUI	Graphical User Interface
IR	Infrared
LED	Light Emitting Diode
LEDS	Light Emitting Diodes
LTI	Linear Time Invariant
MTM	Master Manipulator
PID	Proportional-Integral-Derivative
POG	Point of Gaze
PSM	Patient-side Manipulator

PSOM Parameterized Self-Organizing Map

RANSAC RANdom SAmple Consensus

RAS Robot-Assisted Surgery

ROC Receiver Operating Characteristic

ROI Region of Interest

ROS Robot Operating System

SDK Software Development Kit

SLAM Simultaneous Localization And Mapping

SVN Subversion

TCP/IP Transmission Control Protocol/Internet Protocol

USB Universal Serial Bus

Acknowledgments

I would foremost like to thank Dr. Tim Salcudean for his supervision during this thesis. He was always pointing me in the right direction, and his guidance was integral to the success of this thesis. I am so thankful for the patience he has had with me, and for allowing me to explore different approaches and solutions even if they did not always work out. Furthermore, his very detailed feedback on this thesis document is greatly appreciated.

I would like to thank Dr. Craig Hennessey for his guidance towards the development of the eye gaze tracker. Any question regarding eye gaze tracking was always answered with words of wisdom and a good solution for how to fix things.

I would also like to thank Dr. Anthony Jarc for his supervision during my time as an intern at Intuitive Surgical Inc. I learned a lot from the discussions we had regarding eye gaze behaviour, and I am inspired by his ability to brainstorm and come up with great research ideas.

I would like to thank Dr. Salcudean for his financial support for my research at UBC, even when I extended past my expected timeline. I would also like to thank Mitacs for financial support, which allowed me to pursue internships at Intuitive Surgical Inc. and Gazepoint. I would lastly like to thank the NSERC-CRSNG for funding the UBC Engineers in Scrubs (EiS) CREATE program, which I had the opportunity to participate in.

I would like to thank the members of the Robotics and Control Laboratory (RCL) at UBC for all of the discussions, support, and technical help. This is a great lab filled with kind, thoughtful and really smart people. I'm also thankful for the constant reminder to exercise with Spartacus, even though I was resistant to the idea at first. Thanks to all the co-op students who worked with me on this project;

Sean Liu, Samuel Tatasurya, Maxwell Li, and Leo Metcalf. Their good work really moved the project along, and they were a pleasure to work with.

I would like to thank my community at the Fujian Evangelical Church for all of the spiritual support and prayers. Thanks to the Praise Team for having me on the team. My fatigue and stress always disappears when I go to practices. Thanks to my small group for listening to my prayer requests and for the fun discussions at the end of each week.

Finally, I would like to thank my family, Jessie Tong, Tat Keung Tong, and Iva Tong for their love and support during my thesis. Their love and patience with me when I was facing deadlines helped me push through everything. They have made me who I am, and are always pushing me to improve. Thanks especially to Lamont Chan for his love and patience throughout this thesis degree.

Chapter 1

Introduction

1.1 Eye Gaze Tracking

Humans are visual beings and use sight to perceive and interact with the surrounding environment. Eye movements can depict a person's cognitive processes and reflect their physical and emotional state. The measure of eye movements, oculography, provides a rich source of information that is applicable to a wide range of fields including psychology, human-computers interaction, and robotics.

Oculography, more commonly referred to by the more colloquial term "eye gaze tracking", is made possible through the use of eye gaze tracking technology. Most modern eye gaze tracking technologies perform video-oculography, and capture images of the eyes in order to determine gaze. Eye gaze trackers can be designed to calculate the point of gaze on a display screen or the Three Dimensional (3D) gaze position in a physical space. There are different configurations for gaze trackers depending on the environment in which the tracker will be used. Desktop eye gaze trackers monitor gaze on a computer monitor screen. Head-mounted eye gaze trackers allow users to move freely in a space, and track where they look in their surrounding environment. Gaze trackers can further be integrated into specific applications, such as a stereoscopic displays in virtual reality devices.

1.2 Robotic Surgery

Minimally invasive surgeries are carried out using small ports cut in the abdomen through which thin tools and cameras are inserted. These types of procedures do not require the large incisions made during traditional open surgeries, resulting in faster recovery times, reduced trauma, and reduced scarring. During manual laparoscopy, the surgeon uses hand-held surgical tools to perform surgery. A major limitation is the hand-eye mapping between the tool tip and the endoscope feed. As the surgical tools pivot at the port openings, orientation of the tools is not intuitive. Additionally, most laparoscopic surgeries are carried out with a Two Dimensional (2D) display, and so the surgeon's sense of depth within the surgical scene is limited.

Robotic technology is used in surgical procedures to allow surgeons to have more precision and better control of surgical tools and improved visualization of the surgical scene during minimally invasive surgery. An example of a commercial surgical robot system is the da Vinci ®Surgical System by Intuitive Surgical Inc. (Sunnyvale, CA). The da Vinci is a tele-operated robotic system in which the surgeon operates the robot using hand-held robotic manipulators and foot controls to remotely control patient-side manipulators that operate instruments inside the patient. There are two master manipulators, one for each hand of the surgeon. The movement of the surgical tools in each robotic slave arm of the da Vinci is mapped to the movements made by the surgeon using the master manipulators. A robotic platform has the benefit of allowing for a more intuitive hand-eye mapping, increased wrist dexterity and 3D imaging. This allows surgeons to perform more complex minimally invasive surgeries than what is achievable with manual laparoscopy.

There is an opportunity for eye gaze tracking to further improve the human-robot interface. Eye gaze information carries the intent and focus of a surgeon, and directly portrays how the surgeon is visualizing the surgical field. Eye gaze information can be used to adjust the visual field by adjusting the camera based on where a surgeon is looking [7], [34], or assist in tool movement based on the surgeons intent [45], [46]. Eye gaze tracking technology also allows researchers to better understand how surgeons interact with the surgical robot platform and for

measuring surgeon behaviour [3]. While eye gaze has been previously studied with laparoscopic surgery, the use of eye gaze tracking technology in the study of surgeon behaviour with robotic surgical platforms is relatively new and unexplored.

1.3 Objectives of Thesis

The research carried out during this masters research thesis is threefold. The first objective of this thesis is to develop and evaluate a stereo eye gaze tracker designed for use with the da Vinci surgical robot. The design described in this thesis is retro-fit to the da Vinci surgical robot. The eye gaze tracker is able to determine a user's eye gaze position in 2D or 3D space within the da Vinci surgeon console display. For 2D eye gaze tracking a standard polynomial regression is carried out, and for 3D eye gaze tracking a novel hybrid geometric and regression-based approach is used. The eye gaze tracker is evaluated for accuracy in both 2D and 3D tracking schemes.

The gaze tracker is then applied to a human-robot interface, and a control framework is established where the 3D eye gaze position is used to direct the movement of surgical tools. The objective of this work is to demonstrate the feasibility of a gaze-based control framework for a surgical robot system. This is the first time gaze-contingent motor control has been demonstrated in a full surgical robot system. This system is evaluated with a dry lab task (peg placement). The result of this work has been published in the conference proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015).

Thirdly, surgeon eye gaze behaviour is observed in a realistic, clinical-like setting. The objective of this work is to use eye gaze information to predict camera movement. We evaluate the use of gaze data towards identifying whether a surgeon performing robot-assisted surgery has adjusted their endoscope and predicting in which direction they move their cameras. This is a novel approach to automated endoscope control based on eye gaze information, as the endoscope movement is predicted using a data-driven model instead of heuristic rules. This behavioural model can be used to enable automatic camera control during robotic surgery.

1.4 Organization of Thesis

This thesis is organized into the following sections:

- **Chapter 1** Introduction: Discusses the motivations and objectives of this thesis.
- **Chapter 2** Background: Discusses background information and a review of state-of-the-art technologies.
- **Chapter 3** Eye Gaze Tracker Design: Describes the design of the eye gaze tracker hardware and software algorithm.
- **Chapter 4** Eye Gaze Tracker Accuracy: Presents the results of an accuracy evaluation for 2D and 3D eye gaze tracking.
- **Chapter 5** Gaze Motor Control: Discusses the use of the eye gaze tracker for controlling robot movement.
- **Chapter 6** Camera Movement Prediction: Discusses the use of eye gaze data for predicting endoscope movement during robot-assisted surgery.
- **Chapter 7** Conclusion and a list of contributions of this thesis.

Chapter 2

Background Information

This chapter presents background information for an eye gaze tracking system in robot-assisted surgery. It provides an introduction the eye's anatomy and physiology. An overview of existing eye gaze tracking technologies and methodologies is discussed. Finally, state-of-the-art eye gaze tracking applications in robot-assisted surgery is discussed.

2.1 Eye Anatomy and Physiology

2.1.1 Eye Anatomy

Knowledge of the anatomy of the eye is important for the design of an eye gaze tracking system, and for the interpretation of eye gaze tracking data. The eye is a sensory organ through with humans can visualize the world. The eye is complex and has many anatomical components which convey and process light in order to achieve vision. A diagram of the eye is shown in Figure 2.1.

The anterior outer fibrous layer of the eye is the cornea. This is a transparent dome through which light enters the eye. The cornea protects the eye from harmful particles and also aids in refracting and focussing light. It has five main layers- the epithelium, Bowmans layer, the stroma, Decements membrane, and the endothelium. Roughly 65-75% of light focussing in the eye is done in the cornea

[1] Image obtained from https://nei.nih.gov/photo/anatomy-of-eye.

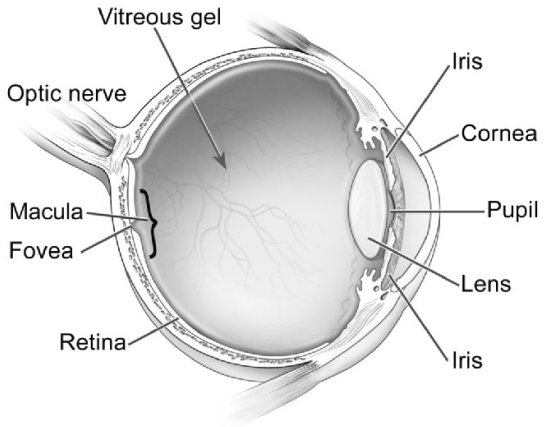


Figure 2.1: Diagram of the eye[1].

[24]. Due to the transparent nature of the cornea, the light from the Infrared (IR) Light Emitting Diode (LED) is both reflected and transmitted through the cornea and subsequent structures in the eye. At the boundary between each transparent layer, the partially reflected light produces a different reflected image. These images are referred to as Purkinje images. The first Purkinje image is the reflection of light off the outer cornea surface, and is the image considered by most gaze tracking algorithms since it is the brightest. The fourth Purkinje image can also be used to estimate the pupil position in a 3D model of the eye [1]. Gaze estimation involves the calculation of the exact location of the pupil in space. However, since the cornea is dome shaped and refractive, the image of the pupil shows a refracted image and not the actual location of the pupil. The index of refraction and radius of curvature of the cornea can be taken into account to achieve a more accurate estimation of the pupil location. The average value of the index of refraction is 1.376 and the average value of the radius of curvature is 0.7 cm [13]. To simplify equations, the cornea is approximated as a perfect spherical structure, when in reality it has a higher radius of curvature towards the centre, and flattens out around the edges.

The posterior outer fibrous layer of the eye is composed of the opaque sclera

and lamina cribrosa. The sclera has a rougher texture than the cornea, and reflected IR light glints appear distorted and blurry. This can cause a problem when trying to find a single glint point on the eye.

The middle layer of the eye is also referred to as the uvea. It is vascular and consists of the IRis, ciliary body and choroids. The IRis is a disc-shaped structure containing a sphincter muscle which dilates and constricts to regulate the amount of light entering the eye. Through the center of the IRis is the pupil, which is the hole through which light passes through. On the circumference of the IRis is the ciliary body. Suspended by the ciliary body is a transparent lens. The lens is a pliable structure which transmits and focuses light according to its thickness. Light then travels through the vitreous humour, which is a transparent fluid that fills the body of the eye.

Perception occurs at the retina, which is a layer on the inner surface of the eye that detects light and allows us to see images. The fovea is a small area in the center of the retina with the most visual acuity. Light travelling through the vitreous humour reaches the retina where it is sensed by photoreceptors. There are two photoreceptors called rods and cones for detecting light. Rods allow for scotopic vision, and allow for vision in dim lighting [40]. Conversely, cones allow for photopic vision and allow for colour vision. Light is focussed by the lens onto the fovea, which is a region on the retina that is dense with cone photoreceptors. Sight perception decreases outside of the fovea. One notable feature of the fovea is that it only comprises 1° of visual angle. This sets the limit for the accuracy of gaze trackers, as it is only possible to detect where a person is looking to the precision of the fovea.

Gaze tracking technologies also consider the optical axis and the visual axis of the eye. The optical axis is defined as the line passing perpendicular to the surfaces of the eye and through the center of each circular structure (cornea, IRis, and lens). The visual axis is the line passing from the center of the fovea. The fovea is offset from the center of the retina, so there is a slight degree offset between the optical axis and visual axis (between 4-5° horizontally and roughly 1.5° vertically [17]). The optical axis may be thought of as a theoretical axis as there is no true optical center for the slightly aspherical components of a real human eye.

2.1.2 Types of Eye Movements

The first aspect of analysing eye gaze data is extracting meaningful metrics from the raw gaze position reported by a gaze tracking device. The physiology behind eye movements is important for interpreting gaze. Many eye gaze metrics used for understanding cognitive behaviour are based on different types of eye movements. There are several different types of eye movements, but of most importance to eye gaze behaviour studies and applications are fixations, saccades, smooth pursuit, and vergence.

A fixation eye movement occurs when the point of gaze remains within a small area for an extended period of time. The duration of a fixation is typically between 200-400 ms with a minimum of 100 ms [53]. Fixations are of interest to researchers because they indicate visual perception and areas of interest.

Saccades are rapid ballistic movements of the eye. They can reach velocities of up to 500° per second for durations in the order of tens of milliseconds [57]. These movements are both voluntary and involuntary and occur for a variety of reasons. Saccades can be goal oriented to direct the eyes at a certain point of interest or occur as reflexes to quickly look at objects. Saccades also stabilize gaze by resetting the eye to a center position in the orbit. The beginning of a saccade has a latency of roughly 100-300 ms from the occurrence of a stimulus. The eye is never completely still, and micro-saccades are constantly occurring in order to correct the eye position.

Smooth pursuit eye movements are a relatively slow motion of the eye. These movements are voluntary and are done when tracking moving objects. During a smooth pursuit, the eyes are directed to maintain vision of a moving target. The maximum velocity of a smooth pursuit is around 300° per second, with a latency of 100-150 ms [5]. Smooth pursuits are initialized with visual signals, but during a pursuit the motion of the eye is based off the visual signal in memory.

Vergence eye movements are the coordinated movement of both eyes in opposite directions in order to view an object at a certain distance away. There may be convergence when the eyes rotate towards each other to view closer objects, or divergence when the eyes move away for farther object. Gaze tracking algorithms can use this property of eyes to calculate a single gaze point on a two dimensional

screen with the constraint that the two eyes must be looking at the same point. Alternatively, gaze tracking can be extended into three-dimensions by calculating the intersection of the line of sight for both eyes.

2.2 Overview of Eye Gaze Tracking Techniques

There are many variations on eye gaze tracking techniques. This reflects the breadth of applications for gaze tracking. For a camera-based system, the image of the eye is first processed to find specific features, such as pupil position, which can be related to gaze position. Once eye features have been found, eye gaze estimation is then carried out through a calibration procedure to map eye feature locations to the point of gaze. This section details current implementations in literature for both feature detection and gaze estimation.

2.2.1 Eye Feature Detection

A common technique used in eye gaze tracking is to image the eye under IR lighting using IR bandpass camera filters. The pupil of the eye is noticeably dark in IR lighting as the IR light is absorbed within the pupil while the IRis reflects the light. This is illustrated in Figure 2.2. Additionally, by placing IR light sources near the visual axis of the camera lens, the IR light reflects off of the retina in the back of the eye and is imaged by the cameras, resulting in a bright pupil effect. The difference between the dark pupil and bright pupil images can be calculated to isolate the pupil.

There are appearance-based and feature-based methods for gaze estimation. Appearance-based methods use the entire image of the eye as inputs to machine learning algorithms which correlate the appearance of the eye with the point of gaze. Examples of this technique are found in [37, 60, 64]. Feature-based eye gaze estimation use specific features of the eye image in order to calculate the gaze position. The image of the eye is processed to identify features such as the pupil, cornea, light reflections or eyelids. The pupil is an important feature as it can be directly related to the direction of gaze. Typically, the pupil is detected under IR lighting, while the cornea is detected in visible light. To detect the center of

Figure 2.2: Image of the eye under infrared lighting.

the pupil, a common strategy is to fit an ellipse to the pupil contour, and take the ellipse center to be the pupil center. In [33], edge detection is carried out along rays extending from an initial starting point. They find additional potential pupil edges by further searching along rays from each detected edge. The RANdom SAmple Consensus (RANSAC) algorithm is used to fit an ellipse to the detected points. The center of the fitted ellipse is taken as the starting point for the next iteration of the Starburst algorithm until convergence. This method is simple and intuitive, however it is negatively affected by occlusions and reflections near the pupil contour. Pre-processing of the image is required to remove any non-pupil features. In [2], the pupil contour is detected by detecting edges in the image, and processing the edge image in order to extract pupil edges. This involves filtering out non-pupil edges with morphological patterns, and acceptance criteria such as line curvature and length.

Another approach to pupil detection is to perform binary thresholding on the eye image to segment dark areas in the image. In [20], an adaptive threshold is applied to threshold the image. Contour detection is then carried out, and the pupil

contour is determined as the most circular contour, based on the isoperimetric quotient criterion. This method is discussed in more detail in Chapter 3. In contrast to finding the pupil contour, an alternative is to directly estimate the center of the pupil. In [66], the gradient of the image is calculated, and an objective function is defined at every pixel location which uses the dot product between the image gradient and the vector from the potential pupil center position. This objective function gives a higher value to the center of circular objects. By applying a weighting based on image intensity (higher weight for lower intensity), the objective function can be used to find the center of the dark and circular pupil. This algorithm is applicable to desktop-based eye gaze trackers where the pupil contour is approximately circular. However, there would be more inaccuracies with a head-mounted eye gaze tracker where the pupil is at a more extreme angle and is elliptical. With IR lighting, the location of the pupil can also be easily determined by finding the difference of the dark and bright pupil images, such as the system described by Morimoto et al. in [41]. The drawback for using such a system is the need for more complex hardware for synchronizing light sources and camera frame capture.

Using IR light sources results in bright light reflections off of the surface of the cornea which can be detected by gaze tracking cameras while not distracting the user. In eye gaze tracking literature, the IR light reflections on the cornea surface are referred to as "glints". A useful feature for eye gaze estimation is the vector between the center of the pupil and the center of a glint in the gaze camera image. This is referred to as the "pupil-glint vector". The pupil-glint vector can be mapped to 2D screen coordinates because the glints act as stationary reference points since the IR light sources do not change position and the cornea surface is approximately spherical. Glint detection is commonly accomplished by thresholding for bright pixels. The presence of several glints from different IR light sources poses a challenge as some glints may not appear due to eyelids or eyeglasses. In [22], Hennessey et al. use point pattern matching to identify glints. They generate a template glint pattern by manually selecting glint locations from a pre-recording image frame. Then, all possible glint positions are matched by distance to the template. This method is translation invariant and can handle cases where individual glints are not present, however it is not scale or rotation invariant.

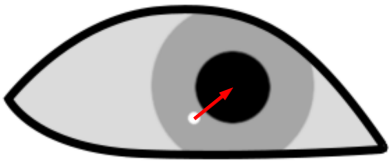
2.2.2 Eye Gaze 2D Estimation

There are two types of techniques for relating eye features to gaze position- regression and model-based methods. Regression methods implement a mapping directly between eye features and gaze estimation. The simplest gaze estimation regression model is a polynomial between the pupil-glint vector and positions on the observed screen. Calibration of the regression model can be carried out by displaying calibration targets to the user and recording the positions of the eye features corresponding with each calibration target. A second-order polynomial is commonly used as it requires a simple calibration and has good accuracy [42]. Figure 2.3 illustrates the relationship between the pupil-glint vector and screen coordinates.

Model-based methods use the detected eye features to calculate a 3D model of the eye. The optical axis is then intersected with the plane on which the eye is gazing at in order to determine the gaze position. In [70], Villanueva and Cabeza propose that the minimum hardware required for accurate model-based eye gaze tracking is one camera and two light sources. The main drawback of model-based methods is that the physical measurements of the system (cameras, light sources, display screen) need to be precisely known and the accuracy of the estimated gaze is very sensitive to inaccuracies of the system measurements [22].

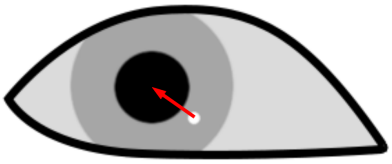
2.2.3 Eye Gaze 3D Estimation

For stereoscopic displays, a common approach to 3D gaze tracking is to calibrate for the 2D plane of the display screen, and then triangulate the two gaze positions to find the depth. The two eyes will rotate inwards to focus on close objects, and rotate outwards to focus on farther objects in a movement called vergence. Daugherty *et al.* conducted an experiment to measure gaze disparity within a stereoscopic image [8]. They concluded that there is a significant difference in gaze disparity when viewing objects in a frontal image plane compared to mid and back planes. Their work shows that gaze disparity is measurable with a binocular gaze tracker and is indicative of gaze depth, but only while viewing images which appear in front of the 2D plane of the display screen. This was in agreement with work by Duchowski et al. who measured vergence error in virtual 3D environment, and observed greater vergence errors for stereo images behind the display screen plane



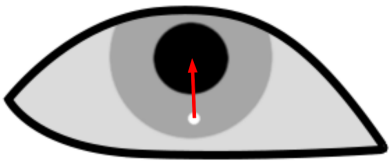
(a) Eye position while looking left.

(b) Gaze position while looking left.

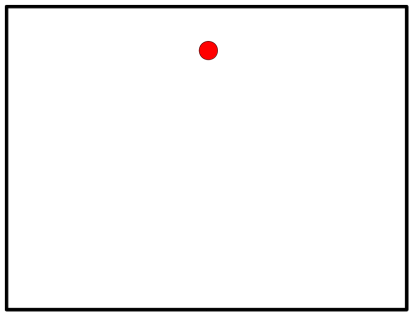


(c) Eye position while looking right.

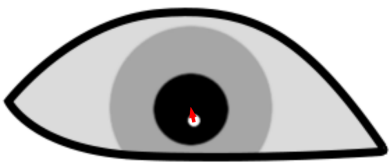
(d) Gaze position while looking right.



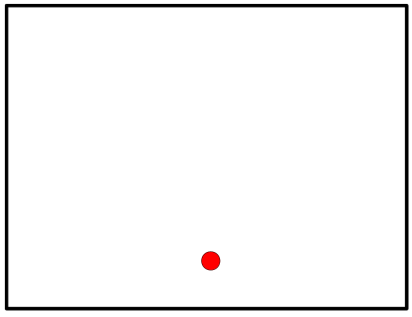
(e) Eye position while looking up.



(f) Gaze position while looking up



(g) Eye position while looking down.



(h) Gaze position while looking down.

Figure 2.3: Relationship between pupil-glint vector (shown as red arrows) to eye gaze position (shown as red dots).

[9].

Lanata et al. evaluate the performance of different feature mapping functions to find the 3D gaze position [30]. They evaluated a linear mapping and a quadratic mapping between the 2D pupil position and gaze depth, and a geometric mapping which calculates gaze depth based on the vergence of the left and right eyes using stereo triangulation. They showed that a geometric mapping has a lower depth error than other mapping functions. Takemura et al. propose a head-mounted eye gaze tracking system which uses Visual Simultaneous Localization And Mapping (SLAM), which generates 3D coordinates of the environment as well as estimates head pose [63]. Visual SLAM tracks feature points found in the video image frames (such as edges or corners) and calculates the 3D position of the feature points based on the change in position of the features over several frames. A Delaunay triangulation is then applied to the world feature points, and the 3D gaze position can be calculated by determining which triangle the 2D gaze position coincides with. This eye gaze tracking technique requires only a 2D calibration, however they did not measure the accuracy of their system.

Pfeiffer et al. used a machine learning approach to determining the 3D point of gaze [51]. They implemented a Parameterized Self-Organizing Map (PSOM) to map 2D gaze positions on a screen to 3D positions within a virtual environment. They argue that a purely geometric approach has disadvantages because the physical dimensions of the system need to be known, and users may demonstrate varying behavioural patterns of vergence movements. They compared their approach with a geometric approach which finds the minimal distance between the two visual axes. They found that PSOM was significantly more precise and accurate. Wang et al. countered these findings and implemented a modified geometric approach and compared it with the PSOM algorithm [73]. In their system, they use triangulation to map the 2D gaze position of the left and right eyes to a 3D point in a stereoscopic display. A difference in their geometric approach to [51] is that the vertical position is taken to be the average of the left and right eyes instead of directly using the visual axes of both eyes. They additionally filter the 2D screen coordinate data points with a Butterworth filter. The calibration sequence for their system requires a user to watch a calibration point move in a 3D trajectory and they note that an advantage of their system is the large number of samples captured dur-

ing calibration. The accuracy for their geometric approach was significantly higher than their PSOM implementation in depth estimation, but was comparable for horizontal and vertical gaze estimation. They note that the calibration procedure in the geometric approach is more than twice faster than with PSOM. The discrepancy between [51] and [73] is that the PSOM and geometric approaches compared do not have the exact same implementation. However, it may be concluded that there could be different situations that warrant the use of either method. For systems with an apparatus or environment that is difficult to measure precisely, using the PSOM method would be advantageous. For systems where speed of calibration is important or a more accurate depth estimation is needed then the geometric method can be applied.

There are also instances of 3D gaze tracking techniques for physical environments, which skip the intermediate step of calibrating to a 2D plane. Hennessey and Lawrence take a purely geometrical approach and determine the 3D Point of Gaze (POG) by calculating the 3D visual axis of both eyes and finding the midpoint of the shortest line between the two visual axes [21]. They obtain an accuracy of 3.93 ± 2.83 cm over a depth range of 30 cm. Tostado et al. use Gaussian Process Regression (non-parametric Bayesian supervised learning algorithm) to map the 2D pupil position to the gaze position in 3D [69]. Their gaze tracking system was designed for the control of a robotic arm using 3D gaze positions. To calibrate, the user watches the end of a robotic gripper as it moves over a 3D trajectory. Through their calibration process, they obtain 500-1500 calibration data points. Additionally, by using Gaussian Process regression, they are able to determine the uncertainty of estimated gaze positions, which is useful for safety considerations when controlling a robotic arm. Their method shows that 2D gaze features can be used to estimate a 3D position of a robotic system with an accuracy (1.6 ± 1.7 cm) over a depth range of 20 cm. This accuracy is reported based on an analysis in which 70% of recorded data was used for calibration, and 30% was used for calculating error. The accuracy of their system for separate calibration and testing data sets is not reported. In this approach, the head motion of the user needs to be constrained by a head rest as only the pupil position is used.

While most 3D gaze tracking methods require the use of at least one camera for each eye to observe vergence, some groups have performed 3D gaze tracking

through measuring a single eye. In addition to vergence, eye lens accommodation can be measured as an indicator of depth because the eye lens becomes thinner to focus on farther objects and thicker to focus on nearer objects. In [1], Lee et al. measures the change in lens curvature with the change in position of the 1st and 4th Purkinje images in relation to the pupil center position, as well as change in pupil diameter. The depth position is then determined with a multi-layered perceptron. Users calibrated their gaze by looking at nine reference points at five depth positions between 10-50 cm. They achieved an accuracy of 0.48 cm in the horizontal direction, 0.77 cm in the vertical direction, and 4.59 cm in depth. However, their method is user-dependent, and the authors note that for users with poor eyesight, lens accommodation will not be as evident, and the pupil size does not adjust with depth.

There are a variety of approaches for accomplishing 3D eye gaze tracking, and no single approach will be optimal for all situations. The selection of an approach for 3D eye gaze tracking is dependent on the environment in which gaze tracking is being carried out. Factors include the 3D stimulus object (real or virtual), user head motion, speed and time limitations, and the user's eyesight.

2.3 State-of-the-Art Eye Gaze Tracking Applications in Robotic Surgery

Pioneering work in the field of gaze contingent robot control has been carried out by researchers at Imperial College. They have integrated a commercial Tobii x50 eye gaze tracker within the da Vinci stereo console by using dichroic beam splitters placed within the stereo console to direct the IR light to the eye gaze tracker [45, 48, 59, 72]. They have implemented their eye gaze tracking system in a variety of applications in robotic surgery including tool motion control, endoscope auto-focus, and motion stabilization. More recent work by the same group is carried out with remote eye gaze trackers attached to a stereo screen and a teleoperated robotic system [14, 15].

In an application directly related to visualization during surgery, Mylonas et al. use eye gaze for motion stabilization and depth recovery [43]. They track the depth of eye gaze and control the endoscope using a Proportional-Integral-

Derivative (PID) controller to maintain a fixed depth between the gaze position and the endoscope. To calculate 3D fixation positions, the pupil-glint vectors are mapped to 3D gaze positions with a radial basis function. To calibrate, subjects gaze at nine targets presented at three different depths. They evaluated motion stabilization in a virtual experiment, where a sphere oscillating along the z-axis is shown to the subjects. By focusing on the sphere, the subjects were able to stabilize the motion of the simulated camera. Their work shows the feasibility of tracking 3D eye gaze and outlines a control framework for gaze-contingent motion stabilization, however, their experiment is relatively simple compared to a surgical scene (single sphere target). They also used eye gaze to determine the depth of soft tissue in live and simulated scenes. Subjects followed predefined paths across images, and the reconstructed depths closely followed the actual surfaces with a regression ratio error of 0.103 ± 0.0912 (size of error in terms of distance units was not specified).

In [45], Mylonas et al. describe the use of eye gaze tracking for channeling motor movements. They use a haptic device for interacting with a virtual surgical tool shown within the da Vinci surgeon console. In their system, gaze contingent motor channeling is accomplished by applying a haptic force to the robotic manipulator according to the distance from the surgical tool tip to the users point of fixation. Both 2D and 3D experiments were conducted. In the 2D experiment, subjects were asked to use the tip of the surgical tool to track a moving target. They identified the optimal haptic force profile to be a spring force as it had the smallest tracking error. In the 3D experiment, subjects ablated fiducials on a deforming synthetic cardiac tissue. In addition to motor channeling, a safety boundary was set which is updated based on the subjects fixation point. The boundary was set to be parallel to the endoscope cameras x-y plane, and located at the depth of the fixation point. A conical channel was set at each fiducial based on which fiducial was closest to the subject's fixation points. They observe a 30% accuracy improvement with gaze-contingent motor channeling in the 2D case. For the 3D experiment, they observe an accuracy improvement of 37.72%. They expand on this work in [46], in which the cognitive demand of gaze-contingent motor channeling is assessed with Functional Near-Infrared Spectroscopy (FNIRS). They found that performing tasks using eye gaze was more cognitively demanding. They propose that subjects

were able to focus on task planning and quality, which requires more use of the prefrontal cortex, thus resulting in the increase in cognitive demand.

2.3.1 da Vinci Research Kit Framework

The da Vinci Research Kit (DVRK) is an open-source system that allows researchers to access and control components of the first-generation da Vinci surgical robot. The DVRK is comprised of electronic, firmware, and software components.

For each robotic "arm" (slave arm, master manipulator, endoscope manipulator, there is one controller box used to interface with the arm (see Figure 2.5. With three slave arms, two master manipulators, and one endoscope manipulator our system has 6 controller boxes. Within each controller box are two sets of electronic boards. Each set has a Quad Linear Amplifier (QLA) board and a IEEE-1394 Field-Programmable Gate Array (FPGA) Controller board. The QLA board drives four DC motors, and has Analog-to-Digital (ADC) converters as well as Digital-to-Analog (DAC) converters for setting and monitoring the motor current. The FPGA Controller board controls the motors, and communicates with the an external Linux computer with the FireWire protocol. Closed loop control of the robotic arm is achieved with sensor feedback connected to the FPGA board.

The software for the DVRK is built upon the Center for Computer-Integrated Surgical Systems and Technology ("cisst") package and Surgical Assistant Workstation (SAW) open-source libraries developed at the Johns Hopkins University (JHU). The cisst package (named after the Center for Computer-Integrated Surgical Systems and Technology at JHU) contains low level functionality for computer-assisted interventions. The SAW libraries contain robotic and imaging functionality for computer-assisted surgical applications. A console GUI application is provided which contains an interface for sensor monitoring, PID control, and teleoperation status. An example of the GUI is shown in Figure 2.4.

The Robot Operating System (ROS) framework is also implemented with the da Vinci Research Kit. The main components of ROS are the communication infrastructure, robot-related functions, and development tools. The communication infrastructure sends messages between different "nodes", which may be different ROS-enabled applications. Communication between nodes is established for dif-

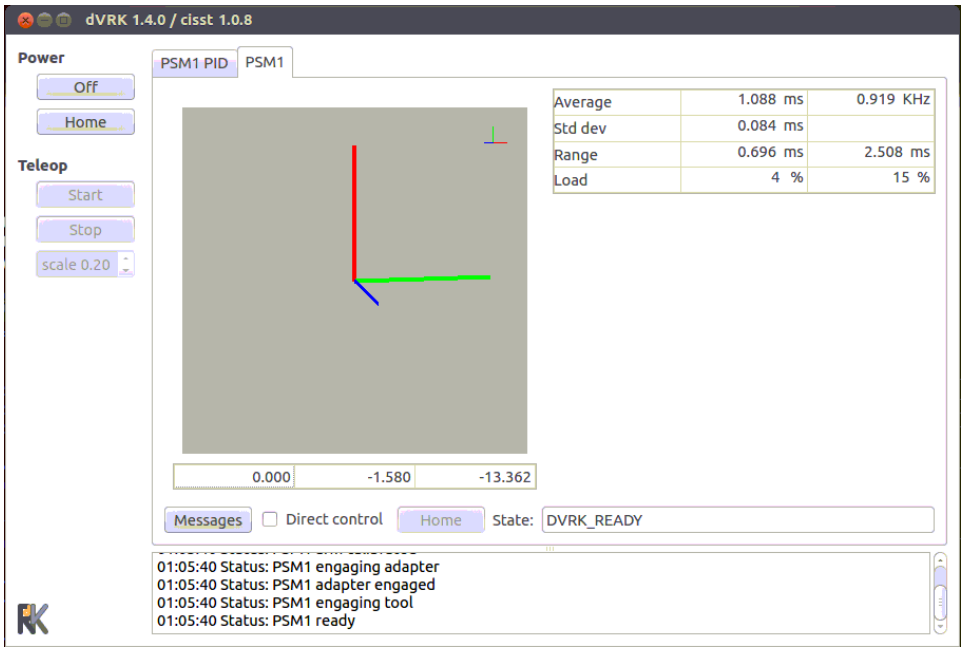


Figure 2.4: The da Vinci Research Kit console user interface (showing robot position).

ferent "topics", which are names of data streams such as robot position. Each node may publish topics and subscribe to topics. Additionally, "services" may be set up between nodes when a structure is needed for request-based two-way communication (one node requests information and the other node replies). A ROS "Master" application coordinates all of the communication in between nodes. Each node will register topics with the ROS Master, which then initializes the communication between publishing and subscribing nodes. The ROS master additionally contains a Parameter Server, which stores data in a central location. Data stored in the Parameter Server is sorted by key.

The robot-related functions in ROS includes pose estimation and a geometry library for calculating coordinate transforms. The "tf2" ROS package is used for pose estimation at any point in time. It monitors the coordinate frames within the robotic system. ROS also includes imaging processing packages for reading and capturing images and performing camera calibration. ROS contains a package for interfacing ROS images with OpenCV functions.

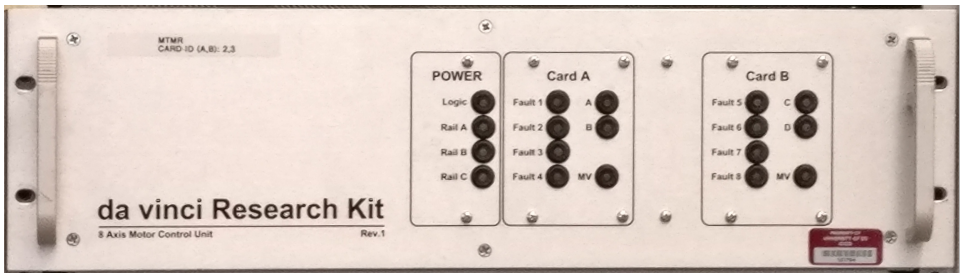


Figure 2.5: A da Vinci Research Kit controller box.

The development tools in ROS includes simulators and GUI development tools. The simulation tool is especially useful for prototyping applications prior to executing them on the full robot system. The simulator platform is Gazebo, which is a separate toolkit for 3D rendering of robot simulations. Gazebo communicates with ROS nodes through Transmission Control Protocol/Internet Protocol (TCP/IP) communication. Graphic rendering is carried out with the Ogre 3D rendering engine. The simulator tool is implemented in the DVRK library, which contains the mesh models for Patient-side Manipulator (PSM) and Master Manipulator (MTM) robot arms as well as various tools. The Graphical User Interface (GUI) development tools in ROS are built with Qt. The GUI is modular, and consists of various plugins which are shown as dockable windows within a main application. Existing GUI plugins can be implemented for any project, or a custom GUI plugin can be written.

ROS can be used with programs written in C++, Python, and Matlab. There may be a variety of applications running in a ROS system,for example one node can be a Qt C++ application communicating with the robot, while another node is a Python script computing a path trajectory for the robot to follow.

Chapter 3

Gaze Tracker Design

3.1 System Overview

In this section, the design of an eye gaze tracker for the da Vinci Surgical System is described. The purpose of the eye gaze tracker is to determine where the surgeon is looking within the da Vinci surgeon console. The hardware of the eye gaze tracker contains a set of cameras and IR Light Emitting Diodes (LEDS) directed at the surgeon's eyes. The IR LEDS cause light reflections on the surface of the cornea which are referred to as "glints" in this thesis. We assume that the surgeon does not move their head when operating with the da Vinci because a headrest in the surgeon console stabilizes the surgeon's head position. Also, the surface of the cornea is assumed to be approximately spherical. Therefore, as the surgeon looks around the surgical scene, their pupil moves but the IR light reflections on the cornea surface remain at stable positions. The eye gaze tracker can calculate the point of gaze using the vector between the pupil position and the position of IR light reflections. This "pupil-glint" vector is mapped to 2D or 3D eye gaze positions through a calibration procedure. Therefore, the software algorithm of this eye gaze tracker has two main parts- image processing for finding the pupil and glint eye features, and a calibration process for mapping the pupil-glint vector to the 2D or 3D eye gaze position.

The full eye gaze tracking system consists of an eye gaze tracker and additional peripheral devices for communication with the da Vinci Research Kit. The eye gaze

tracker is connected to a computer running the eye gaze tracking software. This computer also outputs a display on the da Vinci surgeon console. A secondary computer runs the DVRK software, and communicates with the eye gaze tracking computer using sockets via the TCP/IP protocol and ROS. The DVRK computer communicates via the Firewire protocol to the DVRK controllers, which output power to the individual da Vinci robotic arms. This system is shown in Figure 3.1.

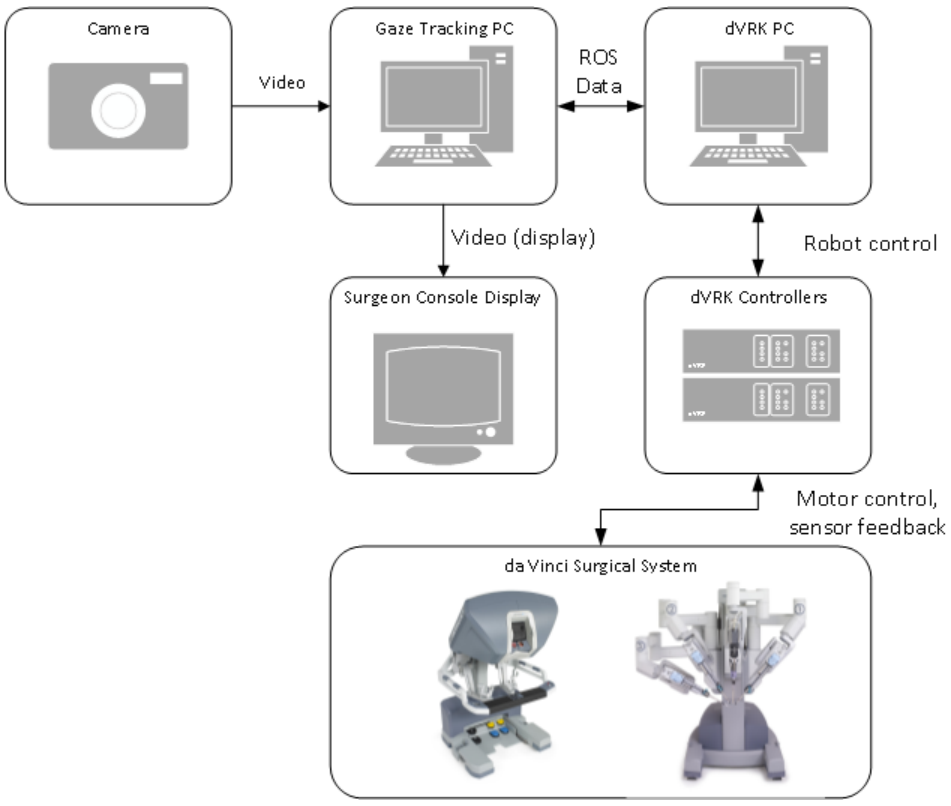


Figure 3.1: System overview diagram showing components and communication channels.

3.2 Hardware

The eye gaze tracker is designed to fit onto the da Vinci first generation surgeon console. Within the context of a tele-operated surgical robot with a stereo viewer, one eye gaze tracking solution is to integrate an eye gaze tracker within the stereo viewer console [43]. While this is a viable solution when designing a stereo viewer with eye gaze tracking functionality, embedding an eye gaze tracking system within an existing stereo viewer such as the da Vinci surgeon console requires invasive modification of the surgical system. A retro-fit design was selected over placing the gaze tracker inside the surgeon console to avoid invasive modifications of the da Vinci. This additionally makes it possible for other groups to easily use this design, and enables applications in which the gaze tracker is mounted on glasses-like frames around the eyes. The frame consists of a compartment for cameras and holes for placing LEDS. The eye piece was modelled in Solidworks, and prototypes of the eye tracker frame were 3D printed. The cameras are placed so that they are just below the surgeons eyes.

The requirements for the eye gaze tracker are as follows:

- Captures a perpendicular view of the surgeon's eyes
- Does not disrupt the surgeon's field of view
- Safe for surgeon to use for extended periods of time

Firstly, it is important for the eye gaze tracker to capture a perpendicular view of the user's eyes so that the eyelid does not obstruct the pupil. Also, if the eye gaze tracker is placed at a sharp angle then the calibration may be less accurate since pupil's range of motion in the vertical direction is smaller. Secondly, the eye gaze tracker should not reduce the surgeon's field of view within the stereo console. Lastly, the eye gaze tracker is designed to use IR light LEDS which are directed at the eyes. The light emission of the LEDS should be at a safe level to protect the surgeon's eyes.

In the following sections, the main design iterations of the eye gaze tracker are described in detail. Each hardware design iteration had associated software iterations that are described in 3.4.

3.2.1 Hardware Version #1

The camera used in the first design iteration is the Leopard Imaging LI-OV5640-USB-72. This camera runs at a frame rate of 30 Frames Per Second (FPS) and a resolution of 640x480 pixels. Two cameras were placed on the eye gaze tracking frame (one for each eye), which is shown in Figure 3.2. To illuminate the eye, two Hamamatsu L1915 IR LEDS were placed in the center of the frame. Because the LEDS were placed in the center of the frame and had a wide angle of emission, both LEDS were visible as glint reflections on the cornea. A Kodak Wratten 87C IR bandpass filter is placed on top of the camera lenses. An example of the image output from the eye gaze tracker is shown in Figure 3.3.

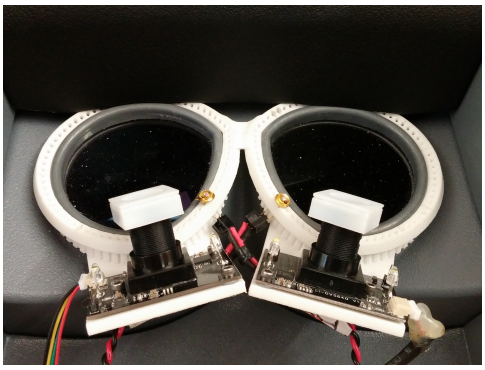


Figure 3.2: First eye gaze tracking iteration.

Figure 3.3: Image captured by first hardware setup.

The shortcomings of this design are largely due to the camera. The camera has

a large form factor; the base of the camera is 4 cm wide and 3 cm long, and the height of the camera is 2.28 cm. Placed underneath the surgeons eye, a user has to place their head farther from the eye piece to avoid making contact with the eye gaze tracker. Additionally, the frame rate of the camera is limited to 30 FPS.

3.2.2 Hardware Version #2

For the second design iteration, the Leopard Imaging LI-OV580-STEREO camera was used. This camera improves on the previous system because it has a frame rate of 100 FPS for a resolution of 640 x 480. Furthermore, the camera is a stereo camera with two camera sensors connected to a central camera board. The two cameras are synchronized on the camera device and only one Universal Serial Bus (USB) 3.0 cable is used to connect with the computer. The eye gaze tracker is shown in Figure 3.4, and an example image captured by the cameras is shown in Figure 3.5. A benefit of this configuration is that the central camera board can be placed out of view underneath the surgeon console. The cameras are also smaller than the previous design, the base of the camera board is 2.6 x 1.8 cm and the height of the camera is 2.3 cm. This allows the cameras to be placed at the level of the rim of the surgeon console eyepiece. Four Vishay Semiconductor TSHF5210 IR LEDS were placed on the eye frame, two for each eye. These LEDS had a smaller viewing angle than the previous design, and so only two glints were visible per eye. Again, Kodak Wratten 87C IR bandpass filters are placed on top of the camera lenses.

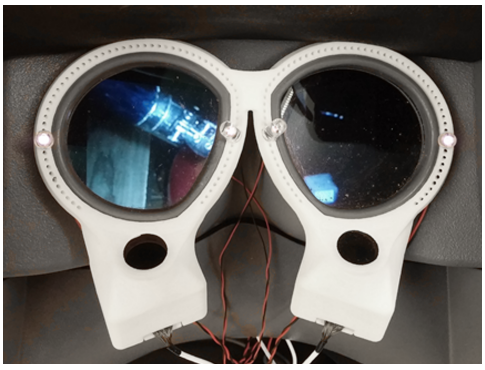


Figure 3.4: Second eye gaze tracking iteration.

The shortcomings of this design are that there is a poor angle of the eye, and the

Figure 3.5: Image captured by second hardware setup.

camera has poor light sensitivity when running at speeds faster than 30 FPS. Due to the smaller camera size, users are able to place their heads much closer to the eye gaze tracker. However, this results in a sharp angle of the eye, and the eyelid is almost always covering part of the pupil. The camera does not have good light sensitivity when running at fast frame rates (60 FPS or higher). Adding additional LEDS did not solve this issue. As a result the pupil is not as easy to segment from the image, resulting in poor feature detection.

3.2.3 Hardware Version #3

The most recent hardware iteration of the eye gaze tracker uses the Leopard Imaging LI-OV580-OV7251ST stereo camera, with two Leopard Imaging LI-OV7251M-FF-105H camera sensors. This camera is much smaller than the previous designs, and the camera sensor boards have a base size of 0.6 x 0.6 cm, and a height of 0.58 cm. A significant difference in the frame design for this eye tracker is that the cameras are placed within the stereo console eyepiece. The cameras were placed such that they do not block the surgeon's point of view and the display screens within the surgeon console are still fully visible. To illuminate the eyes and obtain glint reflections, six IR LEDS with a peak wavelength of 890 nm were placed on the eyepiece directed at the eyes. Three LEDS were used for each eye, and were evenly space across the bottom of the eyepiece. The middle LED is placed near the top of the camera. Three LEDS were used in order to provide even illumination of the eye. Each LED was placed in series with a 270 Ω resistor. This hardware version

has no bandpass filter as the user's head is close to the camera and blocks most of the visible light.

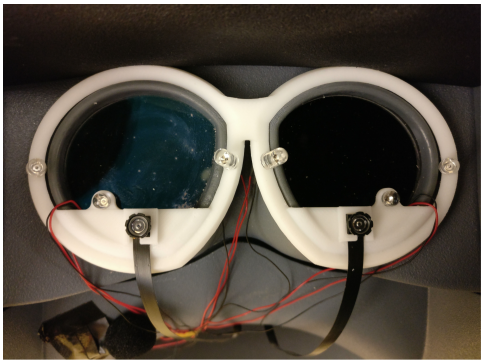


Figure 3.6: Third eye gaze tracking iteration.

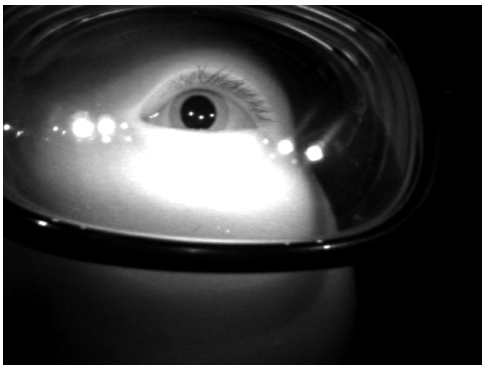


Figure 3.7: Image captured by third hardware setup.

Table 3.1 shows a summary of the cameras and camera settings used in the three iterations of the eye gaze tracker.

3.3 Software System Architecture

3.3.1 Eye Gaze Tracker Software

Software was developed and and implemented on a computer running Ubuntu 14.04, with an Intel Core i7-6700K processor (8 cores). The DVRK software is

Table 3.1: Eye Tracker Camera Models

Camera Device	#1 Leopard Imaging LI-OV5640- USB-72	#2 Leopard Imaging LI-OV580- STEREO	#3 Leopard Imaging LI-OV580- OV7251ST
Frame Rate	30 FPS	100 FPS	100 FPS
Resolution	640x480 pixels	640x480(x2) pixels	640x480(x2) pixels
Format	MJPEG	YUV	RAW
Imaging Sensor	OV5640	OV4689	OV7251
Focal length	2.8 mm	2.8 mm	1.3 mm
F number	2.6	2.0	2.8
Stereo	No	Yes	Yes

compatible with Ubuntu 12.04, 14.04, or 16.04. Ubuntu 14.04 was used because it was the latest version when the newest version of the eye gaze tracker software was being developed, and it has long-term support from Canonical Ltd. The Graphics Processing Unit (GPU) used was an NVidia GTX-660x. Software was maintained under version control using Subversion (SVN). The Unfuddle project management tool was used to store the repository as well as keep track of tasks and issues for the project. The eye gaze tracking software application, GazeTrackGUI, was developed in the C++ language using Qt for graphical user interface elements, OpenCV for image processing, and Boost for numerical calculations. The graphical user interface for the software is shown in Figure 3.8.

The GazeTrackGUI software, has the following functionalities:

- Image or video source selection for left and right eyes
- 2D calibration control and configuration
- 3D calibration control and configuration
- Accuracy evaluation mode
- Data log control and configuration
- ROS communication
- Display of gaze feature detection algorithm steps

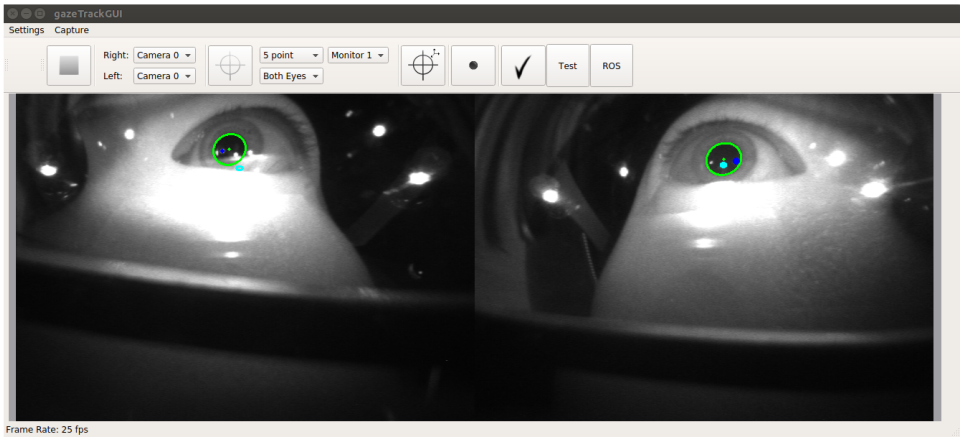


Figure 3.8: GazeTrackGUI software user interface.

A typical use case of the software is shown in Figure 3.9.

The software system uses the Boost Threading library to run several processes simultaneously and achieve real-time eye gaze tracking. A single thread interfaces with the Leopard imaging camera and continuously grabs frames at 100 FPS. The output of the camera is a 16-bit monochrome image, and is converted to a 12-bit image using Leopard Imaging's Software Development Kit (SDK). The gaze tracker algorithm runs on another thread, and pulls the most recent camera frame from the camera thread synchronized with a mutex. The mutex locks certain resources when a thread is using it so that only one thread can access the locked resources at a time. A threading diagram is shown in Figure 3.10.

In total, the gaze tracking algorithm takes 10-20 ms to finish processing a single frame, depending on whether the pupil location is known in previous frames, or if the algorithm has to search the entire image for the pupil. Details for the eye gaze tracking algorithm are described in Section 3.4.

Visual computing methods used for eye gaze tracking were prototyped first in Matlab or Python, and then ported to C++ for the final software. Prototyping in this manner was carried out in order to quickly test algorithms while maintaining fast and efficient code.

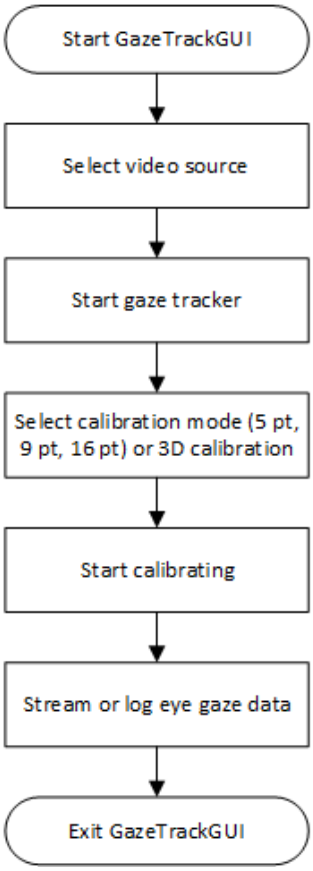


Figure 3.9: GazeTrackGUI software user interface.

3.4 Eye Feature Detection

The algorithm for pupil detection was developed in two main iterations. First, a thresholding algorithm was pursued, adapted from [20]. This algorithm will be described in order to emphasize the motivation for the final eye feature detection algorithm.

3.4.1 Initial Eye Feature Detection Algorithm

In this algorithm, an adaptive threshold is used to roughly segment the pupil and then the pupil contour is improved. Each frame is first converted to greyscale. A histogram of the image intensity is generated, and a threshold is set such that a

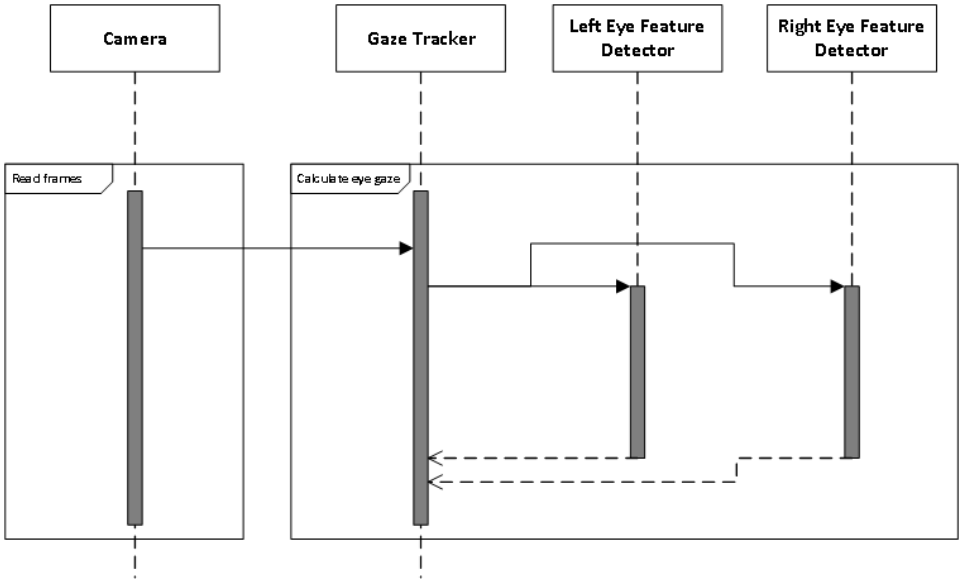


Figure 3.10: Diagram of camera and eye gaze tracking thread processes in eye gaze tracking software.

pre-determined percentage of pixels are set to black. This creates a binary image in which the pupil is a black elliptical contour.

The next step is to find all the contours in the binary image, and from these contours the pupil contour can be found. The contour detection implemented is the border following algorithm proposed by Suzuki and Abe in [61]. This contour detection algorithm is implemented in OpenCV. The output of the border following algorithm is a list of all closed contours in the binary image.

The most circular contour is determined by calculating the isoperimetric quotient IQ:

$$IQ = 4A/P^2 \tag{3.1}$$

Where A is the area and P is the perimeter. The contour with the isoperimetric quotient that is closest to 1 is assumed to be the pupil.

An ellipse is then fit to this contour using an ellipse fitting algorithm described by Fitzgibbon *et al.* in [11]. The "algebraic distance" algorithm is implemented, which applies a least mean square objective function to fit an ellipse to the points

on the pupil contour.

The centre of the fitted ellipse is taken to be the rough center of the pupil. This rough pupil location needs to be further refined for a more accurate estimation of the pupil center. From the original greyscale image, the mean intensity of the pixels within the rough pupil contour is calculated. This mean intensity is then applied as a threshold limit to create another binary image with the pupil as a black region. Again, all of the contours in the binary image are found. The contour which is closest to the rough pupil location is used as the refined pupil contour. An ellipse is then fit to this contour, and the pupil center is the center of the fitted ellipse.

Using the fine pupil contour, the image is then reduced to a Region of Interest (ROI) surrounding the extracted pupil in order to find glints. The glints are detected by applying a threshold at a value tuned experimentally across multiple users and LED intensities. A threshold of 200 was selected (the highest image intensity is 255) to create a binary image in which the glints are white. The contours within the binary image are detected, and the list of potential glint contours was matched to a pre-determined glint pattern to find the glints.

With this threshold-based pupil detection method, the pupil can be detected but the threshold is sensitive to variations in lighting conditions. Dark shadows cast near the pupil also distort the detected rough pupil. The main issue with this method was that the adaptive threshold method implemented still requires a pre-set value for the percentage of pixels to set to black. As the users eyes are close to the cameras, variations of facial features such as eyelash thickness or strong shadows require adjustment of the pre-set percentage value. There was not a universal value for the percentage of pixels to set to black which would work for all users. Additionally, after changing the camera used for the eye gaze tracker, the pre-set percentage value needed to be re-adjusted because the camera had a different light sensitivity. This established the need for an eye gaze tracking algorithm that would be robust to lighting and intensity changes.

3.4.2 Final Eye Feature Detection Algorithm

For the current pupil detection algorithm, we selected techniques that rely on image gradients rather than image intensity values, which allows this method to function

under various lighting conditions and with different users. All parameters used in the eye gaze tracking algorithm are listed in Appendix A. The major steps in the pupil detection algorithm are shown in Figure 3.11:

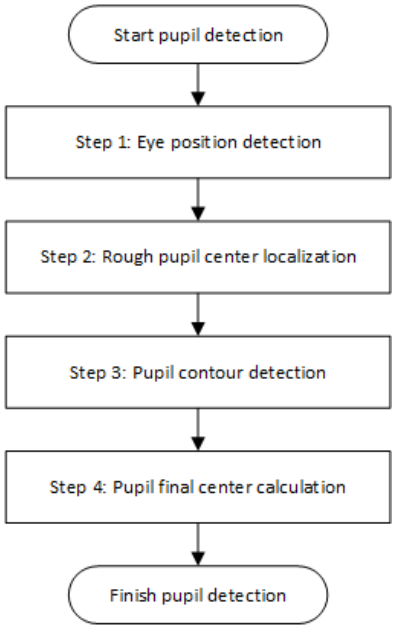


Figure 3.11: Main steps of the pupil detection algorithm.

First, we detect the position of the eye in order to reduce the search space for the pupil. Then, the rough center of the pupil is located. Using this center, we find the pupil contour, and fit an ellipse to the contour to achieve a more accurate pupil center. The detailed implementation of these steps are described in the following section.

Step 1: Eye position detection

The raw image frame obtained from the gaze tracker cameras has a resolution of 640x480 pixels, and is monochromatic. The black pixels have a value of 0, and white pixels have a value of 255. A sample frame is shown in Figure 3.12.

A Haar cascade detector is first used to detect the presence of an eye. We use the Haar detector to find the approximate location of the eye, and then use computer vision approaches to find the eye features. The Haar cascade detector can reliably find the eye, however it is slow in processing the images and does not

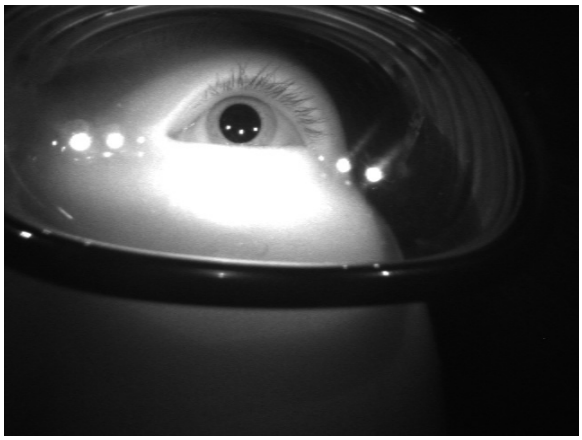


Figure 3.12: Raw image frame from gaze tracker camera (left eye).

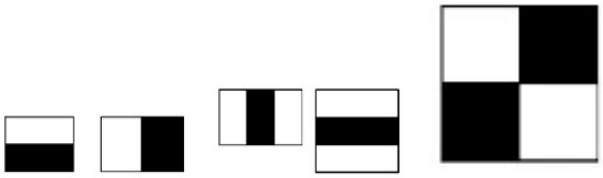


Figure 3.13: Haar-like features used by OpenCV: (a) edge features, (b) line features, (c) corner features[1].

accurately locate the center of the pupil. Thus, in subsequent frames, the region of the eye is centred on the last known position of the pupil. The Haar detector only implemented again if the pupil is lost for a certain number of frames.

The OpenCV implementation of the Haar classifier is used, which is based upon [36, 71]. The Haar classifier is a machine learning approach to object detection in computer vision. A Haar feature is calculated using the feature templates shown in Fig 3.13. The sum of the black region is subtracted from the sum of the white region. For each image, these features are calculated for many positions and scales. In the training step, a set of positive images (images of eyes), and a set of negative images (not eyes) are processed to find Haar features. For detection, a cascade of features is applied in several stages. At each stage if the detection fails then the process is aborted.

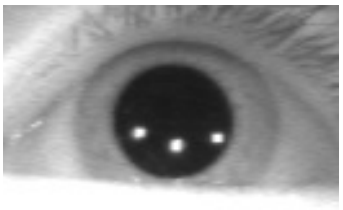


Figure 3.14: Rough pupil region of interest.

The Haar cascade classifier is trained with images matching the object which needs to be recognized. A pre-trained eye detector from the OpenCV library was used. The

Once the pupil is found, the region surrounding the pupil is extracted, as shown in Figure 3.14.

Step 2: Rough pupil center localization

The rough center of the pupil is found by searching within the region of interest. The outcome of this step should always be a point at the center of the pupil, but not necessarily the accurate position of the pupil center. A refinement step is carried out once the rough pupil location is known.

The rough pupil center detection is achieved through a gradient-based method proposed by Timm and Barth in [67, 68]. This algorithm was selected as it takes into account the curvature and darkness of the pupil while not requiring the use of an intensity-based threshold. The implementation in our eye gaze tracker closely follows the method proposed in [68], with adjustments of specific parameters to work with our system.

The premise of the gradient eye localization method is that the displacement vector from the centre of a circular object to a point on its perimeter should be in the same orientation as the gradient vector at the perimeter point. That is, for a potential circle centre position c, the unit vector d_i from c towards any point x_i and the gradient vector g_i at point x_i will be directed in the same orientation if x_i is at the boundary of the circle. This is illustrated in Figure 3.15.

For an image with N pixels, an objective function $f(c)$ can be defined as:

[1] Image adapted from docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html.

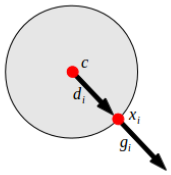


Figure 3.15: Diagram illustrating the premise of the gradient eye localization method. Point c is the center of the circle, point x_i is a point on the boundary of the circle. The vector d_i points from c to x_i and g_i is the image gradient at point x_i.

$$f(c) = \frac{1}{N}\sum_{i=1}^{N} w_c (d_i^T g_i)^2, \tag{3.2}$$

$$where\ d_i = \frac{x_i - c}{\|x_i - c\|_2}, \forall i : \|g_i\|_2 = 1. \tag{3.3}$$

The objective function sums the values of the dot product between d_i and g_i for every pixel x_i in the image. The low intensity of the pupil is taken into account by multiplying by the normalized inverse of image intensity w_c.

The optimal centre c_r is calculated by finding the position c which results in the highest value of $f(c)$:

$$c_r = \underset{c}{\operatorname{argmax}} \frac{1}{N}\sum_{i=1}^{N} w_C (d_i^T g_i)^2 \tag{3.4}$$

The map of $f(c)$ calculate across the entire image is shown in Figure 3.16. The center of the pupil is can be seen as the brightest area in the map.

To calculate the optimal centre using Equation 3.4 is computationally expensive because Equation 3.2 needs to be calculated for every potential center position c. To reduce the processing time for finding the pupil centre, a gradient ascent approach is used which iteratively converges on the maximum value of c. This method is described by Timm and Barth in [68]. Taking the derivative of Equation 3.2 with respect to candidate pupil center positions c results in the following equation:

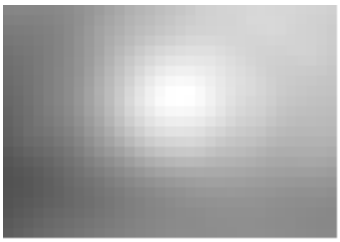


Figure 3.16: Output of gradient eye localization method where the brightest pixel is the optimal pupil center.

$$\frac{\partial f(c)}{\partial c} = \frac{2}{N}\sum_{i=1}^{N}\frac{(x_i - c)e_i^2 - g_i e_i n_i^2}{n_i^4}, \tag{3.5}$$

$$where\ e_i = (x_i - c)^T g_i, n_i = ||x_i - c||_2. \tag{3.6}$$

An initial position for pupil center c_0 is set as the output of the Haar algorithm, or the position of the pupil center in a previous frame if it is known. For a position $c = (c_1, c_2)$, the gradient direction ∇f is found as:

$$\nabla f = \begin{bmatrix} \frac{\partial f(c)}{\partial c_1} \\ \frac{\partial f(c)}{\partial c_2} \end{bmatrix}. \tag{3.7}$$

A range of step sizes along the direction of ∇f are tested, as proposed in [68]. They used a range of 10 step sizes from 10^{-2} to 10^5. We tested a range of 10 exponentially increasing step sizes from $10^{0.001}$ to $10^{0.5}$. Our image scale is different from the implementation in [68], so they are not directly comparable. For each step size s, a new pupil center location c is calculated as:

$$c = s\nabla f + c_0 \tag{3.8}$$

The location of c yielding the greatest value of $f(c)$ is then set as c_0, and this process is repeated until convergence on the final rough pupil center position. This is achieved when the difference in c position between iterations is less than 1 pixel.

Another way in which processing time is decreased is any image gradient magnitude $g = (g_1, g_2)$ lower than a threshold $thres_g$ is not considered in calculations

when computing $f(c)$. The mean μ_g and standard deviation σ_g of all gradient values within the frame is computed. Then, the threshold is set as:

$$thres_g = \mu_g - 0.3\sigma_g. \tag{3.9}$$

Furthermore, the image is sub-sampled to 25% of the original image size to reduce the number of pixels in the image. This introduces a reduction in precision, as one pixel in the sub-sampled image represents four pixels in the original image. When mapping the final pupil center c to the original image frame, the final pupil center c_r is calculated as:

$$c_r = \frac{1}{a}c + \frac{1}{2a}, \tag{3.10}$$

where a is the scaling factor ($a = 0.25$ for our method).

Step 3: Pupil contour detection

Once the rough pupil center has been determined, a more accurate approximation of the pupil center is found by fitting an ellipse to the pupil contour. We base our method on the Starburst method [33] to detect the contour of the pupil.

The premise of the Starburst algorithm is to find potential pupil edge points by radially tracing lines extending from an approximate pupil center, and noting edge points when a large intensity difference between pixels in each ray is detected. The original Starburst algorithm conducts additional ray searches from each detected edge point back towards the pupil center in order to converge on a pupil center. However, this is sensitive to gradients from other eye features (eyelashes, shadows from eyelid, iris) and as our algorithm already has an estimate for the center of the pupil the search is only performed stemming from the rough pupil center.

The image of the eye is first processed in order to reduce the effect of noise. A morphological open operation is applied followed by a Gaussian filter (Figure 3.17). We then use the Canny edge detector to detect edges in the image. Edges due to glint reflections are masked. This is accomplished by generating a binary image where all values above a glint threshold g_{thres} are white. The mean μ_{im} and standard deviation σ_{im} of the image intensity values are calculated. The glint threshold value $thres_{glint}$ is then calculated as:

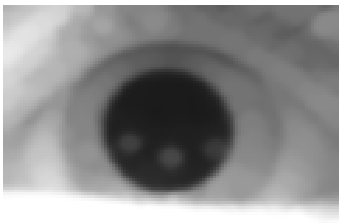
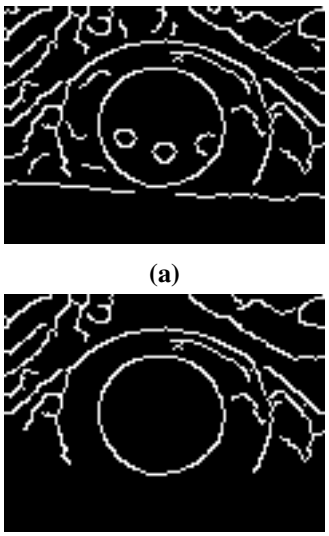


Figure 3.17: Filtered region of interest.



(a)

(b)

Figure 3.18: Edge image (a) after the Canny edge detector, and (b) after removal of glints and small lines.

$$thres_{glint} = \mu_{im} + 1.5\sigma_{im} \text{ where } thres_{glint} \in (200, 250). \tag{3.11}$$

Starting from the rough pupil center, c_r, each point $x_{r,\theta}$ is evaluated along rays of length $r \in (0, maxRay)$ pixels and for each angle $\theta \in (0, 360) degrees$.

Each ray is searched radially until a maximum radius is reached. The maximum radius *maxRay* is set to 100 pixels by default, but if a pupil has been found in a previous frame, then the maximum radius length is set to 1.5 times the length of the major axis for the previously detected pupil. A size of 100 pixels is much larger

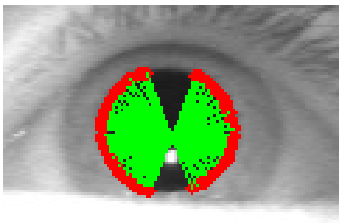


Figure 3.19: Starburst algorithm, with detected features shown in red, and ray traces shown in green.

than the typical pupil size (determined experimentally). While tracing along each ray, a point $x_{r,\theta}$ is only evaluated if there exists a edge in the edge image with a positive intensity difference ($x_{r+1,\theta} - x_{r-1,\theta} > 0$). If so, then the position $x_{r,\theta}$ is added to a list of potential points. Once two potential points have been found, the search is stopped for the current angle, and the point with the higher intensity difference is recorded as a potential feature point. The search stops at two potential points because the amount of edges within the cornea is sparse, and detecting more than two edges along a ray may contain edges on the eyelid. The feature points in 10% of the top and 10% of the bottom are removed to avoid detecting points on the eyelids (see Figure 3.19).

Step 4: Pupil final center calculation

After performing a starburst search, there is a set of potential pupil edge points. The RANSAC method [10] is used in order to fit an ellipse to the edge points. RANSAC is used as it is able to effectively fit a model to data in the presence of many outliers. We implemented a version of the RANSAC algorithm modified for pupil detection proposed by Swirski et al. in [62]. The RANSAC algorithm iteratively fits the ellipse model to a subset of randomly selected data points. Five points are used as this is the minimum needed in order to define an ellipse. The conic equation describes the ellipse $Q(x,y)$ with coefficients A,B,C,D,E,F,

$$Q(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F. \tag{3.12}$$

First of all, early rejection criteria are defined which forces the algorithm to stop the current iteration and select a new subset of five data points. There are

three conditions that must be met; the ellipse center must be within range of the previously known pupil center, the size of the ellipse must be a reasonable size for a pupil, and the image gradients must be in agreement with the ellipse gradient for the five data points. We define the maximum range from a previously known pupil to be 30 pixels, and a reasonable pupil size to be greater than 10 pixels and smaller than half of the ROI size (4800 pixels). These requirements are quite loose because the RANSAC algorithm will find the optimal ellipse with a support function, and the early rejection requirements are only used to prevent computation of the support function for clearly incorrect ellipses. The agreement of the ellipse and image gradients is determined by calculating the dot product between the ellipse $\nabla Q(x,y)$ and image gradients $\nabla I(x,y)$ for each point (x,y):

$$\nabla Q(x,y) \cdot \nabla I(x,y) \leq 0 \tag{3.13}$$

The ellipse fit is improved by calculating inlier data points which are near the perimeter of the ellipse, and iteratively fitting a new ellipse to the inlier points. The distance between data points and the ellipse perimeter is estimated using the gradient weighted algebraic distance, EOF_2, defined by Rosin in [52]:

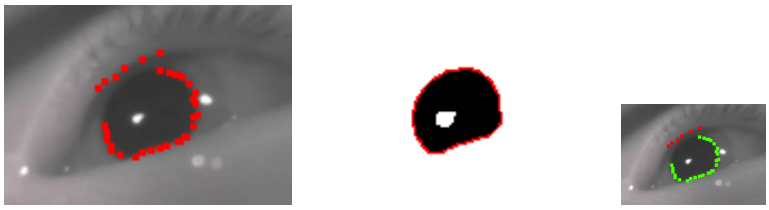
$$EOF_2 = \frac{Q(x,y)}{|\nabla Q(x,y)|} \tag{3.14}$$

The value of EOF_2 is only an approximation of the distance from a point (x,y) to the ellipse perimeter. Thus, a scaling factor α is used to normalize the error of being one pixel from the ellipse perimeter to 1. Thus, the error from each data point (x, y) to the ellipse perimeter is calculated as:

$$error = \alpha \frac{Q(x,y)}{|\nabla Q(x,y)|} \tag{3.15}$$

The "inlier points" are defined as any feature points for which $error < maxerror$. For our system, the value of *maxerror* is set to 3 pixels. The *error* value for each feature point is calculated, and all inlier points are identified. A new ellipse is then fit to the inlier points. This process refines the ellipse to be in agreement with more data points.

For a camera setup with poor light sensitivity such as our second hardware



(a) **(b)** **(c)**

Figure 3.20: Intensity-based inlier refinement steps; (a) original feature points shown in red, (b) binarized image with pupil contour shown in red, and (c) final inlier points shown in green.

iteration, there may be cases where the eye lid or iris contains data points that meet the criteria for a pupil. Thus, a intensity-based inlier refinement step may be added to improve pupil detection. This step is added before the calculation of inliers. Furthermore, this step was implemented for our past eye gaze tracker versions, but was not implemented in the final eye gaze tracking algorithm because a camera with good light sensitivity was added for hardware version #3.

Figure 3.20 illustrates the process of intensity-based inlier refinement. First, the mean pixel intensity value within the candidate pupil ellipse is calculated, μ_Q. A binary threshold is applied to set all image intensity values less than μ_Q to 0. This results in a binary image in which the pupil is black, and surrounding anatomy is white. The contour of the pupil in the binary image is found. Another ellipse $Q_{contour}$ is then fit to the contour. The feature points which are inliers for $Q_{contour}$ are calculated with Equation 3.15. A new ellipse, $Q_{refined}$ is finally fit to the inlier points.

A support function is calculated for the ellipse to give a score for goodness of fit to the pupil. The RANSAC algorithm will use this score to determine the best fitting ellipse out of all the iterations. The support function takes into account the image gradient $\nabla I(x,y)$ and ellipse gradient $\nabla Q(x,y)$ for all of the inliers:

$$support(Q,I,inliers) = \sum_{(x,y)\in inliers} \frac{\nabla Q(x,y)}{|\nabla Q(x,y)|} \cdot \nabla I(x,y) \tag{3.16}$$

The support function is designed such that an ellipse has a higher value if there are many inliers, the image gradient at the ellipse perimeter agrees with the ellipse

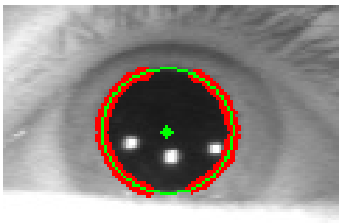


Figure 3.21: Outcome of RANSAC algorithm, with pupil center shown as green point and detected contour shown as green circle.

gradient, and if the image gradient is large, which corresponds to a strong boundary.

Lastly early termination is defined as with standard RANSAC implementations, so that if a good fit is made prior to the last iteration, the algorithm will terminate. The criteria for early termination is the number of inliers is equal to 90% of the total number of feature points. If no early termination is made, the final pupil contour is the ellipse with the highest support function. The center of the pupil is finally estimated as the center of the pupil contour.

3.4.3 Glint Detection

Locating glints is necessary as they are used for estimation of the point of gaze. For each eye, there are three glints, g_0, g_1, and g_2, see Figure 3.22. The vector from g_0 to g_1 is labelled G_1, and the vector from g_0 to g_2 is labelled G_2.

In an initial setup, a template of glint positions is created by manually locating the positions of all glints in a single frame. This template is set once in the software, and only needs to be changed if the position of the LEDS changes. The glint positions in the template are g_{T0}, g_{T1}, and g_{T2}, with $G_{T1} = g_{T1} - g_{T0}$ and $G_{T2} = g_{T2} - g_{T0}$.

The basic outline of the glint detection algorithm has two main steps. Firstly, all potential glint candidate positions are found in the frame. Then, a sorting process is used to match the candidate glints to the glint template. If the glint positions in a previous frame are known, then instead of matching to a template the glints are matched to the previous frame. This is illustrated in Figure 3.23.

Step 1: Find candidate glint contours

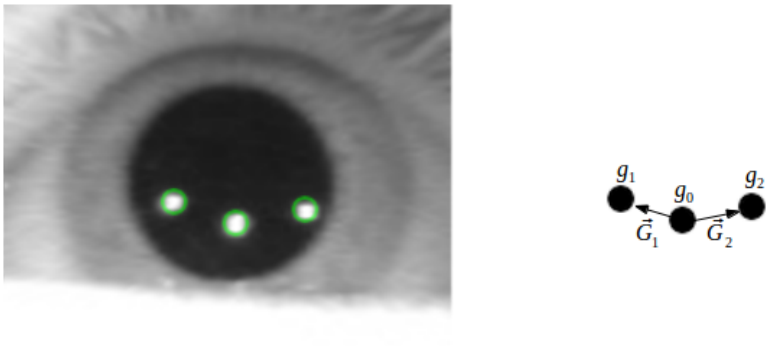


Figure 3.22: Glints highlighted in green. Glint g_0 is the central glint. The glint g_1 is closer to the nose, and g_2 is away from the nose.

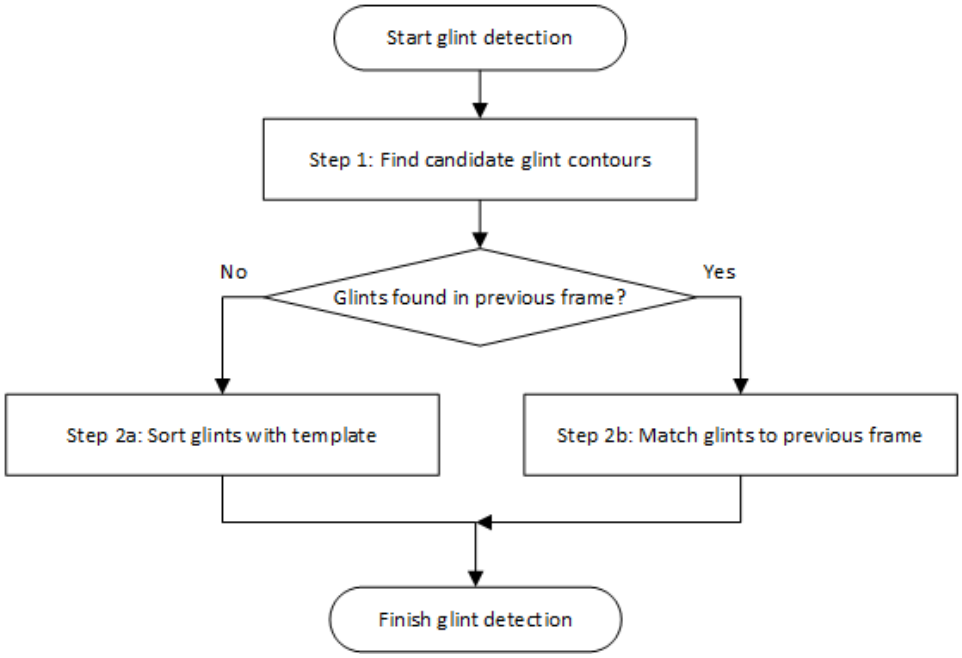


Figure 3.23: Flowchart showing the basic steps for glint detection.

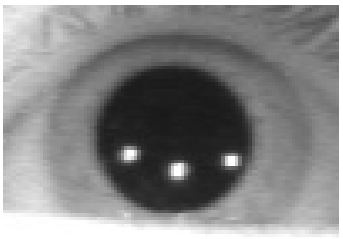


Figure 3.24: ROI set for glint detection.

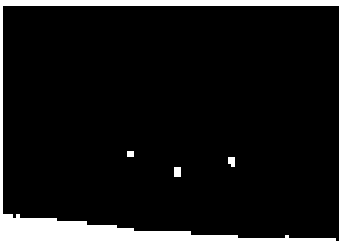


Figure 3.25: Thresholded binary image of ROI.

First a ROI is set, centered on the detected pupil. The ROI size used is 100x80 pixels. This ROI image is filtered using a Gaussian filter. An example of the ROI is shown in Figure 3.24.

A threshold is then calculated in order to convert the frame into a binary image consisting only of potential glint regions. The thresholded binary image is shown in Figure 3.25. The mean intensity level μ_{glint} and standard deviation σ_{glint} of the intensity is calculated, and a threshold $thres_{glint}$ is found as:

$$thres_{glint} = \mu_{glint} + 1.5 * \sigma_{glint} \text{ where } thres_{glint} \in (150, 250) \tag{3.17}$$

The threshold limit of $thres_{glint} \in (150, 250)$ is set because any value lower than 150 would allow eye features such as eyelid or sclera to be visible in the binary image (this is set experimentally). If the threshold is set very high (250-255), then the glint regions may be cut out from the binary image.

All closed contours in the image are then found using the border following al-

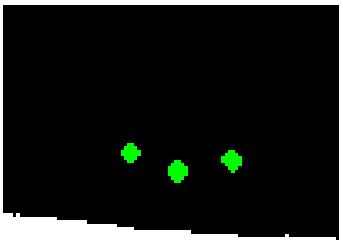


Figure 3.26: Detected contours of candidate glint regions shown in green.

gorithm described in [61]. For each detected contour, a size requirement is applied to make sure the contour area is greater than 5 pixels and less than 100 pixels. Finally, a list of N candidate glint positions $g_c = [g_{c0}, g_{c1}, ..., g_{cN-1}]$ is obtained. Detected glint contours are shown in Figure 3.26.

Step 2a: Sort glints with template

The next step is to sort through all glint candidates $g_c = [g_{c0}, g_{c1}, ..., g_{cN-1}]$ to locate the true glints and to identify which glint is g_0, g_1, and g_2. Since there are three glints, this is carried out using a cascaded method where the algorithm exhaustively searches all possible combinations in g_c for G_1, and searches for G_2 only if a potential G_1 is found.

Firstly, the glint pair G_1 with g_0 and g_1 is detected. A test vector G_1^* is drawn between each test candidate glint g_0^* and g_1^*:

$$G_1^* = g_1^* - g_0^* \tag{3.18}$$

where $g_0^* = g_{ci}$ and $g_1^* = g_{cj \neq i}$ for $i, j \in [0, N-1]$.

Two metrics are used to determine whether the test vector G_1^* matches G_{T1}: the difference in length, dR_1, and the difference in angle, $d\theta_1$:

$$dR_1 = ||G_1^*|| - ||G_{T1}|| \tag{3.19}$$

$$d\theta_1 = \arccos\left(\frac{G_1^* \cdot G_{T1}}{||G_1^*||\ ||G_{T1}||}\right) \tag{3.20}$$

The requirements for a good fit are $dR_1 < 0.70||G_{T1}||$ and $d\theta_1 < 45°$. If these test requirements are met, then the algorithm proceeds to find G_2.

Then in the same manner, a test vector G_2^* is calculated from candidate glints g_0^* and g_2^*:

$$G_2^* = g_2^* - g_0^* \tag{3.21}$$

where g_0^* is known from the previous step and $g_2^* = g_{ck \neq i,j}$ for $i, j, k \in [0, N-1]$.

To determine whether a test vector G_2^* matches the template G_{T2}, the same length requirement dR_2 and angle requirement $d\theta_2$ are used:

$$dR_2 = ||G_2^* - G_{T2}|| \tag{3.22}$$

$$d\theta_2 = \arccos\left(\frac{G_2^* \cdot G_{T2}}{||G_2^*||\ ||G_{T2}||}\right) \tag{3.23}$$

Again, if $dR_2 < 0.70||G_{T2}||$ and $d\theta_2 < 45^\circ$ then a match has been found.

The final combination of g_0, g_1, and g_2 is determined as the combination with the smallest values of dR_1 and $d\theta_1$ where $dR_2 < 0.70||G_{T2}||$ and $d\theta_2 < 45^\circ$. The final glint detection is shown in Figure 3.27.

Step 2b: Match glints to previous frame

To avoid repeating the search for glints in every frame, once the glints have been found, candidate glints in subsequent frames are matched to previous frames if they fall within a small distance threshold. The search for glint pairs described in Step 2a is only carried out at the beginning of eye gaze tracking and if the glints are lost (in a blink, or due to head movement).

3.5 2D Point of Gaze Estimation

Firstly, a 2D gaze estimation is carried out to map the pupil-glint vector to each of the monitor screens in the console. The premise of the pupil-glint vector is that when the surgeon looks around the surgical scene, the pupil will move while the glints remain relatively stationary. Thus, the vector between the pupil and glints is representative of where the surgeon is looking. In our system, as the eye is very close to the camera the assumption that the cornea is a sphere rotating about it's center is not true, and so the glints are not exactly stationary. However, the pupil-glint vector can still be mapped to a gaze position with reasonable accuracy that is

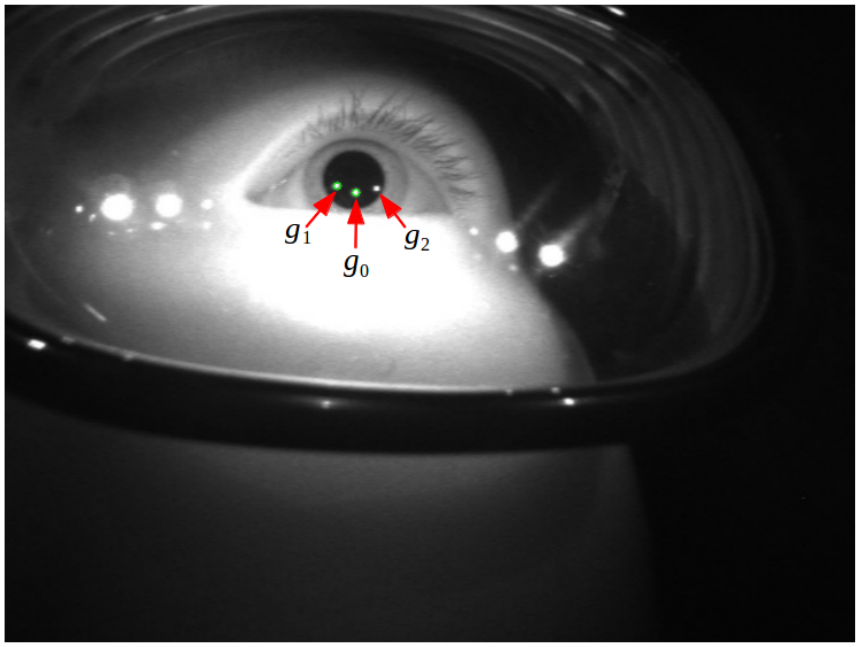


Figure 3.27: Final glint detection, with the main glint pair, g_0 and g_1 highlighted by algorithm and all glints annotated.

measured and reported in Chapter 4.

For 2D calibration, five, nine, or sixteen points are displayed on the screen and the user gazes at each point for 2.5 seconds. At each calibration target position, the target shrinks to a small point. This is important to provide a visual stimulus for a user to focus on during calibration.

The vector pixel displacement between the pupil and the glint (x, y) is approximated to on-screen pixel coordinates (p_x, p_y) with a second order polynomial function:

$$p_x = a_0x^2 + a_1xy + a_2y^2 + a_3x + a_4y + a_5 \tag{3.24}$$

$$p_y = b_0x^2 + a_1xy + b_2y^2 + b_3x + b_4y + b_5 \tag{3.25}$$

For a total of n points, this can be written in matrix form with the polynomial coefficients as variables:

$$\begin{bmatrix} p_{x0} & p_{y0} \\ p_{x1} & p_{y1} \\ \vdots & \vdots \\ p_{xn-1} & p_{yn-1} \end{bmatrix} = \begin{bmatrix} x_0^2 & x_0y_0 & y_0^2 & x_0 & y_0 & 1 \\ x_0^2 & x_0y_0 & y_0^2 & x_0 & y_0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0^2 & x_0y_0 & y_0^2 & x_0 & y_0 & 1 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \\ a_5 & b_5 \end{bmatrix} \tag{3.26}$$

The coefficients of (3.24) and (3.25) are found through least squares estimation.

Head movement compensation for the Z (moving towards or away from the camera) was accomplished according to a method described in [22]. This method is simple and effective for reducing the effect of head movement in the Z direction. The average distance d_calib between the two glints during the calibration sequence is recorded and used to normalize all future pupil-gaze vectors by a factor s defined as:

$$s = d_{calib}/d \tag{3.27}$$

where d is the distance between the two glints at the current video frame. As the user moves their head farther away from the eye gaze tracker, the distance between the corneal glints decreases, and as they move closer to the eye gaze tracker the distance between the corneal glints increases. By applying a scaling factor s, the pupil-glint vector can be scaled back to the original head placement. Head movement in the horizontal and vertical direction was not compensated, however the da Vinci surgeon console has a headrest which helps maintain head position.

3.5.1 2D POG Filtering

A simple moving average filter is applied to the 2D POG. An extra clause is added to the filter because saccadic eye movement means that eye gaze data is not always smooth. If a large change in position greater than a distance $thres_{POG}$ is detected, then the filter is reset and starts to compute the average gaze position starting from the most recent data point. For a window size of n, at each new data point i the filtered gaze position $\overline{POG_i}$ is defined as:

$$\overline{POG_i} = \begin{cases} \frac{POG_i + POG_{i-1} + ... + POG_{i-(n-1)}}{n}, & \text{if } POG_i - \overline{POG_{i-1}} < thres_{POG} \\ POG_i, & \text{otherwise (window is reset).} \end{cases} \quad (3.28)$$

3.6 3D Point of Gaze Estimation

To determine the mapping between eye features and the 3D point of gaze, a calibration procedure is performed in which the surgeon is asked to control the da Vinci surgical instruments while gazing at the tip of the tools. The motivation for using this calibration method is that the surgeon's eyes will be calibrated to the da Vinci tool coordinate frame. We propose two calibration protocols, a two-step calibration and a one-step calibration. For a two-step calibration the surgeon is first asked to perform a 2D eye gaze calibration. Afterwards they perform a 3D eye gaze calibration by looking at the surgical tools. A one-step calibration requires the surgeon to only perform the 3D calibration looking at the surgical tools.

An alternative calibration procedure could be to render 3D calibration targets within the surgeon console display. However, by calibrating in the surgical view we are able to directly compute the surgeon's gaze position within the surgical scene as opposed to a virtual scene. Additionally, our calibration scheme can be extended in future work to allow surgeons to naturally carry out a surgical task while their eye gaze is being calibrated using tool position. While the surgeon may not always be gazing directly at their tools, they will be moving their tools in close proximity to where they are looking. Therefore, with a large set of data points collected by observing the continuous tool movement and eye features, it may be possible to carry out an online calibration. This is beneficial as it removes the need for the surgeon to explicitly calibrate their eye gaze and allows them to focus on surgical procedures instead.

3.6.1 Geometric System Overview

We chose a geometric approach to 3D eye gaze estimation because the gaze tracker is used in a static environment in which the position of the user's head, gaze tracker,

and screen display are known. The 3D gaze position can be determined by considering the vergence of the left and right eyes when focusing on objects at different depths. The two eyes converge when looking at closer objects, and diverge when looking at farther objects. This estimation method is dependent on head position, and the surgeon needs to re-calibrate in case of head movement.

Prior to eye gaze calculation, there are several components of this system that need to be calibrated. The endoscope camera intrinsic and extrinsic matrices and the transformation between tool coordinate system and camera coordinate system need to be determined. These calibration procedures as well as the eye gaze calibration scheme will be discussed in the following sections.

3.6.2 Endoscope Intrinsic and Extrinsic Matrix Calibration

Firstly, the intrinsic matrices, distortion components, and extrinsic matrices of the endoscope cameras need to be found. A simple pinhole camera model is used and the conventions for camera intrinsic matrix and distortion coefficents are based on [18]. These parameters describe how a real-world point P is related to it's corresponding image pixel position p_{pix}.

A point $P = (X_c, Y_c, Z_c)$ can first be projected onto image plane coordinates $p = (x, y)$. The coordinates p represent the projection of the real-world point P on the camera sensor.

$$p = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.29}$$

Real lenses in camera systems additionally have distortion components, and the distorted image plane coordinates are noted as $p_d = (x_d, y_d)$. The two main sources of distortion are radial and tangential. The distortion components $k_c = (k_1, k_2, k_3, k_4, k_5)$ are applied to obtain a corrected (distorted) image plane coordinates p_d:

$$p_d = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6) p + d_x \tag{3.30}$$

where $r^2 = x^2 + y^2$ the tangential distortion vector d_x is:

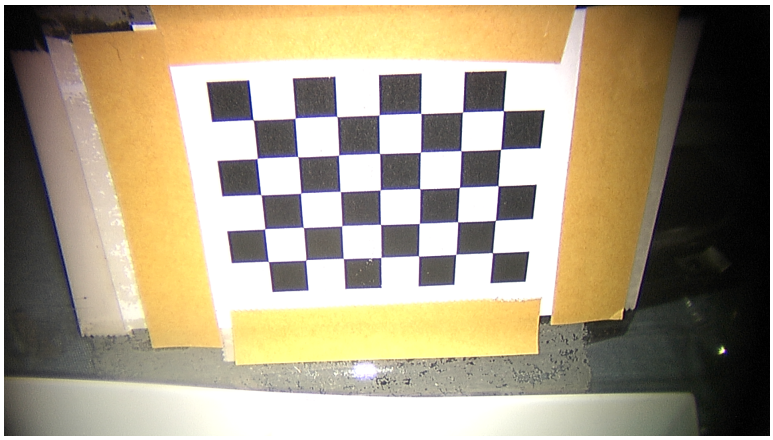


Figure 3.28: Checkerboard pattern used for stereo endoscope calibration.

$$d_x = \begin{bmatrix} 2k_3xy + k_4(r^2 + 2x^2) \\ k_3(r^2 + 2y^2) + 2k_4xy \end{bmatrix} \tag{3.31}$$

From the image sensor, the image is digitized to image pixel coordinates $p_{pix} = (u, v)$. The intrinsic matrix A is used to transform world units to pixel units as $p_{pix} = Ap_d$. The intrinsic matrix is defined as:

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.32}$$

Where f_x,f_y are the focal lengths and c_x, c_y are the principle points for the horizontal and vertical axes respectively. The focal lengths describe the scaling from real-world units to pixel units. The principle points shift the origin of the camera frame to the image frame, which is typically the top left corner of an image.

Camera calibration for the intrinsic matrix A and distortion coefficients is accomplished using the OpenCV camera calibration toolbox. A checker board pattern was used to carry out calibration. The pattern was moved to 30 different positions within the endoscope camera view. Figure 3.28 shows an example of the calibration checkboard within the endoscope view.

The extrinsic parameters of a stereo endoscope describes the rigid motion trans-

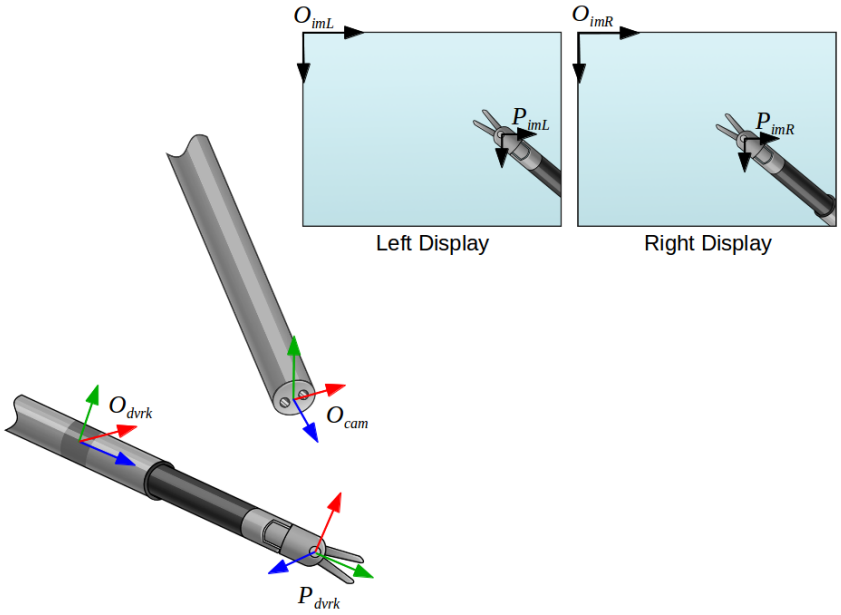


Figure 3.29: Hand-eye calibration system diagram.

formation consisting of a rotation R and translation T between the left and right cameras. Conventionally, the transformation is defined such that a position in the left camera is transformed to the right camera frame with $X_{right} = R \cdot X_{left} + T$.

3.6.3 Tool to Camera Coordinate Frame Calibration

The transformation T_{dvrk}^{cam} between the DVRK tool frame and endoscope camera frame needs to be obtained with a second calibration process. A diagram of the hand-eye calibration components is shown in Figure 3.29. The tool frame is defined by the DVRK , with the center of rotation as the coordinate system origin O_{dVRK}. The camera coordinate system origin is defined as O_{cam}.

A surgical tool is moved to several different locations within the endoscope view. At each location, the tool position in the DVRK frame $P_{dvrk} = (x_{tool}, y_{tool}, z_{tool})$ and images of the endoscope scene are recorded. Using the recorded images, the

pixel location of the tool tips for the left and right imaging channels, $P_{imL} = (u_L, v_L)$ and $P_{imR} = (u_R, v_R)$, are manually determined. Then, the tool position in the camera frame can be resolved by triangulating the pixel locations of the tool tip to obtain 3D coordinates $P_{cam} = (x_{cam}, y_{cam}, z_{cam})$. The specific point on the tool which is used is the last joint on the distal end. The tool position in the camera frame can then be calibrated to the DVRK frame using a least squares fitting to obtain T_{dvrk}^{cam}.

$$P_{cam} = T_{dvrk}^{cam} \cdot P_{dvrk} \tag{3.33}$$

3.6.4 Two-Step 3D Point of Gaze Calibration

To achieve a 3D point of gaze calibration, one method is to first calibrate the surgeon's eye gaze in 2D. If the 2D points of gaze on the left and right screens of the stereoscopic display are known, these gaze positions can be triangulated to calculate the 3D gaze position. However, with any error in either the left and right 2D points of gaze, the reconstructed 3D position will be incorrect. Thus, a second step is added to more accurately calibrate the depth of gaze to the 3D surgical scene. For this second step, instead of directly triangulating the left and right 2D gaze positions, we calculate the horizontal gaze disparity and then calculate the 3D gaze position by finding a mathematical function which relates gaze disparity to 3D depth.

First, the surgeon is asked to complete the 2D calibration described in the previous section. After a 2D calibration, the surgeon's eye gaze positions for the left and right eye are $p_R = (p_{xR}, p_{yR})$ and $p_L = (p_{xL}, p_{yL})$ respectively. We can calculate disparity as:

$$Disparity = dx = p_{xR} - p_{xR} \tag{3.34}$$

Then, to relate their 2D gaze position to 3D positions, the surgeon completes a 3D calibration by moving the da Vinci tool to several different locations $P_{tool} = (X_{tool}, Y_{tool}, Z_{tool})$ across the view and at varying depths. The surgeon is asked to look at the tool when they have positioned it and it is stationary. During this time, the surgeon's pupil-glint vector data and tool tip position data are recorded. After three seconds, the gaze tracker stops recording gaze positions and the surgeon is

asked to move to the next target.

The gaze disparity dx is then fit to the position of the tool in the camera frame Z direction. This is illustrated in Figure 3.30. In the fitting polynomial function we additionally include the mean horizontal and vertical positions of the POG, p_{xmean} and p_{ymean}:

$$p_{xmean} = \frac{p_{xR} + p_{xL}}{2}, \tag{3.35}$$

$$p_{ymean} = \frac{p_{yR} + p_{yL}}{2}. \tag{3.36}$$

With coefficients $c_0 ... c_6$, the polynomial is defined as:

$$Z_{tool} = c_0 dx^2 + c_1 dx + c_2 p_{ymean}^2 + c_3 p_{ymean} + c_4 \tag{3.37}$$

This is expressed using matrix notation as:

$$Z_{tool} = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & c_4 \end{bmatrix} \begin{bmatrix} dx^2 \\ dx \\ p_{ymean}^2 \\ p_{ymean} \\ 1 \end{bmatrix} \tag{3.38}$$

The coefficients $c_0 ... c_6$ are found through a least-squares estimation.

3.6.5 One-Step 3D Point of Gaze Calibration

For single-step 3D eye gaze calibration, the surgeon is not required to first perform a 2D calibration to determine their point of gaze on the display screen. Instead, during the 3D calibration step, the positions of the tool tip are back-projected onto the image plane pixels.

During calibration, the surgeon only needs to move the da Vinci tool to several different locations across the view and at varying depths. The surgeon then looks at the position of the tool tip when it is stationary. During this time, the surgeon's pupil-glint vector data and tool tip position data are recorded. After three seconds, the gaze tracker stops recording gaze positions and the surgeon is asked to move to

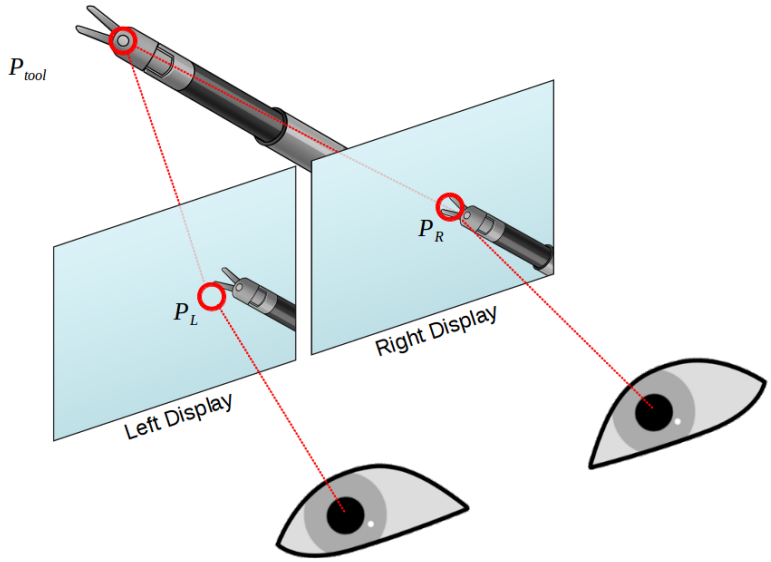


Figure 3.30: Two-step calibration diagram.

the next target.

To calculate the 2D gaze point, the recorded tool tip positions are transformed from the DVRK reference frame to the camera reference frame. The tool tip position in the camera frame P_{tool} can then be projected onto the image planes of the left and right cameras, and transformed to pixel coordinates $p_{toolpixL}$ and $p_{toolpixR}$ using the method described in Section 3.6.2 (see Figure 3.31). These pixel positions can then be related to PG vectors for the left and right eyes using the same second order polynomial fit described in Section 3.5. Least squares estimation is carried out to find the coefficient matrix B for the left and right eyes. After the 2D gaze is calibrated, Equation 3.38 is applied to calibrate for depth.

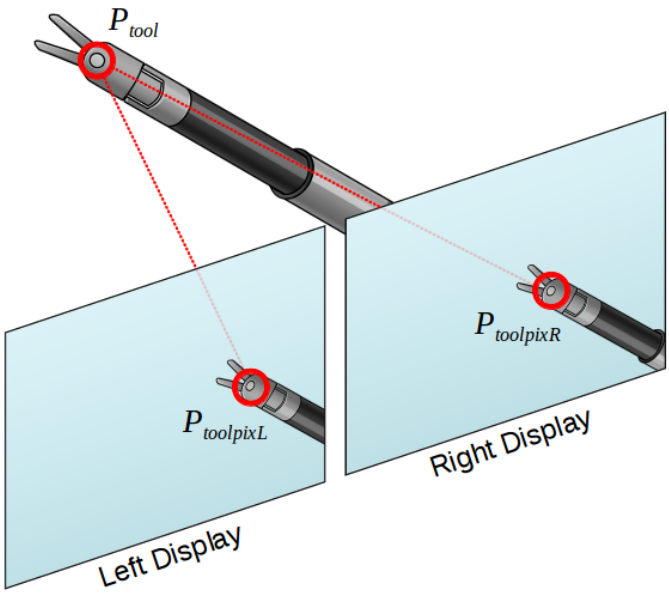


Figure 3.31: Tool projection onto stereo display screens.

3.6.6 3D POG Estimation

After calibration, the 3D POG $P = (X,Y,Z)$ can be determined with the pupil-glint vector of the left and right eyes. The pupil-glint vectors are first scaled to head depth position using Equation 3.27. The right 2D POG $pr_{pix} = (p_{xr}, p_{yr})$ and left 2D POG $pl_{pix} = (p_{xl}, p_{yl})$ is computed using Equation 3.24 and Equation 3.25.

The disparity dx is calculated using Equation 3.34 and a moving average filter is applied to the disparity values. The mean POG position from both eyes (p_{xmean}, p_{ymean}) is calculated using Equation 3.35 and Equation 3.36. Using the previously computed coefficients $c_0 ... c_6$, the depth Z is calculated using::

$$Z = c_0 dx^2 + c_1 dx + c_2 p_{xmean}^2 + c_3 p_{ymean}^2 + c_4 p_{xmean} + c_5 p_{ymean} + c_6 \qquad (3.39)$$

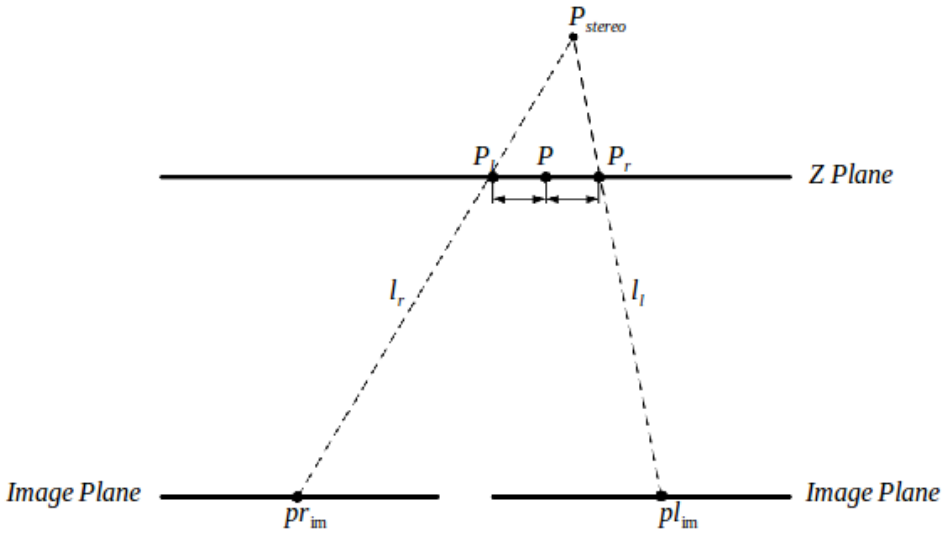


Figure 3.32: Diagram of 3D POG estimation.

A limit is set for Z to reduce error in further calculations. If the Z position is above 0mm (in front of camera) then it is set to a depth of -10mm, and if the Z position is farther than -150mm it is set to a depth of -150 mm.

Using the fitted gaze Z position, the eye gaze P position can be approximated. We calculate this approximation by intersecting two lines l_l and l_r stemming from the 2D POG of the left and right eyes, to a plane parallel to the camera's XY-plane and set at a depth of Z (labelled "Z Plane" in Figure 3.32). Each line can be described using vector notation as:

$$x = p_1 + t(p_2 - p_1), \tag{3.40}$$

where $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ are points on the line.

The vector notation for a plane is:

$$(x - p_0) \cdot n = 0, \tag{3.41}$$

where $p_0 = (x_0, y_0, z_0)$ is a point on the plane, and $n = (n_x, n_y, n_z)$ is a normal vector to the plane.

The point of intersection, can be calculated by solving Equation 3.40 and Equation 3.41 for t:

$$t = \frac{(p_0 - p_1) \cdot n}{(p_2 - p_1) \cdot n}. \tag{3.42}$$

Thus, setting $p_0 = (0,0,Z)$, $p_1 = pl_{im}$ and pr_{im}, and $p_2 = P_{stereo}$, P_l and P_r can be calculated as:

$$P_l = pl_{im} + \Big(\frac{(p_0 - pl_{im}) \cdot n}{(P_{stereo} - pl_{im}) \cdot n}\Big)(P_{stereo} - pl_{im}) \tag{3.43}$$

$$P_r = pr_{im} + \Big(\frac{(p_0 - pr_{im}) \cdot n}{(P_{stereo} - pr_{im}) \cdot n}\Big)(P_{stereo} - pr_{im}) \tag{3.44}$$

Finally, the 3D gaze position is approximated as $P = (P_l + P_r)/2$. This is an approximation of the 3D POG position.

In this method, the effects of 2D calibration errors on the 3D POG are reduced by mapping the eye gaze disparity to depth position instead of directly performing stereo triangulation. We use this corrected depth position to calculate an approximate X and Y location of the POG.

Chapter 4

Gaze Tracker Accuracy

We evaluated the accuracy of the eye gaze tracker for both 2D and 3D eye gaze tracking. A study was performed with 11 subjects at the Robotics and Control Lab in the University of British Columbia. Participants were contacted through an invitation email to the robotics@ece.ubc.ca mailing list. As we are mainly interested in measuring the accuracy of the eye gaze tracker, participants were qualified to participate regardless of whether they had experience with the da Vinci surgical robot or with eye gaze tracking devices. Out of the 11 participants, 10 took part in the 3D accuracy evaluation, while all 11 participants took part in the 2D accuracy evaluation.

During the study we recorded the following data from each participant:

- Timestamp
- Eye gaze 3D position (X,Y,Z)
- Eye gaze 2D position (X,Y for both eyes)
- da Vinci 3D tool position (X,Y,Z for both tools)
- Eye pupil information (diameter, position for both eyes)
- Infrared light reflection on cornea (position for both eyes)
- Displacement vector between pupil and light reflections for both eyes

4.1 Study Environment Setup

The study was held in the Robotics and Control Laboratory in room 3090 of the University of British Columbia Fred Kaiser building. Within the laboratory there is a full da Vinci surgical robot in addition to the da Vinci Research kit. The da Vinci research kit was used to retrieve the position of surgical tools and to move the robot or set haptic forces.

The eye gaze tracking device was placed directly onto the da Vinci surgeon console. Two computers are involved with the study setup, one computer (PC 1) for handling eye gaze calculations and data logging, and another computer (PC 2) for handling communication with the da Vinci Research Kit (see Figure 3.1).

Participants were seated at the da Vinci surgeon console, shown in Figure 4.1. The surgeon console has a stereoscopic display consisting of two separate Barco MCD214 CRT monitors. The height of the surgeon console was adjusted to the height of each participant for a comfortable fit. During each study, the participants placed their foreheads on a foam headrest and look into the surgeon console at the display screens.

The da Vinci patient-side slave was set up with two robotic tools, one on either side on the endoscope camera. The individual tools are shown in Figure 4.2. The arm on the right was equipped with a large needle driver instrument which participants controlled using the right-hand master manipulator. This instrument will be henceforth referred to as PSM 1. The arm on the left of the endoscope was equipped with a pro-grasp instrument, this instrument will be referred to as PSM 2.

4.2 2D Gaze Tracking Accuracy Test Protocol

Participants were first shown a sample video of the calibration sequence, and were instructed that during calibration they would be required to gaze at the calibration target. Then, participants performed eye gaze calibration using 5-point, 9-point, and 16-point calibration (shown in Figure 4.3, Figure 4.4, and Figure 4.5). Following each calibration, the participants were instructed to look at nine test positions on the screen (Figure 4.6), and to gaze at each position for 3 seconds, during which time the researcher logged their eye gaze position.

Figure 4.1: Surgeon console with retro-fit eye gaze tracker

(a) **(b)**

Figure 4.2: Intuitive Surgical da Vinci instruments (a) large needle driver and (b) pro-grasp forceps.

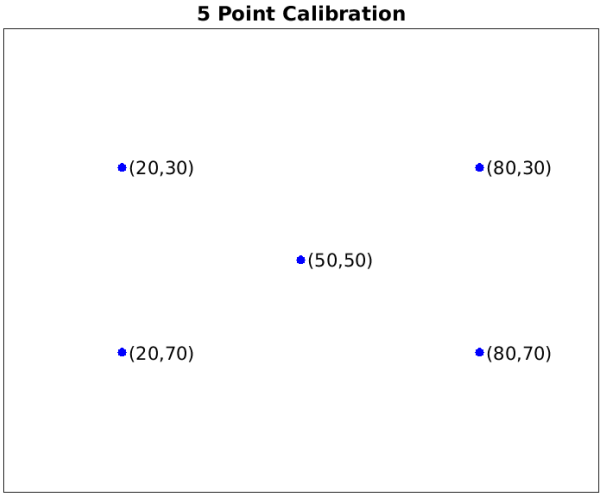


Figure 4.3: 5 point calibration target locations.

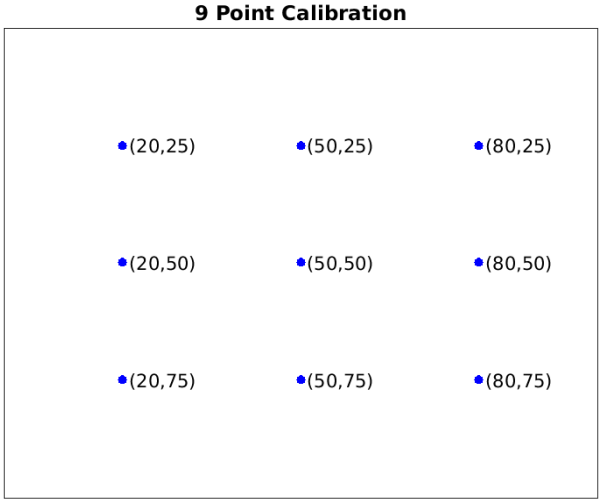


Figure 4.4: 9 point calibration target locations.

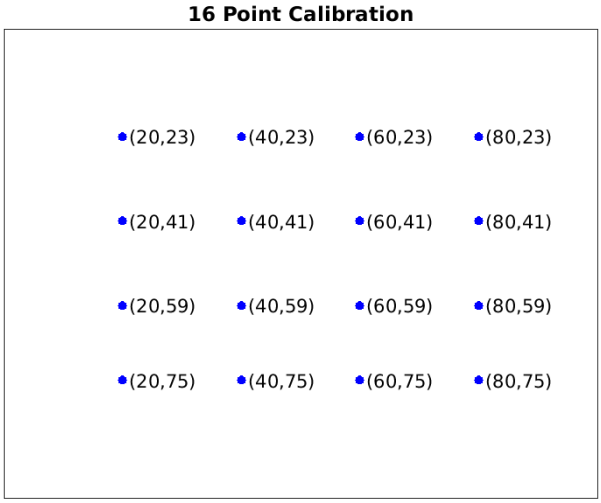


Figure 4.5: 16 point calibration target locations.

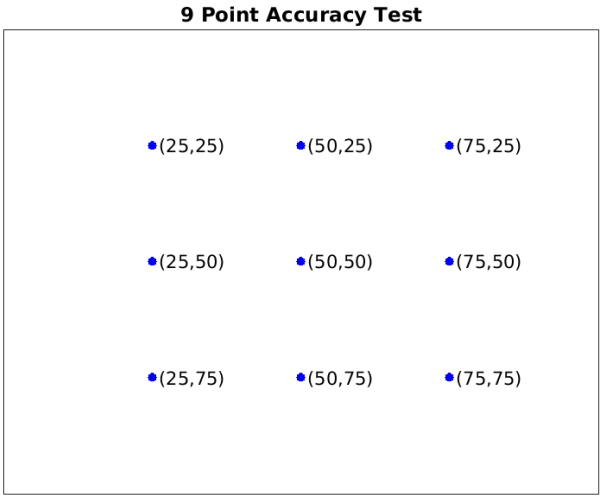


Figure 4.6: 9 point accuracy test target locations.

4.3 3D Gaze Tracking Accuracy Test Protocol

We performed three tests to evaluate the 3D gaze tracking accuracy- a depth perception test, a manual 3D calibration test, and an automated 3D calibration test. Firstly, a depth perception test was carried out in order to gauge each participants sense of depth within the da Vinci stereo viewer. We based this test on the Howard-Dolman test [23], which is used to evaluate depth resolution in real space. For the Howard-Dolman test, two vertical poles are shown, and one is adjusted until it is just nearer than the other stationary pole. Alternatively, in our setup we used the da Vinci surgical tools to measure depth acuity. The left hand tool, PSM 2, equipped with a pro-grasp forcep instrument held a small 3D printed board that has a repeatable grasp design. Each participant was instructed to move the right tool, PSM 1, until it was at the same depth as the center of the end of the board. This is shown in Figure 4.7.

For the manual 3D calibration test, a test board containing 9 labelled positions was placed in view of the endoscope, as shown in Figure 4.8. The labels were placed in three rows and three columns, at varying heights. Each participant was required to control PSM 1, and move the tool end effector to the labelled positions shown. Each participant was first shown a video of this procedure to explain the process, and then was given time to practice controlling the robot. To start the calibration, a 2D calibration measurement was taken using 9 calibration points. Then, the participants were instructed to start moving the tool to the labelled points. At each position, the participants were asked to hold the tool steady, and to gaze at the jaw joint. After two seconds, the researcher conducting the study would prompt the participant to move to the next labelled position. Once the 9 points had been finished, a separate test board containing another 9 labelled positions was shown to the participants, and they were required to repeat the process of moving the tool tip to each position and gazing at the end-effector. The first test board was used for calibration, and the second test board was using for testing the calibration.

The automated 3D calibration test was carried out in the same manner as the manual test, except the participants did not have to control the tool themselves, and instead PSM 1 was controlled to follow a pre-set trajectory using the ROS interface. This was carried out because while controlling the robot, participants may exhibit

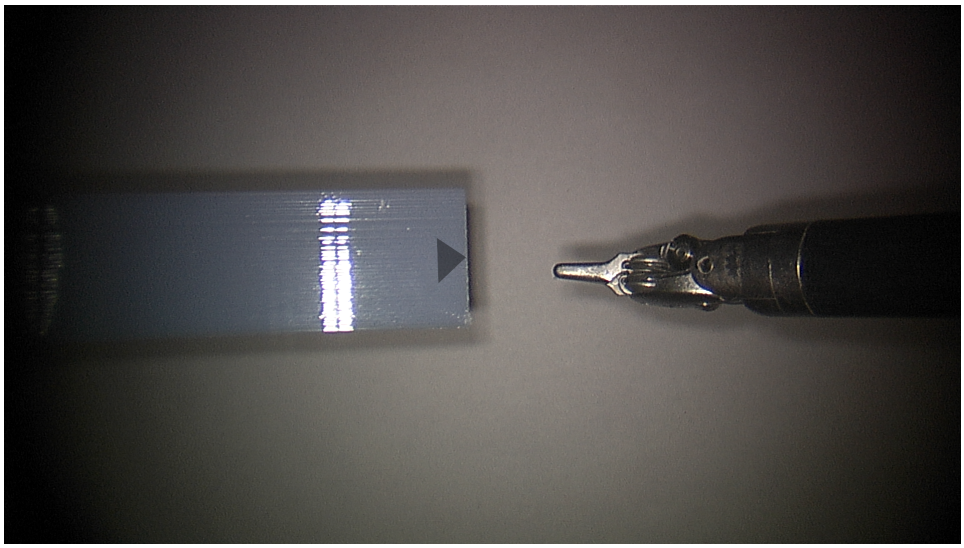


Figure 4.7: Depth test view within da Vinci surgeon console.

head movement. We wanted to see whether this head movement significantly affects calibration results.

4.4 2D Gaze Tracking Accuracy Results

The accuracy of the 2D eye gaze tracker for 11 users was calculated for the three calibration modes- 5 point, 9 point, and 16 point calibration. The eyes of one user was not tracked consistently, and thus their dataset has been removed from the final calculations. The 2D eye gaze tracking accuracy was calculated across the remaining ten users. Gaze positions which were calculated to be outside of the viewing screen are considered invalid points, and were removed from the data set.

In eye gaze tracking literature, the accuracy of an eye gaze tracker is reported as degrees of visual angle. The error is the angle between the line from the user's eye to the estimated point of gaze, and the line to the true point of gaze. This is calculated by considering the physical dimensions of the eye gaze tracking setup. The da Vinci stereo viewer consists of two 14 inch Barco M2 cathode ray tube monitors with display dimensions of 11.2 inches wide and 8.4 inches high. The surgeon's eyes are approximately 18 inches from the display [16]. Thus, for a

Figure 4.8: 3D accuracy evaluation view within da Vinci surgeon console.

Table 4.1: 2D Eye Gaze Pixel Accuracy

Eye Side	5 Point Calibration	9 Point Calibration	16 Point Calibration
Right	49.32 $\pm$ 16.37 pixels	32.73 $\pm$ 11.90 pixels	31.19 $\pm$ 12.31 pixels
Left	47.93 $\pm$ 15.91 pixels	26.61 $\pm$ 9.98 pixels	30.37 $\pm$ 11.45 pixels

vertical error of ΔX inches and a horizontal error of ΔY inches the degree of visual angle can be calculated as:

$$\theta = 2tan^{-1}\left(\frac{\frac{\sqrt{\Delta X^2+\Delta Y^2}}{2}}{18\ inches}\right) \tag{4.1}$$

The 2D accuracy results are summarized in Table 4.1 as pixels and Table 4.2 as degrees of visual angle.

Table 4.2: 2D Eye Gaze Degree Accuracy

Eye Side	5 Point Calibration	9 Point Calibration	16 Point Calibration
Right	2.96 $\pm$ 0.90 $^\circ$	2.03 $\pm$ 0.59 $^\circ$	2.31 $\pm$ 1.43 $^\circ$
Left	2.91 $\pm$ 0.80 $^\circ$	1.73 $\pm$ 0.68 $^\circ$	2.14 $\pm$ 1.08 $^\circ$

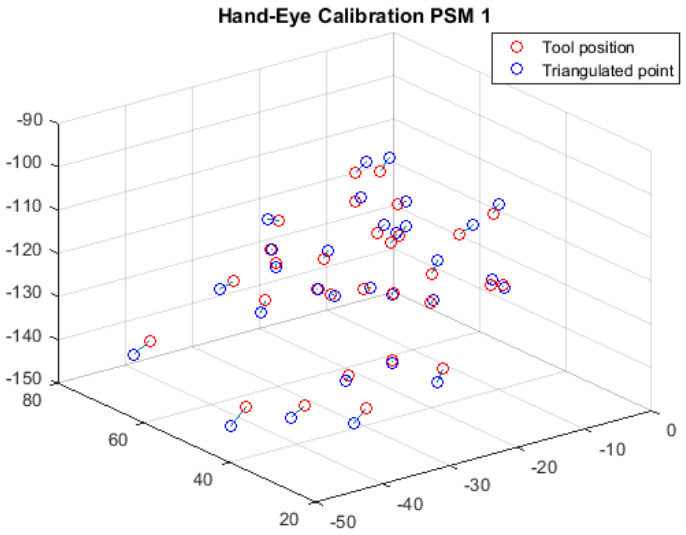


Figure 4.9: Hand-eye calibration results for PSM 1, tool position shown in red and estimated tool position shown in blue.

4.5 Hand-Eye Calibration Results

A hand-eye calibration was carried out as defined in 3.6.3 for both PSM 1 and PSM 2. In total, 30 positions of each tool was recorded. The BlackMagic Design Decklink Quad 2 capture card was used for capturing frames from the da Vinci endoscope. The accuracy of the calibration was 1.85 mm for PSM 1 and 2.05 mm for PSM 2. Plots showing the tool position and estimated tool position from pixel coordinates are shown in Figures 4.9 and 4.10.

4.6 Depth Perception Test Results

The positions of PSM1 and PSM2 at three depths were recorded. The depths tested were 90 mm, 75 mm, and 60 mm. A Python script was written to set the position and orientation of PSM 2 to the pre-set positions.

With the position P_1 and orientation R_1 of the end-most joint on PSM 1, the tool tip $P_{1,Tip}$ is calculated as:

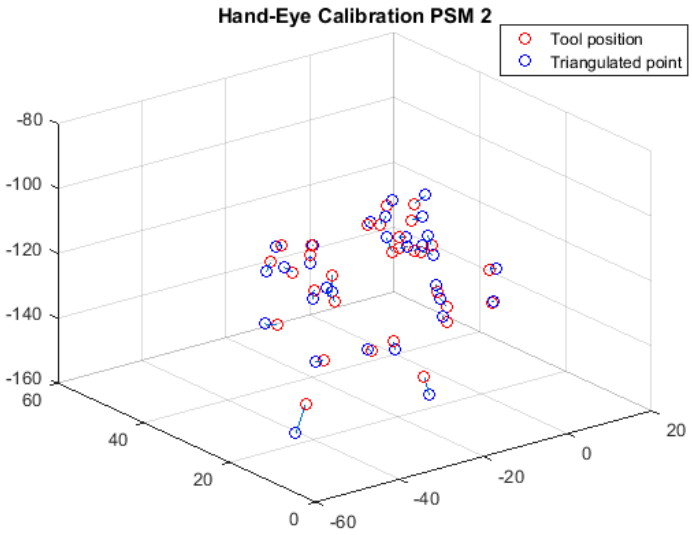


Figure 4.10: Hand-eye calibration results for PSM 2, tool position shown in red and estimated tool position shown in blue

$$P_{1,Tip} = P_1 + R_1 \begin{bmatrix} 0 \\ 0 \\ 10 \end{bmatrix} mm. \tag{4.2}$$

We calculate the end of the depth test plate similarly. With the position P_2 and orientation R_2 of the end-most joint on PSM 2, and the dimensions of the depth test plate measured through Solidworks, the end of the plate P_{plate} is found as:

$$P_{plate} = P_2 + R_2 \begin{bmatrix} -50 \\ 0 \\ 22 \end{bmatrix} mm. \tag{4.3}$$

The error can then be calculated as $error = ||P_{1,Tip} - P_{plate}||$. For all ten participants, the resulting depth error during this test was 6.20 ± 3.60 mm mm.

4.7 3D Gaze Tracking Accuracy Results

Of the ten participants for the 3D gaze tracking evaluation, one of the participants was not considered in calculations because of data logging errors that were fixed for subsequent participants. The 3D accuracy of the eye gaze tracker was 17.18 +/- 8.13 mm for a one step calibration, and 13.33 +/- 3.23 mm for a two step calibration considering both manual and automatic robot control tests. The results are shown in Table 4.3.

Table 4.3: 3D Eye Gaze Accuracy

	One-Step	Two-Step	Combined
Manual	18.44 $\pm$ 8.15 mm	13.67 $\pm$ 3.77 mm	16.06 $\pm$ 6.63 mm
Auto	15.93 $\pm$ 8.40 mm	12.98 $\pm$ 2.78 mm	14.45 $\pm$ 6.26 mm
Combined	17.18 $\pm$ 8.13 mm	13.33 $\pm$ 3.23 mm	15.26 $\pm$ 6.41 mm

The effect of manual or automated robot control on eye gaze tracking was also evaluated during both one-step and two-step calibration. A paired t-test was carried out to test the null hypothesis that the error from controlling the robot automatically is equal to the error from controlling the robot manually, with the alternate hypothesis being that manually controlling the robot has a significantly larger error than using an automated control. A significance level (α)of 0.05 was used.

For one-step calibration, there was no significant difference between manual control (18.44 $\pm$ 8.15 mm) and automated control (15.93 $\pm$ 8.40 mm); t(8)=0.6004, p=0.2824. Similarly, for two-step calibration, there was also no significant difference between manual control (13.67 $\pm$ 3.77 mm) and automated control(12.98 $\pm$ 2.78 mm); t(8)=0.7616, p=0.2341. We can conclude that allowing participants to manually control the robot does not significantly affect the calibration results. In future iterations of the eye gaze tracker, we will not use an automated position control, but will continue to allow surgeons to freely and naturally control the robot. A calibration scheme can be designed such that the surgeon will be able to continuously move the robot while gazing at the tool tip. This would be less tedious than stopping the tool tip at several positions to calibrate.

Lastly, we compared whether there is any significant difference between the two calibration methods. Including both manual and automated test methods, a

paired t-test was carried out to test the null hypothesis that the there is no significant difference between one-step and two-step calibration methods, with the alternative hypothesis that the one-step calibration has a significantly higher error. Again, a significance level of 0.05 was used. The result of the test is that a one-step calibration (17.18 $\pm$ 8.13 mm) is significantly higher than the two-step calibration (13.33 $\pm$ 3.23 mm); t(17)=2.2116, p=0.0205.

The difference between one-step calibration and two-step calibration is that the 2D calibration is carried out by projecting the tool position to corresponding pixel positions in the display screens. This process includes the calibration errors from hand-eye calibration and endoscope calibration, thus 2D calibration is less accurate than directly displaying 2D targets. However, a one-step calibration method holds value in that it is more convenient than a two-step calibration, so future work will be carried out to improve system calibration and improve the results for a one-step calibration method.

4.8 Discussion

The known measured factors contributing to the 3D tracking error are the error from the initial hand-eye calibration (1.85 mm) and the depth perception limitations for each user (5.77 $\pm$ 3.74 mm). Taking these two factors into account, we believe that the accuracy of our gaze tracker is reasonable. Other sources of error which are included in the final results are lost or incorrectly detected pupil and glint features, head movement by the participants, and instances when the participants were not looking directly at the specified target during data collection. Furthermore, with a small displacement of 5 mm between the left and right imaging channels of the stereo endoscope, depth perception even with perfect POG estimation can be expected to have some error.

4.9 Conclusion

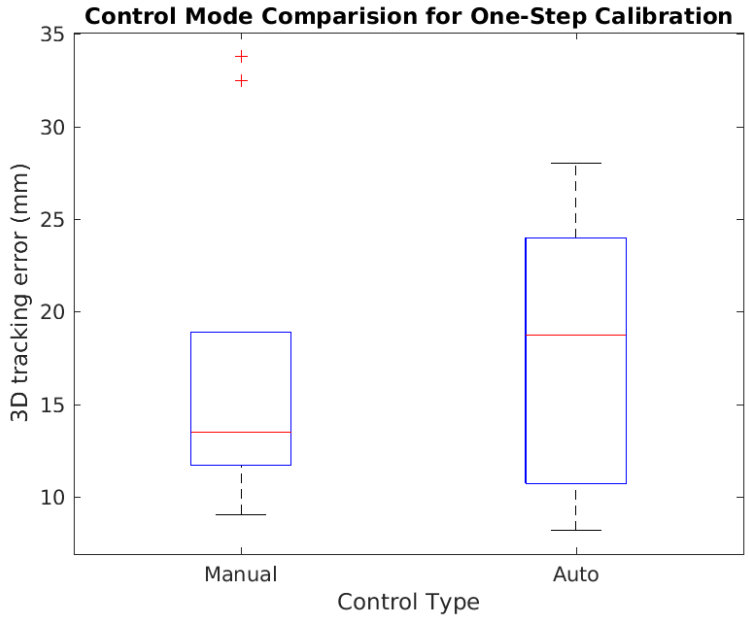
We have developed a retro-fit eye gaze tracker for the da Vinci surgeon console. The eye gaze tracker consists of two cameras and six IR LEDS which capture imagery of the surgeon's eyes while operating the da Vinci surgical robot. Software was developed which consists of a user interface for researchers to conduct eye

gaze tracking studies and collect eye gaze data, as well as eye gaze tracking algorithms for monitoring the surgeon's eye gaze in both 2D and 3D.

The eye gaze tracker uses the pupil-glint method to track eye gaze, where the glint is used as a stationary reference point while the pupil changes positions. 2D eye gaze tracking is carried out by mapping the pupil-glint vector to on-screen coordinates through a calibration procedure. 3D eye gaze tracking is carried out by instructing the user to move a surgical tool to several positions, and to look at the tool during each position.

Future work for the eye gaze tracker is an improved glint and pupil feature detection for more users (two data sets in the accuracy evaluation were not used because of feature detection failure). Improved feature detection will result in better calibrations in both the 2D and 3D calibration methods. Furthermore, the end goal for 3D eye gaze tracking is a calibration method in which the surgeon will move their surgical tool in a continuous motion while gazing at the tool tip. Our current 3D eye gaze tracking framework calibrates the surgeon's eye in 3D when they look at the surgical tool end-effector at different positions, which may be time consuming and tedious. This can be extended to a continuous motion by matching the pupil-glint vector motion with tool motion.

Lastly, this eye gaze tracker will be provided to the da Vinci Research Kit community as a tool for researchers to develop eye gaze tracking applications and observe surgeon eye gaze behaviour. An eyepiece frame for other models of the da Vinci surgical systems will be designed. This eye gaze tracker will help researchers design more user-centric surgical interfaces and improve the surgeon experience during robotic surgery.



(a)

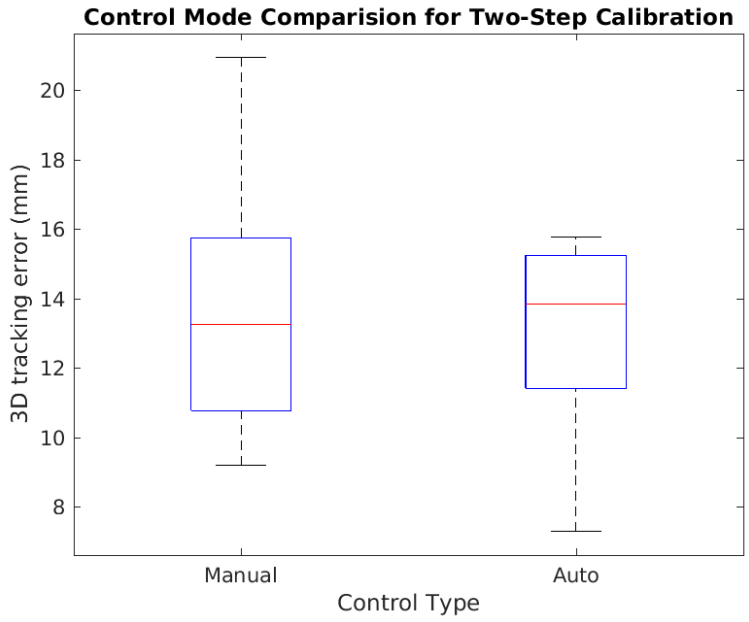


Figure 4.11: Box plots showing comparison of test modes for (a) one-step calibration and (b) two-step calibration.

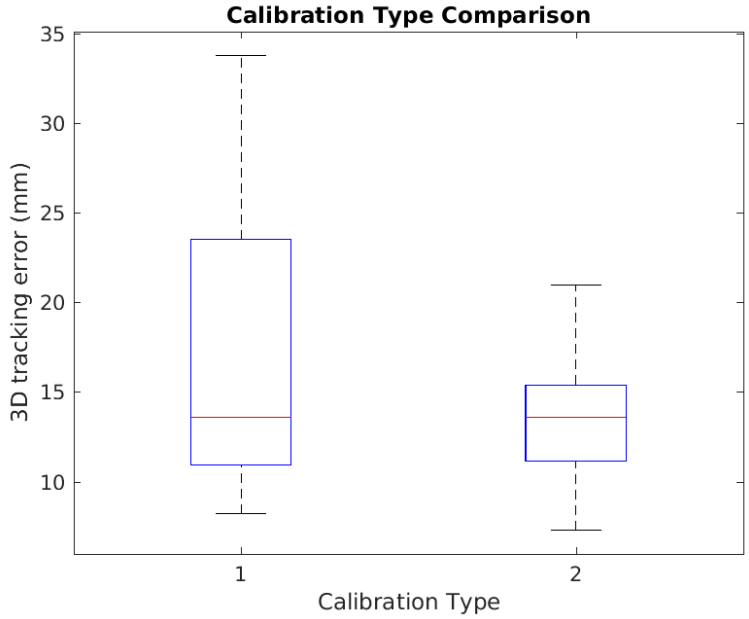


Figure 4.12: Box plot showing comparison of calibration methods.

Chapter 5

Gaze Motor Control

5.1 Introduction

Eye gaze is not only a passive way of perceiving the world, but it is also closely linked to action processes with a bidirectional relationship [65]. It is known that the locations of gaze fixations are tied to interaction goals with objects of interest. Eye gaze is directed towards regions which are relevant to decision making or for planning motor behaviour [6, 19]. This implies that during surgery, a surgeon's intentions can be derived from analysis of their gaze patterns.

A study by Atkins et al. on the use of eye gaze tracking technology during laparoscopic minimally invasive surgical procedures found that a surgeons eye gaze precedes motor movement with a consistent delay [4]. It was also noted that gaze fixations occurred earlier with tasks requiring higher precision. Using this observation regarding the strong link between gaze and action, gaze can be employed as an active control input to a surgical robot. Gaze information allows us to determine where a surgeon is planning on placing surgical tools, and robotic assistance can then be given to help the surgeon in manipulating the tools.

In this work, we demonstrate gaze-contingent motor control for the first time in a complete physical master-slave system, with a standard skill assessment task. Previous related work has demonstrated gaze-contingent motor control on simulated slave robots and simulated tasks. In our implementation, we use a full da Vinci standard surgical robot along with the DVRK controllers. The DVRK is a re-

search platform with first-generation da Vinci components provided by Intuitive Surgical. Open source software developed by John Hopkins University is used to control and interface with the DVRK [28].

To achieve this novel demonstration of both gaze control, and the capabilities of the DVRK, we integrated a new eye gaze tracker design that can be easily retrofitted to the da Vinci surgeon console. In Section III, we describe the design of the tracker, the signal processing it employs, and the calibration required in order for it to track eye-gaze in the 3D coordinate space of the da Vinci. The approach includes a novel calibration sequence that uses the da Vinci tool coordinates recorded at multiple points in the work-space to find the relationship between gaze disparity and depth. In Section IV we describe the control architecture used to achieve shared gaze-master control of the slave robot. We provide the robot operator with a haptic force from the master manipulators which pulls them to move the slave robot tool towards their gaze point. A similar force feedback teleoperation control framework implemented with the DVRK is described in [39].

5.2 Control Architecture

The gaze control framework is shared control between the human operators hands and gaze. For our system, the slave robot is directly teleoperated by the master manipulators and a spring-damper coupling is placed in between the operators gaze and the master manipulators. In [46], it was observed that a spring force is the optimal force profile for gaze-contingent motor channelling. We also incorporated a damping component as using only a spring force resulted in oscillations. The force F applied to the master manipulators is defined as:

$$F = K(X_{gaze} - X_{tool}) + B(\dot{X}_{tool}) \tag{5.1}$$

where K is the spring constant, B is the damping coefficient, X_{gaze} is gaze position, X_{tool} is tool position, and $\dot{X}_{tool}$ is tool velocity. X_{gaze}, X_{tool}, and $\dot{X}_{tool}$ are relative to the master manipulator coordinate frame.

The K and B values were adjusted to provide a good response across all users. The final values selected were $K = 55N/m$ and $B = 50Ns/m$. The da Vinci system foot pedals are incorporated into the system as control inputs for the operator to

switch between following teleoperation and gaze control modes. The foot pedals are utilized as they are easily accessed by the operator and have support through the DVRK framework. Since the camera is fixed to maintain the tool to camera frame transformation, the camera focus foot pedal is used instead to enable or disable gaze control. This functionality is required as there may be cases in which it is not desirable for the operator to be pulled towards where they are looking. For example, a constantly applied gaze force will not allow the operator to glance at areas on the display that do not pertain to the motor task at hand.

The bimanual teleoperation control structure is shown in Figure 5.1. The master and slave dynamics are represented as lumped Linear Time Invariant (LTI) impedances Z_{hL}, Z_{hR}, Z_{MTML}, Z_{MTMR}, Z_{PSM1}, Z_{PSM2} each representing the seven degrees of freedom of the manipulators. Torques , τ_{hR}, τ_{MTML}, τ_{MTMR}, τ_{PSM1}, τ_{PSM2} are the torques at the left hand, right hand, left master manipulator, right master manipulator, patient side manipulator 1, and patient side manipulator 2 respectively. Torques τ_g, τ_h^*, τ_{GC} are the gaze input torque, hand input torque, and gravity compensation torque respectively. The PID controllers C_{PSM1} and C_{PSM2} manage joint level position control of each PSM and the controller. Gravity compensation is handled for both master manipulators by the gravity compensation controller C_{GC}. The controller C_G uses the gaze point P_G, position and velocity of the slave robots and calculates the spring-damper force to be applied to each of the master manipulators. A motion scale of 1:5 was applied between master and slave movements.

The gaze tracking software for this study uses a sampling frequency of 30Hz which is limited by the capture rate of the cameras. The teleoperation and joint-level control loops run much faster. Control loop frequencies were set at 1 kHz for the input/output communication level, 333 Hz for the kinematics and robot logic control, and 200 Hz for the teleoperation loop.

5.3 Experimental Setup

The gaze tracking and DVRK control software were run on two separate computers, both running the Ubuntu operating system. Two different computers were used in order to avoid decreasing the real-time performance of the robot control software

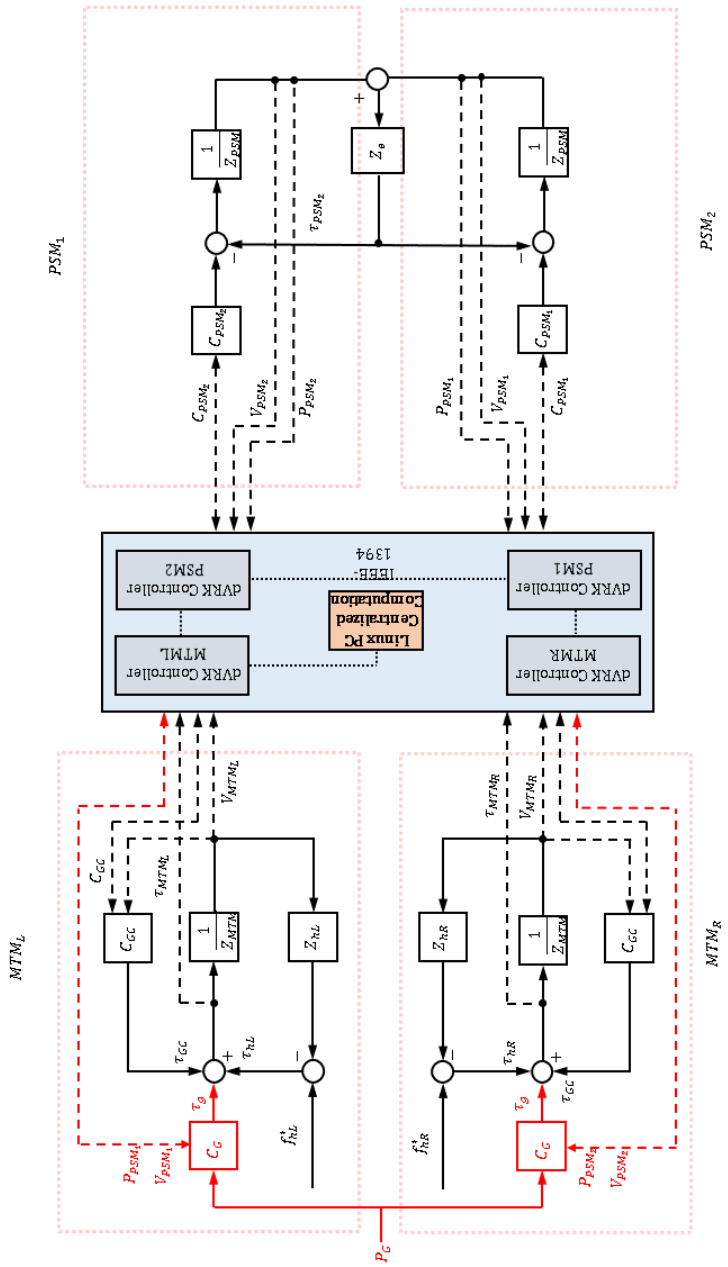


Figure 5.1: Control diagram for teleoperation with gaze control.

Figure 5.2: Experimental peg placement task.

due to CPU processor limitations. Communication between the two computers was made through TCP/IP sockets. As the frequency of the DVRK control loop is much higher than the gaze tracking frequency, messages received by the gaze client side are flushed at the end of each software loop to prevent the accumulation of old messages.

5.3.1 User Study

Five subjects were selected for the study, two have had over a year of experience with the da Vinci robot, two have had less than a year of experience, and one was completely new to the system. Four of the users wore glasses, and one user wore contact lenses. All five users are from an engineering background. The task for each subject was to place rings on pegs, which is a common laparoscopic training procedure. The peg board was custom made and 3D printed. Only one slave arm was used due to the simplicity of the pick and place task, however the control framework can be extended to bimanual tasks. The subjects repeated this task ten times with gaze and ten times without gaze. An image of the experimental setup is shown in Figure 5.2.

5.4 Results

5.4.1 Gaze Tracking Accuracy

The accuracy of gaze tracking in 3D was assessed for each subject. The subjects were instructed to gaze at all of the 12 pegs on the peg board for three seconds. A set of gaze points in 3D was recorded for each peg. The average gaze position at each peg was then calculated, and the error was defined as the difference between the distances between gaze positions and the true distances between pegs. Each subject repeated this test twice, and the average error from both tests was calculated.

The result of this accuracy test is shown in Table 5.1. The total average error is 10.18 mm with a standard deviation of 1.89 mm. This error incorporates the error from the camera to tool transformation previously described in Section V for the left tool in addition to the gaze estimation error.

5.4.2 Peg Placement Task

An extracted plot of the 3D tool trajectory and gaze positions is shown in Figure 5.3. The tool is shown moving towards the points of gaze where the user is fixating. The tool trajectory from a single user in the x, y, and z directions is shown in Figure 10. The movement of the tool clearly correlates with the points of gaze shown. The gaze points are intermittent as subjects were allowed to turn on and off gaze control depending on whether they were using fine or broad movements and we have removed the data points in which gaze was not enabled. Once gaze is enabled, the tool quickly moves towards the gaze position.

The percentage of time each user spent in gaze mode as well as the time spent completing the peg task with and without gaze is shown in Table 1. Gaze was enabled for less than 20% of the time for all users as picking up and dropping pegs took more time than moving from one end of the peg board to another.

User feedback on the system from subjects was positive as gaze reduced the effort needed to move from one side of the peg board to another. Subjects turned off gaze only for picking up and dropping pegs. While all subjects noted that they felt tasks were faster and easier with gaze control, no significant reduction in task

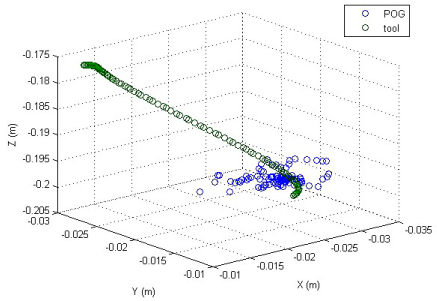


Figure 5.3: Tool trajectory (green) and gaze position (blue) in 3D space extracted from a peg placement task. The tool is shown moving towards the gaze position.

Table 5.1: Summary of motor control user study.

Subject	Gaze Tracking Error	Percentage of Time Spent Using Gaze	Average Time With Gaze	Average Time Without Gaze
1	11.93 mm	12.13%	1:10 min	0:55 min
2	11.62 mm	15.28%	1:08 min	0:55 min
3	7.31 mm	11.88%	1:26 min	2:13 min
4	9.32 mm	5.61%	1:24 min	1:13 min
5	10.72 mm	14.31%	2:17 min	2:34 min

completion time was noted. An increase in top speed occurred for two of the three novice users, but a decrease in speed occurred for the other three users.

The peg placement task is especially simple and did not present quantitatively conclusive results. In the future a more difficult task will be tested, such as suturing or target following, which will allow for analysis of user precision. Additionally, all users noted that the use of the camera foot pedal for gaze control in addition to the clutch foot pedal made the task more difficult as they needed to coordinate their foot movements. Lastly, another limitation of this system observed during user studies is that in order to move the tool across the peg board, the master manipulators would have to be moved to the edge of the workspace within the surgeon console. The motion scaling between the master manipulators and slave robot should be increased in future studies to account for this effect.

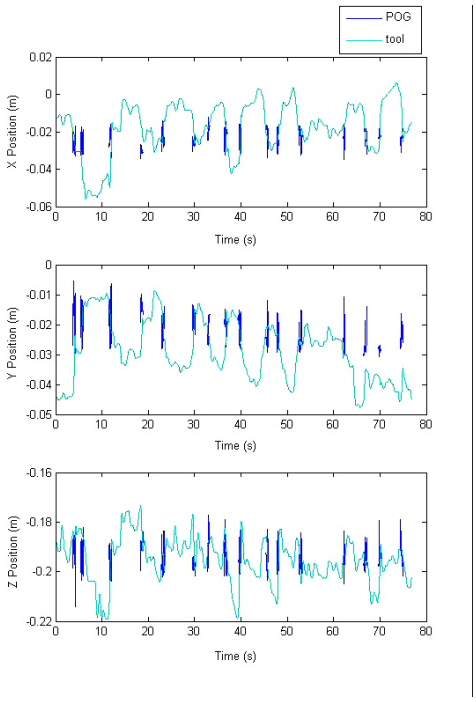


Figure 5.4: Tool trajectory (light blue) for X, Y, Z directions with gaze points (dark blue).

5.5 Conclusion

In the future we plan to improve the accuracy and calibration process for our eye gaze tracker. Currently, calibration takes approximately five minutes to complete, which may be inconvenient in a realistic surgery scenario. A possible improvement consists of expanding the 2D point calibration approach for the eye tracker to include depth, through projection of the 3D tool position onto 2D pixel positions. This would enable us to calibrate the system by having the operator gaze at the tool at several positions, without the need of an additional display sequence. This would be a more natural hand-eye calibration procedure. An additional improvement to the system would be to include a more convenient mechanism for enabling and disabling gaze control. Our current use of a foot pedal adds an additional layer of control for the user that could be replaced with simpler approaches based on specific gaze sequences or voice control.

To conclude, this chapter presents the integration of gaze as an active input for surgical robot control. While gaze has been integrated before with the da Vinci console, we are not aware of other studies in which the actual da Vinci slave robots were also controlled by gaze. Our proposed control architecture includes both gaze and master input, and should be usable in experiments with the da Vinci Research Kit interface for novel user interfaces. Future work will be carried out to quantify the benefits of this system with more complete and conclusive tasks.

Chapter 6

Camera Movement Prediction

In this chapter, eye gaze information is used for building a predictive model for camera control during robot-assisted surgery. Eye gaze and robot kinematic data was collected from surgeons in porcine and cadaver training laboratories. A random forest classifier is applied to predict camera movement.

6.1 Introduction

During Robot-Assisted Surgery (RAS), the surgeon has direct control over endoscope positioning without the need for a separate camera operator. For example, on the da Vinci Si® Surgical System, surgeons press and hold a foot pedal to move the camera with their hand controllers. The surgeon has to constantly switch between tool control and camera control modes, which disrupts the surgical workflow and increases mental workload. Camera control is also an important skill that surgeons need to learn, as proficient endoscope control results in effective visualization, which is needed to plan surgical approaches and interact efficiently and safely with tissue. Endoscope-related performance metrics are indicative of surgeon experience and linked to technical performance [27]. Experienced surgeons will adjust their viewpoints more frequently, spend less time moving the endoscope, and spend less time in-between endoscope movements. An improvement on current surgical robot systems would be to introduce autonomous camera control which allows surgeons to focus on completing surgical tasks while providing

an optimal viewpoint of the surgery. This would further eliminate the learning curve associated with camera control, and improve surgeon performance by allowing novice surgeons to focus on learning surgical task execution.

One limitation in current approaches for viewpoint control during robotic surgery is the clutch interface for enabling camera control. In manual endoscope control with the da Vinci surgical robot, the surgeon uses a foot pedal to trigger camera control. Existing work in viewpoint control either adjusts the camera position based on tool positions [32, 50, 74, 75] or uses manual selection mechanisms such as a foot pedal, eye gestures or eye gaze dwell time [12, 31, 35, 47]. Using tool position to dictate camera movement may not be optimal when the surgeon needs to operate outside the centre of the field of view. Furthermore, by implementing a manual interface the surgeon still needs to attend to viewpoint control, reducing the effectiveness of an automated camera control system.

Another limitation in current viewpoint control systems is the decision regarding where to move the camera. In [29], current approaches to autonomic camera navigation are described as reactive systems, proactive systems, or a hybrid model. Reactive systems use sensor information and move the camera in response to the surgeon, while proactive systems predict surgeon movement and intent and adjust the camera accordingly. Many systems maintain the surgical tools in the center of view. In [74], the authors propose an endoscope guidance system which predicts the movement of surgical tool end-effectors, and autonomously positions the endoscope to maintain the left and right tools centered in view. They show that by pro-actively moving the camera based on a prediction of tool positions, they can reduce the amount of camera movements by 29%. The disadvantage of using tool position to direct the camera is that it disregards the context of the surgical scene in which the surgeon is operating.

Continuous camera movement is sub-optimal as a continuously changing view may be distracting to the user and makes it difficult to observe the surgical scene. Furthermore, an advantage of the robotic surgery system is the hand-eye mapping between the endoscope view and master tool manipulators. By constantly adjusting the viewpoint, this mapping may either degrade or the master tool manipulators positions need to be continuously adjusted.

One particular measure that is directly related to visualization and can be used

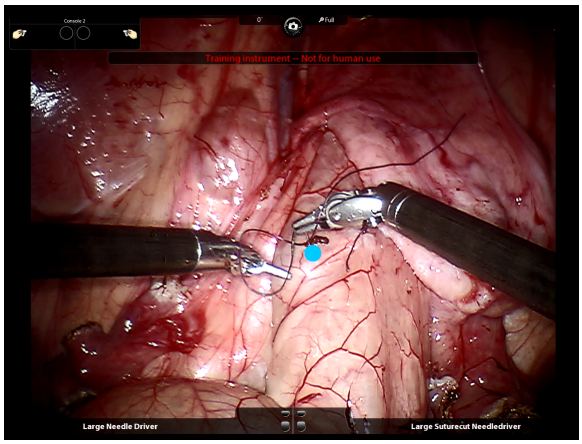
for automatic camera control is the eye gaze of the surgeon. Eye gaze has been used to study surgeon endoscope control policies, and is readily tracked as the surgeon looks at a stereoscopic display. The use of eye gaze for automatic viewpoint control has been studied both within robotic surgery and in other applications. In [44], Mylonas et al. use the depth of eye gaze to stabilize a camera view of a beating heart phantom. In [76], Zhu et al. control a camera to re-centre the viewpoint to the users point of gaze, with the application of tele-operated Receiver Operating Characteristic (ROC)k breaking robots. Eye gaze information has additionally been used to study and classify surgeon behaviour in minimally invasive surgery. Work by Sodergren et al. in manual endoscopy use a hidden Markov model for identifying eye gaze behavioural1 patterns related to effective endoscope re-orientation, evaluated within dry lab exercises [56]. Several groups use eye gaze data to automatically identify surgical tasks and segment the surgical work-flow [3, 26]. The direct relation between eye gaze and visualization as well as the ease at which eye gaze data can be obtained in a surgical robot system makes eye gaze information a good candidate for modelling surgeon behaviour with regards to endoscope control.

6.2 Materials and Methods

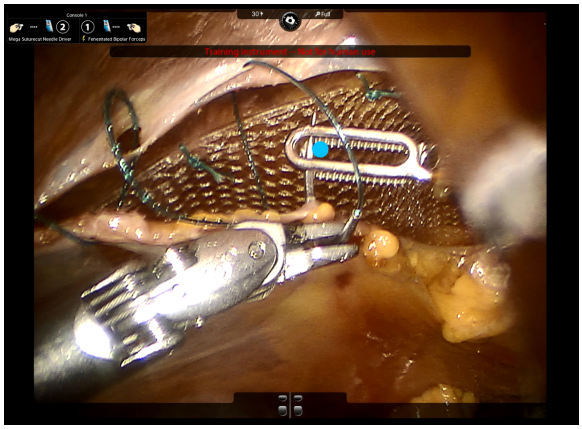
An EyeTech VT2 mini eye gaze tracker was placed within the *da Vinci Si*® Surgical System. This eye gaze tracker has a capture rate of 80 Hz, and a reported accuracy of 0.5 degrees. The EyeTech Quick Link SDK was used for communicating with the eye gaze tracker. All data processing and classifier training was carried out with Matlab 2016.

6.2.1 Dataset

Data was collected from seven RAS surgeons operating the *da Vinci Si*® Surgical System. Informed consent was obtained from all individual surgeons included in the study (Western IRB, Inc. Puyallup, WA). Three inexperienced surgeons (no RAS procedures but prior laparoscopic and open procedures) performed porcine exercises, and four intermediate surgeons (>50 RAS procedures) performed cadaver exercises. Two of the three surgeons who performed porcine exercises specialized



(a)



(b)

Figure 6.1: Frame of the endoscope video for the (a) porcine laboratory and (b) cadaver laboratory with gaze position overlaid (blue circle).

in colorectal surgeries. The remaining surgeon specialized in general surgery. The surgeons who performed cadaver exercises all specialized in gynecology.

Each of the surgeons performed multiple training tasks that focused on dissection, retraction, and suturing skills. During each exercise, eye gaze, instrument kinematics, system events, and endoscope video were recorded. An example of the endoscope video is shown in Figure 6.1.

All of the data collected except for eye gaze was obtained directly from the *da*

Vinci Si® Surgical System. An EyeTech VT2 mini eye tracker (EyeTech Digital Systems, Mesa, AZ) was used to measure eye gaze. Eye gaze data was synchronized with the *da Vinci Si*® data stream. The gaze tracker was calibrated to each subject twice once at the beginning of the lab and a second time approximately 2-3 hours later halfway through the training activities. During calibration, each subject looked at a target which moved to 9 different positions within the surgical scene. If the second calibration was more accurate than the first, it was used to record eye gaze data.

6.2.2 Eye Gaze Pre-Processing

The raw eye gaze data was processed in order to extract instances of fixations and saccades, which are more meaningful than raw eye gaze position data. Saccades are fast movements of the eye, while fixations occur when the gaze is held steady over an area [54]. Visual perception occurs during fixations, and areas of interest can be derived from fixation data. First, a velocity-based algorithm was implemented for saccade detection [54]. The velocity-based algorithm involves setting a velocity threshold which defines whether a saccade or fixation has occurred. The calculation of eye gaze velocity from raw gaze position data is prone to noise, so a Savitzky-Golay filter [55] was applied, based on the saccade detection algorithm described in [49]. This filter works by fitting a polynomial to a specific number of data samples, and is effective in smoothing data while preserving peaks in the signal. Any detected saccades greater than 1 second were excluded. Next, fixations were identified as the time periods between saccades within the range of 100 milliseconds to 3 seconds. A lower fixation duration threshold of 100 milliseconds is a commonly selected threshold in eye tracking research [38, 54]. The upper duration threshold for saccades and fixations were selected in order to remove erroneous detections. An example of our eye gaze data processing is shown in Figure 6.2.

6.2.3 Camera Movement Classification

Camera movement prediction was treated as a binary classification problem (i.e., camera movement or no-camera movement). Eye gaze and kinematics data streams were segmented into 5 second windows with each window being assigned either to

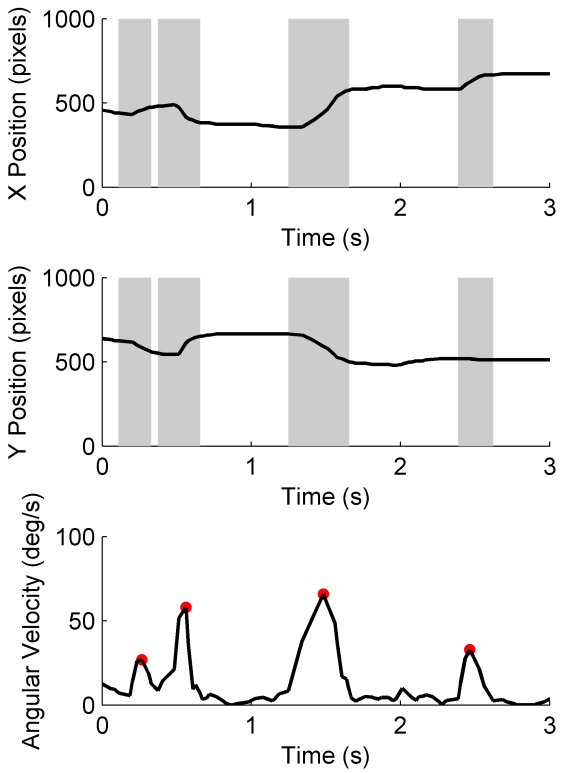


Figure 6.2: Result of saccade detection algorithm. Saccade duration highlighted in gray in (a,b), and saccade peak velocity marked in (c) as red circles.

the camera movement class or the no-camera movement class. A 5 second window size was selected as larger window sizes resulted in unbalanced event data with many more camera movement events than no-camera movement events. This was a result of many fast, consecutive camera movements which would greatly reduce the number of no-camera movement events if a larger window duration was selected. The camera movement class was defined as the time window immediately preceding a camera event derived from system events data, and the no-camera movement class was defined as all other time windows, excluding camera movements . The data was sub-sampled to achieve a balanced dataset and avoid bias in classification.

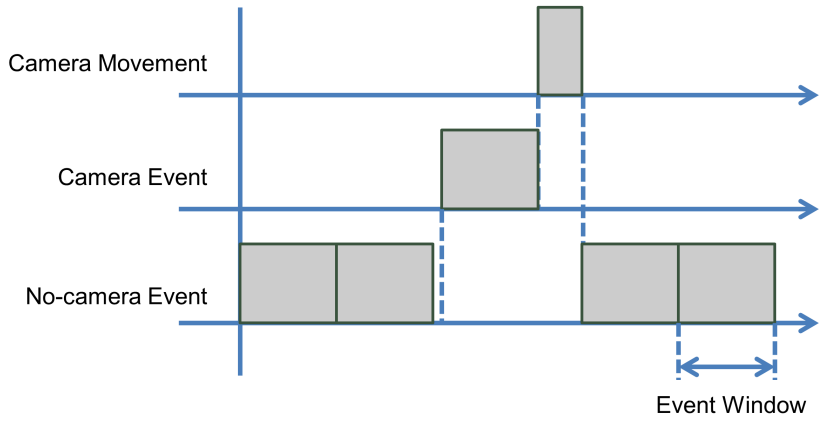


Figure 6.3: Diagram demonstrating how camera events were segmented. The top axis shows the occurrence of a camera movement. The camera event is extracted as the event window prior to the onset of a camera movement and no-camera events are all other event windows.

A random forest algorithm was implemented to perform classification on the segmented data. The random forest classifier was chosen as it is able to handle many features and additionally gives an evaluation of the most important features for classification. The random forest classifier has previously been used in related surgical work-flow research to detect surgical tasks [58]. A branch size of 400 was used as deeper branches did not improve classification results.

Sixty features were calculated; forty gaze features and twenty tool features as shown in Table 6.1. The dwell times for each instrument refer to the total duration of fixations on the instrument tool tips. A circular region of 300 pixels was drawn around each tool tip, and if the fixation fell within this region it was counted towards the dwell time on the instrument. This circular area was defined in order to account for error in the gaze position. If the fixation was located in the region of multiple instruments it was counted towards all of the instruments. We also included general eye gaze fixation and saccade features, as well as tool kinematic features such as position and velocity. The instrument-related features included instrument velocity, distance, and position. All of the instrument features were calculated with respect to the instrument tool tip in the camera reference frame. Three

Table 6.1: Eye gaze and tool features

Eye Gaze Features	Tool Features
Dwell time on tool (#1,#2,#3,all)	Tool (#1,#2,#3,all) mean velocity
Dwell time in the environment	Tool (#1,#2,#3,all) peak velocity
Dwell time ratio on tool (#1,#2,#3,all)	Tool (#1,#2,#3,all) total distance
Dwell time ratio on the environment	Tool (#1,#2,#3,all) mean horizontal position
Mean fixation distance from center pixel	Tool (#1,#2,#3,all) mean vertical position
Latest fixation distance from center pixel	
Fixation frequency	
Fixation duration (mean,total)	
Saccade duration (mean, total)	
Saccade length (mean, total)	
Saccade velocity (mean, peak)	
Mean fixation position (x,y,angle)	
Last fixation position (x,y,angle)	
First-last fixation displacement (x,y,angle)	
Overall gaze displacement (x,y,angle)	
Mean saccade direction (x,y,angle)	
Peak saccade direction (x,y,angle)	
Last saccade direction (x,y,angle)	

instruments contributed to the features since all three separate slave manipulators on the patient-side robot were used through the training activities. Different surgical tasks would require different types of surgical instruments, however the instruments would not be switched between manipulators during a task. The position of each tool tip was calculated with respect to the endoscope camera reference frame, and the mean horizontal and vertical positions were used as features. The motivation behind selecting the horizontal and vertical positions in the image reference frame is that endoscope adjustment may occur because the surgeon is operating on the edge of the field of view.

The performance of the classifier was determined with a ten-fold validation scheme, with 10% of the data set aside for testing during each iteration. The reported results are an average of all iterations. Despite having a small number of surgeons, data was collected from each surgeon over several hours in a day, resulting in a large number of camera events. The total data size was 402 events for

porcine tasks and 2496 events for cadaver tasks. The total time of surgical activity across all subjects was approximately 25 hours.

6.2.4 Camera Direction Classification

Camera direction classification was carried out using two separate categorical classifiers- a horizontal direction model and a vertical direction model. In this manner, the classifiers predict whether camera movement is up/down or left/right. This approach is a simple first step towards a fully automated endoscope system. We first aim to find the features which are indicative of camera movement direction, and in the future will build a more complete model for predicting where a camera should move.

For each window in the camera movement class described previously in Section 6.2.3, the direction of camera movement is calculated. For all camera movements $\overrightarrow{cam} = (x, y)$, if the camera moves in the upward direction (positive y), then it is classified as moving up, otherwise it is moving down. If the camera moves in the positive x direction, it is classified as moving right, otherwise it is moving left. Since the horizontal and vertical classifiers are separate, a single data point will be "left" or "right" as well as "up" or "down".

The same eye gaze and tool features were used as in camera movement classification (see Table 6.1. Data was again sub-sampled before classification in order to have a balanced data set. Both cadaver and porcine data sets were combined for this analysis, as only using camera events results in a smaller dataset. In total, there were 1262 events used, 785 cadaver events and 86 porcine events.

6.3 Results

6.3.1 Camera Movement Prediction Results

The random forest classifier for camera movement prediction showed a 72.6% classification rate for porcine tasks, 71.3% for cadaver tasks, and 77.7% for the combined data set. The ROC curves for all three cases are shown in Figure 4. The ROC curves can be used to gauge classifier performance. The vertical axis represents the sensitivity of the classifier, and the horizontal axis represents the 1-specificity

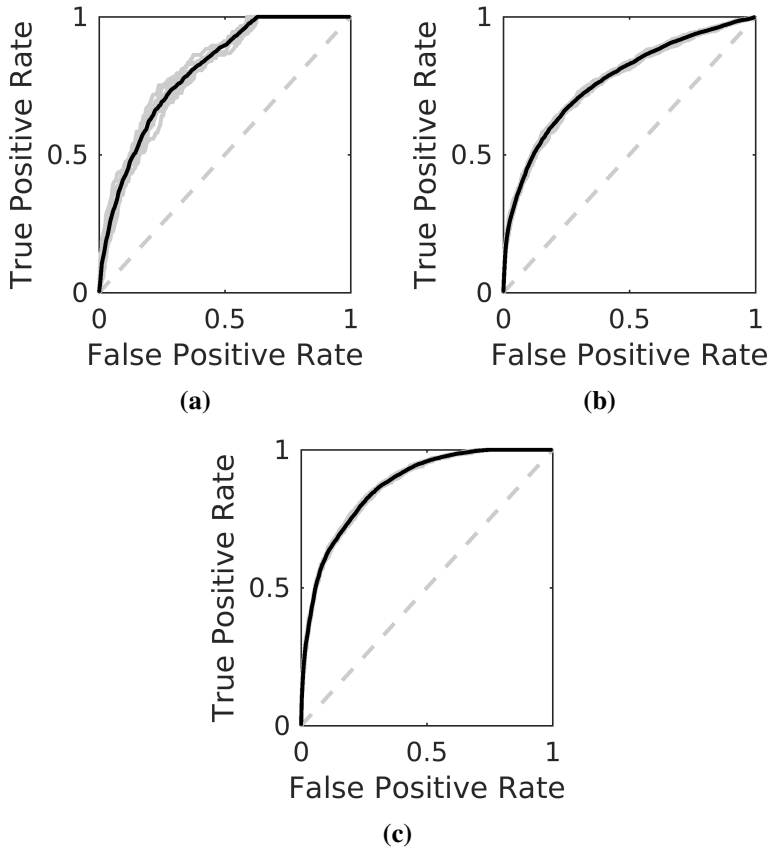


Figure 6.4: ROC curves for the classifier on (a) porcine, (b) cadaver, and (c) both porcine and cadaver tasks. Mean curve (black), individual folds (grey), and chance (dashed grey).

of the classifier. The area under an ROC curve is a common metric used to numerically compare ROC curves, and indicates the accuracy of a classifier, with 1 being a perfect classifier. The Area Under the Curve (AUC) was 0.81 for porcine tasks, 0.77 for cadaver tasks, and 0.87 for the combined data set.

The accuracy achieved by the random forest classifier is comparable to results from related classification problems in literature. In [26], James et al. use eye gaze and instrument information in a Parallel Layer Perceptor model to segment the surgical steps of a laparoscopic cholecystectomy. They collected data from three surgeons performing a total of fifteen procedures on porcine models. They

obtained a segmentation accuracy of 75% with eye-gaze and instrument information. Additionally, in [3] Ahmidi et al. use eye gaze and instrument information to classify surgical steps in a functional endoscopic sinus surgery using hidden Markov Models. They evaluated their model with 11 subjects in a cadaver model. They achieved a task identification accuracy of 74.98% for experts and 80.36% for novices. Classification performance was better for the porcine data set than the cadaver data set. This is perhaps because the cadaver data set was more unstructured, resulting in less predictable behaviour.

One benefit of implementing the random forest classifier is that we are able to compute the importance of each feature for classification. We examined the most important features used by the classifiers over the ten iterations in the ten-fold validation scheme. These features had the most influence on the classification decision between camera and non-camera events. The mean saccade velocity for the combined dataset was the most important feature for seven iterations, and the second most important feature for three iterations. The distribution of mean saccade velocity is shown in Figure 6.5. The mean saccade velocity tends to be higher prior to a camera movement, possibly as the surgeon is less focused on any area on interest and is instead gazing on different anatomy within the operative scene. Conversely, a specific eye gaze feature that was found to be not important was the dwell time on instrument #2. In fact, all of the instrument dwell time features were ranked in the lower half of importance. This shows that the surgeon is observing and interacting with their instruments in the same manner regardless of whether the endoscope needs to be re-oriented.

6.3.2 Camera Direction Prediction Results

For both cadaver and porcine datasets, the random forest classifier for camera direction prediction had a 54.71% classification rate for predicting horizontal camera movement, and a 55.77% classification rate for predicting vertical camera movement. For only the cadaver dataset, the classifier had a 54.52% classification rate for horizontal camera movement and a 55.20% classification rate for vertical camera movement. Lastly, for the porcine dataset, the classifier had a 60.59% classification rate for horizontal camera movement and a 59.72% classification rate for

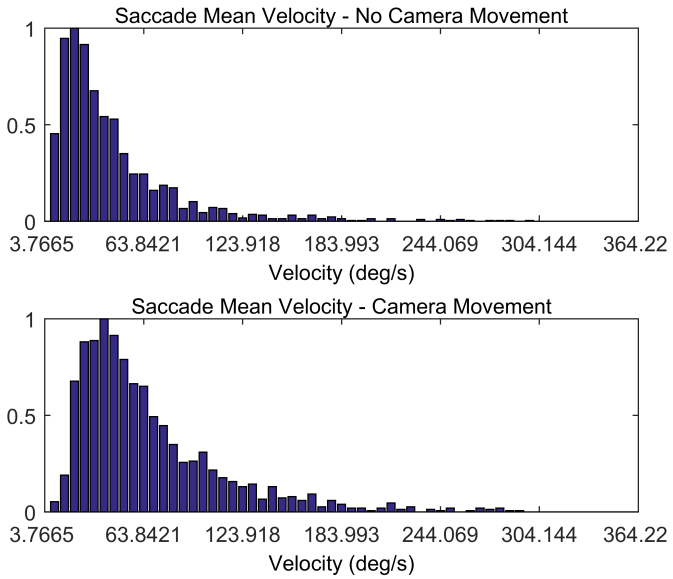


Figure 6.5: Normalized histograms of mean saccade velocity for camera movement and no-camera movement events.

vertical camera movement.

The ROC curves for both cases are shown in Figure 6.6. For both cadaver and porcine datasets, the mean AUC was 0.55 for horizontal camera movement, and 0.44 for vertical camera movement. For the cadaver dataset, the mean AUC was 0.45 for horizontal camera movement and 0.45 for vertical camera movement. For the porcine dataset, the mean AUC was 0.39 for horizontal camera movement and 0.59 for vertical camera movement.

The accuracy results are low for both horizontal direction prediction and vertical direction prediction in all datasets. Prediction is higher for the porcine dataset, but as the data set size was much smaller. The number of camera events from novice surgeons is very low the results are less stable.

6.4 Discussion

There are certain limitations for our analysis, first of all the eye gaze data was classified into two categories (fixations and saccades), while there are other types

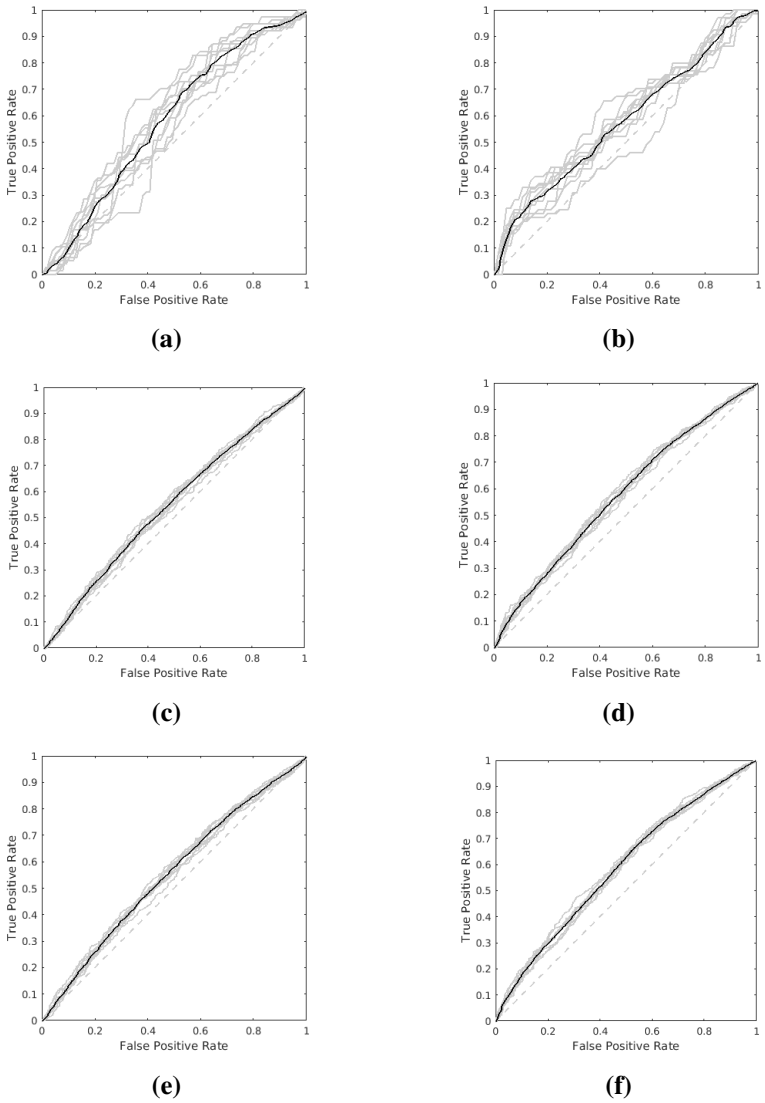


Figure 6.6: ROC curves for the classifier for prediction of (a) porcine horizontal direction, (b) porcine vertical direction, (c) cadaver horizontal direction, (d) cadaver vertical direction, (e) both porcine and cadaver horizontal direction, and (f) both porcine and cadaver vertical direction.. Mean curve (black), individual folds (grey), and chance (dashed grey).

of eye movements not explored such as smooth pursuit, which is when gaze follows the trajectory of a moving object. Additionally, we segmented the eye gaze and tool data collected into 5 second time windows in order to summarize the raw data and calculate metrics such as mean and peak feature values. It will be valuable to further analyze the effects of the window size, and to explore the use of moving windows.

Our results show that eye gaze metrics combined with instrument metrics can be utilized with reasonable accuracy for the prediction of camera movement events. We have shown that certain eye gaze features such as saccade velocity can be significant factors in the classification of camera movement events.

Regarding the prediction of camera movements, we achieved a poor classification accuracy (no greater than chance prediction). We believe that further exploration needs to be done to determine useful features for predicting where camera movements should be directed. Additionally, a continuous data stream as opposed to binning data into time window segments may improve results, by adding more contextual information regarding where surgeons are looking before and during a camera movement.

6.5 Conclusion

In this work, we examined the use of eye gaze metrics to predict whether or not surgeons intend to adjust their viewpoint. Our results suggest that eye gaze measures in combination with tool kinematics can be used to effectively predict camera movement events. This is an important step for camera automation during robotic surgery as our model can be used to initiate camera movement in a natural manner. We have begun preliminary work towards predicting the direction of camera movement. However, further analysis of the eye gaze and tool metrics needs to be carried out in order to improve results. The endoscope acts as an extension of the surgeons eyes, and thus it is intuitive to use eye gaze for controlling camera position. A predictive model for camera direction control would be an improvement on current autonomous endoscope systems which use tool position to determine the optimal camera position.

A well-known problem with eye gaze-based control is the “Midas Touch” prob-

lem, described in [25]. Gaze-based interfaces need to be able to distinguish between when a user intends to make an action and when they are passively observing the interface. The behavioural model we have developed addresses this issue, and allows for a data-driven approach to recognize the intent of a surgeon to move their camera.

In the end, we believe gaze plays a critical role in accurately modelling surgeon behaviour since humans utilize gaze to extract information from their environment to plan and to execute movements and interactions. In future studies we plan to collect larger datasets to investigate classification performance across multiple levels of surgeon expertise and across more similar surgical exercises. We will implement our predictive model in a gaze-based automated camera system.

Chapter 7

Conclusions

The main objective of this thesis was to develop and evaluate a stereo eye gaze tracker use with the da Vinci surgical robot. A second objective was to demonstrate the feasibility of a gaze-based control framework for a surgical robot system. A third objective was to use eye gaze information to predict camera movement. We were able to meet our objectives with the following contributions:

- The design of eye gaze tracking hardware for the da Vinci surgeon console.
- A software algorithm for detecting eye features (pupil and corneal light reflections) using image gradients instead of image intensity.
- A calibration method for determining the point of gaze in 3D using the da Vinci Research Kit.
- The design of a study protocol for evaluating the accuracy of the eye gaze tracker in 2D and 3D.
- A hand-eye calibration procedure for finding the transformation between the da Vinci patient-side manipulator coordinate frame and the endoscope camera coordinate frame.
- A data-driven behavioural model for predicting when and where surgeons move the endoscope. This model is built using a random forest classifier

trained with eye gaze and instrument kinematic data from clinically realistic tasks.

In this thesis, an eye gaze tracker was designed for use with the da Vinci ®Surgical System. The eye gaze tracker is placed onto the top of the da Vinci surgeon console, and uses IR imaging for illuminating the eyes. Image processing techniques are used to find the pupil and corneal light reflections (glints) from the eye gaze tracking cameras. The eye gaze tracking camera runs at 100 FPS, and overall the frame rate of the eye gaze tracker is 90 FPS due to processing time. To estimate eye gaze, the vector between the pupil and glints is mapped to known co-ordinates. A 2D calibration method where users are shown target calibration points within the da Vinci surgeon console. The pupil-glint vector can then be mapped to the target positions using a second order polynomial. A method for carrying out 3D eye gaze tracking was designed which maps the disparity between the left and right eyes to tool depth. To calibrate for 3D eye gaze, the surgeon gazes at the tool at several positions. A two-step 3D calibration was designed where the surgeon first completes a 2D calibration, and then a 3D calibration. A one-step 3D calibration was also designed where the surgeon only completes the 3D calibration. A study was carried out with 11 participants to measure the accuracy of the eye gaze tracker. The 2D accuracy was between 26.61 pixels to 49.32 pixels depending on the number of calibration targets. The 3D accuracy was 17.18 ± 8.13 mm for a one-step calibration and 13.33 ± 3.23 mm for a two-step calibration.

The eye gaze tracker was then employed in a control framework in which eye gaze position is used for aiding instrument position. The objective for this work was to demonstrate gaze-based motor control in a full da Vinci surgical system. A spring-damper haptic force is applied to the surgeon's hands through the da Vinci master manipulators, and guides their hands to move the instrument towards the point of gaze. This control framework was evaluated in a preliminary study with 5 subjects performing a peg placement task. There was no significant reduction in task completion time with or without gaze control.

Thirdly, a behavioural model was designed in order to predict when and how surgeons move their endoscopes. Data was collected from 7 surgeons performing clinically realistic tasks on porcine and cadaver models. The types of data collected

were eye gaze position, instrument kinematics, system event data, and a video stream. The eye gaze data was processed to segment fixation and saccade metrics. Using the eye gaze and instrument data, a random forest classifier was trained to classify the occurrence of camera movement events. The accuracy achieved for classifying the occurrence of camera movement events was 72.6% for porcine tasks, 71.3% for cadaver tasks, and 77.7% for a combined data set.

Bibliography

[1] 3d gaze tracking method using purkinje images on eye optical model and pupil. *Optics and Lasers in Engineering*, 50(5):736 – 751, 2012. → pages 6, 16

[2] ElSe : Ellipse Selection for Robust Pupil Detection in Real-World Environments. *Eye Tracking Research & Applications*, pages 123–130, 2016. → pages 10

[3] N. Ahmidi, G. D. Hager, L. Ishii, G. Fichtinger, G. L. Gallia, and M. Ishii. Surgical Task and Skill Classification from Eye Tracking and Tool Motion in Minimally Invasive Surgery. *Lecture notes in computer science*, 63(63): 295–302, 2010. → pages 3, 86, 94

[4] M. S. Atkins, G. Tien, R. S. Khan, A. Meneghetti, and B. Zheng. What do surgeons see capturing and synchronizing eye gaze for surgery applications. *Surgical innovation*, 20(3):241–248, 2013. → pages 75

[5] F. Behrens and O. J. Griisser. Smooth pursuit eye movements and optokinetic nystagmus elicited by intermittently illuminated stationary patterns. *Experimental Brain Research*, 37(2):317–336, 1979. → pages 8

[6] A. Belardinelli, O. Herbort, and M. V. Butz. Goal-oriented gaze strategies afforded by object interaction. *Vision research*, 106:47–57, 2015. → pages 75

[7] A. Bihlmaier and H. Woern. Automated endoscopic camera guidance: A knowledge-based system towards robot assisted surgery. In *ISR/Robotik 2014; 41st International Symposium on Robotics*, pages 1–6, June 2014. → pages 2

[8] B. C. Daugherty, A. T. Duchowski, D. H. House, and C. Ramasamy. Measuring vergence over stereoscopic video with a remote eye tracker. In

Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications, pages 97–100. ACM, 2010. → pages 12

[9] A. T. Duchowski, D. H. House, J. Gestring, R. Congdon, L. Świrski, N. A. Dodgson, K. Krejtz, and I. Krejtz. Comparing estimated gaze depth in virtual and physical environments. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 103–110. ACM, 2014. → pages 14

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. → pages 40

[11] A. W. Fitzgibbon, R. B. Fisher, et al. A buyer's guide to conic fitting. *DAI Research paper*, 1996. → pages 31

[12] K. Fujii, A. Salerno, K. Sriskandarajah, K. W. Kwok, K. Shetty, and G. Z. Yang. Gaze contingent cartesian control of a robotic arm for laparoscopic surgery. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3582–3589, 2013. → pages 85

[13] D. A. Goss and R. W. West. *Introduction to the Optics of the Eye*. Butterworth-Heinemann Medical, 2001. → pages 6

[14] G. Gras and G. Z. Yang. Intention recognition for gaze controlled robotic minimally invasive laser ablation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2431–2437, Oct 2016. → pages 16

[15] G. Gras, K. Leibrandt, P. Wisanuvej, P. Giataganas, C. A. Seneci, M. Ye, J. Shang, and G. Z. Yang. Implicit gaze-assisted adaptive motion scaling for highly articulated instrument manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4233–4239, May 2017. → pages 16

[16] J. Hallett. 3-d imaging guides surgical operations, 2001. URL http://www.vision-systems.com/articles/print/volume-6/issue-5/features/medical-imaging/3-d-imaging-guides-surgical-operations.html. [Online; accessed June 1, 2017]. → pages 66

[17] D. W. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):478–500, 2010. → pages 7

[18] J. Heikkila. *Accurate camera calibration and feature based 3D reconstruction from monocular image sequences*. PhD thesis, 1997. → pages 51

[19] J. M. Henderson, J. R. Brockmole, M. S. Castelhano, and M. Mack. Visual saliency does not account for eye movements during visual search in real-world scenes. *Eye movements: A window on mind and brain*, pages 537–562, 2007. → pages 75

[20] C. Hennessey. *Eye-gaze tracking with free head motion*. PhD thesis, 2005. → pages 10, 30

[21] C. Hennessey and P. Lawrence. Noncontact binocular eye-gaze tracking for point-of-gaze estimation in three dimensions. *IEEE transactions on biomedical engineering*, 56(3):790–799, 2009. → pages 15

[22] C. A. Hennessey and P. D. Lawrence. Improving the accuracy and reliability of remote system-calibration-free eye-gaze tracking. *IEEE Transactions on Biomedical Engineering*, 56(7):1891–1900, 2009. → pages 11, 12, 49

[23] C. H. J. Howard. A test for the judgment of distance. *American Journal of ophthalmology*, 2(9):656–675, 1919. → pages 65

[24] N. E. Institute. Facts about the cornea and corneal disease, 2016. URL https://www.nei.nih.gov/health/cornealdisease. [Online; accessed June 1, 2017]. → pages 6

[25] R. J. Jacob. Eye tracking in advanced interface design. *Virtual environments and advanced interface design*, pages 258–288, 1995. → pages 98

[26] A. James, D. Vieira, B. Lo, A. Darzi, and G. Z. Yang. Eye-gaze driven surgical workflow segmentation. *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, 10: 110–117, 2007. → pages 86, 93

[27] A. M. Jarc and M. J. Curet. Viewpoint matters: objective performance metrics for surgeon endoscope control during robot-assisted surgery. *Surgical Endoscopy*, pages 1–11, 2016. → pages 84

[28] P. Kazanzides, 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 *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6434–6439. IEEE, 2014. → pages 76

[29] B. W. King, L. A. Reisner, A. K. Pandya, A. M. Composto, R. D. Ellis, and M. D. Klein. Towards an autonomous robot for camera control during laparoscopic surgery. *Journal of Laparoendoscopic & Advanced Surgical Techniques*, 23(12):1027–1030, 2013. → pages 85

[30] A. Lanata, A. Greco, G. Valenza, and E. P. Scilingo. On the tridimensional estimation of the gaze point by a stereoscopic wearable eye tracker. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 2283–2286. IEEE, 2015. → pages 14

[31] H. O. Latif, S. Member, N. Sherkat, A. Lotfi, and S. Member. Teleoperation through Eye Gaze (TeleGaze): A Multimodal Approach. In *Robotics and Biomimetrics (ROBIO)*, pages 711–716, 2009. ISBN 9781424447756. → pages 85

[32] C. Lee and Y.-f. Wang. Image Analysis for Automated Tracking in Robot-Assisted Endoscopic Surgery. pages 88–92, 1994. → pages 85

[33] D. Li, D. Winfield, and D. Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) - Workshops*, 3:79–79, 2005. → pages 10, 38

[34] S. Li, X. Zhang, F. J. Kim, R. Donalisio da Silva, D. Gustafson, and W. R. Molina. Attention-aware robotic laparoscope based on fuzzy interpretation of eye-gaze patterns. *Journal of Medical Devices*, 9(4):041007–10, 2015. → pages 2

[35] S. Li, X. Zhang, F. J. Kim, R. Donalisio da Silva, D. Gustafson, and W. R. Molina. Attention-Aware Robotic Laparoscope Based on Fuzzy Interpretation of Eye-Gaze Patterns. *Journal of Medical Devices*, 9(4): 041007–041007–10, 2015. → pages 85

[36] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings. International Conference on Image Processing*, volume 1, pages I–900–I–903 vol.1, 2002. → pages 34

[37] F. Lu, Y. Sugano, T. Okabe, and Y. Sato. Adaptive linear regression for appearance-based gaze estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(10):2033–2046, 2014. → pages 9

[38] B. R. Manor and E. Gordon. Defining the temporal threshold for ocular fixation in free-viewing visuocognitive tasks. *Journal of neuroscience methods*, 128(1):85–93, 2003. → pages 88

[39] O. Mohareri, C. Schneider, and S. Salcudean. Bimanual telerobotic surgery with asymmetric force feedback: A davinci® surgical system implementation. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4272–4277. IEEE, 2014. → pages 76

[40] E. D. Montag. Rods and cones, 2014. URL http://www.cis.rit.edu/people/faculty/montag/vandplite/pages/chap_9/ch9p1.html. [Online; accessed June 1, 2017]. → pages 7

[41] C. Morimoto, D. Koons, A. Amir, and M. Flickner. Pupil detection and tracking using multiple light sources. *Image and Vision Computing*, 18: 331–335, 2000. → pages 11

[42] C. H. Morimoto and M. R. M. Mimica. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding*, 98(1): 4–24, 2005. → pages 12

[43] G. Mylonas, A. Darzi, and G. Yang. Gaze contingent depth recovery and motion stabilisation for minimally invasive robotic surgery. *Medical Imaging and Augmented Reality*, (October):311–319, 2004. → pages 16, 23

[44] G. P. Mylonas, A. Darzi, and G. Z. Yang. Gaze-contingent control for minimally invasive robotic surgery. *Computer Aided Surgery*, 11(5): 256–266, 2006. → pages 86

[45] G. P. Mylonas, K. W. Kwok, A. Darzi, and G. Z. Yang. Gaze-contingent motor channelling and haptic constraints for minimally invasive robotic surgery. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5242 LNCS(PART 2):676–683, 2008. → pages 2, 16, 17

[46] G. P. Mylonas, K. W. Kwok, D. R. C. James, D. Leff, F. Orihuela-Espina, A. Darzi, and G. Z. Yang. Gaze-Contingent Motor Channelling, haptic constraints and associated cognitive demand for robotic MIS. *Medical Image Analysis*, 16(3):612–631, 2012. → pages 2, 17, 76

[47] D. P. Noonan, G. P. Mylonas, J. Shang, C. J. Payne, A. Darzi, and G. Z. Yang. Gaze contingent control for an articulated mechatronic laparoscope.

In *2010 3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2010*, pages 26–29, 2010. → pages 85

[48] D. P. D. Noonan, G. P. G. Mylonas, A. Darzi, and G.-z. Yang. Gaze contingent articulated robot control for robot assisted minimally invasive surgery. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1186–1191, 2008. → pages 16

[49] M. Nyström and K. Holmqvist. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior research methods*, 42 (1):188–204, 2010. → pages 88

[50] K. Omote, H. Feussner, A. Ungeheuer, K. Arbter, G.-q. Wei, and J. Ru. Self-guided robotic camera control for laparoscopic surgery compared with human camera control. *American Journal of Surgery*, 177(4):321–324, 1999. → pages 85

[51] T. Pfeiffer, M. E. Latoschik, and I. Wachsmuth. Evaluation of binocular eye trackers and algorithms for 3d gaze interaction in virtual reality environments. *JVRB-Journal of Virtual Reality and Broadcasting*, 5(16), 2008. → pages 14, 15

[52] P. L. Rosin. Analysing error of fit functions for ellipses. *Pattern Recognition Letters*, 17(14):1461–1470, 1996. → pages 41

[53] D. D. Salvucci and J. H. Goldberg. Identifying Fixations and Saccades in Eye-Tracking Protocols. *Proceedings of the Eye Tracking Research and Applications Symposium*, pages 71–78, 2000. → pages 8

[54] D. D. Salvucci and J. H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on Eye tracking research & applications*, pages 71–78. ACM, 2000. → pages 88

[55] A. Savitzky and M. J. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964. → pages 88

[56] M. H. Sodergren, F. Orihuela-Espina, J. Clark, A. Darzi, and G.-Z. Yang. A hidden Markov model-based analysis framework using eye-tracking data to characterise re-orientation strategies in minimally invasive surgery. *Cognitive Processing*, 11(3):275–283, 2010. → pages 86

[57] D. L. Sparks and E. J. Barton. Neural control of saccadic eye movements. *Current Opinion in Neurobiology*, 3(6):966 – 972, 1993. → pages 8

[58] R. Stauder, A. Okur, L. Peter, A. Schneider, M. Kranzfelder, H. Feussner, and N. Navab. Random forests for phase detection in surgical workflow analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8498 LNCS:148–157, 2014. → pages 90

[59] D. Stoyanov, G. P. Mylonas, and G. Z. Yang. Gaze-contingent 3D control for focused energy ablation in robotic assisted surgery. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5242 LNCS(PART 2): 347–355, 2008. → pages 16

[60] Y. Sugano, Y. Matsushita, Y. Sato, and H. Koike. An incremental learning method for unconstrained gaze estimation. *European Conference on Computer Vision*, pages 656–667, 2008. → pages 9

[61] S. Suzuki and K. be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985. → pages 31, 46

[62] L. Swirski, A. Bulling, and N. Dodgson. Robust real-time pupil tracking in highly off-axis images. *Etra*, pages 1–4, 2012. → pages 40

[63] K. Takemura, Y. Kohashi, T. Suenaga, J. Takamatsu, and T. Ogasawara. Estimating 3d point-of-regard and visualizing gaze trajectories under natural head movements. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, pages 157–160. ACM, 2010. → pages 14

[64] K.-h. Tan, D. J. Kriegman, and N. Ahuja. Appearance-based eye gaze estimation. *Proc. WACV*, pages 191–195, 2002. → pages 9

[65] B. W. Tatler, C. Kirtley, R. G. Macdonald, K. M. A. Mitchell, and S. W. Savage. *The Active Eye: Perspectives on Eye Movement Research*, pages 3–16. Springer International Publishing, Cham, 2014. → pages 75

[66] F. Timm and E. Barth. Accurate eye centre localisation by means of gradients. *Visapp*, pages 125–130, 2011. → pages 11

[67] F. Timm and E. Barth. Accurate eye centre localisation by means of gradients. *Visapp*, pages 125–130, 2011. → pages 35

[68] F. Timm and E. Barth. Accurate , fast , and robust centre localisation for images of semiconductor components. *Image Processing: Machine Vision Applications IV*, 7877(0):787705–787705–10, 2011. → pages 35, 36, 37

[69] P. M. Tostado, W. W. Abbott, and A. A. Faisal. 3d gaze cursor: Continuous calibration and end-point grasp control of robotic actuators. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3295–3300, May 2016. → pages 15

[70] A. Villanueva and R. Cabeza. Models for gaze tracking systems. *J. Image Video Process.*, 2007(3):4:1–4:16, Nov. 2007. → pages 12

[71] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–511–I–518, 2001. → pages 34

[72] M. Visentini-Scarzanella, G. P. Mylonas, D. Stoyanov, and G. Z. Yang. "i-BRUSH: A Gaze-Contigent Virtual Paintbrush for Dense 3D Reconstruction in Robotic Assisted Surgery". *Medical Image Computing and Computer-Assisted Intervention (MICCAI '09)*, pages 353–360, 2009. → pages 16

[73] R. I. Wang, B. Pelfrey, A. T. Duchowski, and D. H. House. Online 3d gaze localization on stereoscopic displays. *ACM Transactions on Applied Perception (TAP)*, 11(1):3, 2014. → pages 14, 15

[74] O. Weede, H. Mönnich, and B. Müller. An Intelligent and Autonomous Endoscopic Guidance System for Minimally Invasive Surgery. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5762–5768, 2011. → pages 85

[75] G.-q. Wei and G. Hirzinger. Real-Time Visual Servoing for Laparoscopic Surgery. *IEEE Engineering in Medicine and Biology Magazine*, 16(1): 40–45, 1997. → pages 85

[76] D. Zhu, T. Gedeon, and K. Taylor. "Moving to the centre": A gaze-driven remote camera control for teleoperation. *Interacting with Computers*, 23(1): 85–95, 2011. → pages 86

Appendix A

Eye Gaze Tracking Parameters

A.1 Region of Interest (ROI) Settings

Parameter	Value	Range	Units
Rough ROI	120 x 80		pixels
Fine ROI	120 x 120		pixels
Glint ROI	100 x 50		pixels

A.2 Pupil Detection Settings

A.2.1 Gradient Eye Localization

Parameter	Value	Range	Units
Scale	25		%
Number of step sizes	10		Steps
Minimum step size	10^0.001		pixels
Maximum step size	10^0.5	pixels	

A.2.2 Starburst

Parameter	Value	Range	Units
Maximum ray length	$1.5\times$ *previous pupil radius*		pixels
Number of rays	140		Rays
Glint threshold	$\mu + 1.5\sigma$ *of intensity*	$200 < x < 250$	

A.2.3 RANdom SAmple Consensus (RANSAC)

Parameter	Value	Range	Units
Number of iterations	200		iterations
Early termination threshold	90		%
Inlier error threshold	3		pixels

A.3 Glint Detection

Parameter	Value	Range	Units
Glint threshold	$\mu + 1.5\sigma$ *of intensity*	$200 < x < 250$	
Glint area		$0 < x < 60$	$pixels^2$
Sort angle threshold	45		$^\circ$
Sort distance threshold	70		% of template distance