TOWARD INTEGRATION OF A SURGICAL ROBOTIC SYSTEM WITH AUTOMATIC TRACKING, TOOL GESTURE AND MOTION RECOGNITION

by

Jeff Kuang-chen Hsu  
B.A.Sc., Simon Fraser University, 2003

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

MASTER OF APPLIED SCIENCE

In the  
School  
of  
Engineering Science

© Jeff Kuang-chen Hsu 2007

SIMON FRASER UNIVERSITY

Spring 2007

All rights reserved. This work may not be reproduced in whole or in part, by photocopy or other means, without permission of the author.
Name: Jeff Kuang-chen Hsu  
Degree: Master of Applied Science  
Title of Thesis: Toward Integration of a Surgical Robotic System with Automatic Tracking, Tool Gesture and Motion Recognition

Examinig Committee:
Chair: Dr. Lesley Shannon  
Assistant Professor of Engineering Science

Dr. Shahram Payandeh  
Senior Supervisor  
Professor of Engineering Science

Dr. Ghassan Hamarneh  
Supervisor  
Assistant Professor of Computing Science

Dr. M. Stella Atkins  
Internal Examiner  
Professor of Computing Science

Date Defended/Approved: Nov. 30, 2006
DECLARATION OF
PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website <www.lib.sfu.ca> at: <http://ir.lib.sfu.ca/handle/1892/112>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author’s written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada
ABSTRACT

Laparoscopic surgery is performed by entering a patient’s abdominal area with two surgical tools and a laparoscope through small incisions. An assistant is required to hold and move the laparoscope to keep the operation site in view while the surgeon manipulates the two surgical tools. Timely, accurate, and stable adjustments of the laparoscope cannot be guaranteed in lengthy operations due to human fatigue. We propose a novel system with a robot that replaces the human assistant and automatically tracks the tip of a surgical tool. This robot can also be used as a part of a future telesurgical setup. The surgeon may want to have automated access to various medical data and images pertaining to the patient. A novel surgeon-computer interface that features real-time laparoscopic image processing is proposed. The proposed system allows the surgeon to retrieve medical information by recognizing surgical tool gestures and motions.
ACKNOWLEDGEMENTS

Thanks to my senior supervisor, Dr. Shahram Payandeh, for his guidance and encouragement throughout this thesis work. This work would not be possible without his enthusiasm and knowledge of inter-disciplinary research. I also appreciate the time and efforts he put into helping me solve all the problems encountered during the development of this project.

Thanks to my supervisor Dr. Ghassan Hamarneh for his advice and support throughout this thesis work. He brought in substantial expertise in digital image processing, which helped tremendously in the tool motion recognition function of this system.

Thanks to Dr. Stella Atkins for examining this thesis and Dr. Lesley Shannon for chairing my defence.

Thanks to Dr. Doris of Surrey Memorial Hospital for providing feedback on my research and arranging a visit to a real operating room. Thanks to Ternei Li, the designer of the laparoscope-holding robot, for her help on the laparoscope-holding robot control.

Finally, I want to thank my family for their spiritual support and love.
TABLE OF CONTENTS

Approval ........................................................................................................................ ii
Abstract ................................................................................................................................ iii
Acknowledgements ........................................................................................................ iv
Table of Contents .......................................................................................................... v
List of Figures ................................................................................................................... vii
List of Tables .................................................................................................................... xiii
List of Acronyms ............................................................................................................. xv
Chapter 1 Introduction ...................................................................................................... 1
  1.1 Minimally Invasive Surgery ..................................................................................... 1
  1.2 Problem Statement and Motivations ....................................................................... 3
    1.2.1 Camera Positioning ....................................................................................... 3
    1.2.2 Intuitive Surgeon-computer Interface ............................................................... 7
    1.2.3 Telesurgery ...................................................................................................... 15
  1.3 Objectives and Outline ............................................................................................ 19
  1.4 Contributions .......................................................................................................... 21
Chapter 2 Modeling and Control of the Laparoscope-Holding Robot ................................. 23
  2.1 System Overview .................................................................................................... 23
  2.2 Kinematics Modelling and Inverse Kinematics of the Robot ................................... 25
  2.3 Robot Control Program Modifications .................................................................... 41
    2.3.1 Camera Frame to World Frame Transformation ............................................... 41
    2.3.2 Clearance between Laparoscope and Surgical Grasper ...................................... 44
  2.4 System Main Components Examination Results .................................................... 46
    2.4.1 TCP/IP Network Connection .......................................................................... 47
    2.4.2 Robot Control Module ................................................................................... 49
    2.4.3 Robot Control Program .................................................................................. 51
    2.4.4 The Laparoscope-Holding Robot .................................................................... 55
Chapter 3 Automatic Surgical Tool Tracking Improvements ............................................. 64
  3.1 Experimental Setup Change .................................................................................... 65
  3.2 X and Y Coordinate Unit Conversion ..................................................................... 70
    3.2.1 The Pixel-to-Millimetre Conversion Factor ....................................................... 71
    3.2.2 The Pixel-to-Millimetre Unit Conversion Process ............................................. 75
    3.2.3 Results and Analysis ..................................................................................... 75
LIST OF FIGURES

Figure 1.1: Laparoscope surgery................................................................. 2
Figure 1.2: Laparoscope placement in various MIS procedures............. 4
Figure 1.3: The Intuitive Surgical da Vinci® surgical system.................. 5
Figure 1.4: The Computer Motion ZEUS robotic surgical system........... 6
Figure 1.5: The 4-DOF laparoscope-holding robot................................. 7
Figure 1.6: The imaginary operating room with surgical tool gesture and motion recognition................................................................. 8
Figure 1.7: Interactive superimposition of internal organs on a minimally invasive surgery view to see the tumour (in green) and veins (in blue) in transparency ................................................................................. 9
Figure 1.8: The first telesurgical operation performed in 2001............... 16
Figure 1.9: Conceptual master-slave telesurgery system schematic........... 17
Figure 1.10: Conceptual telesurgery system consisted of haptic devices and laparoscope- and surgical-tool-holding robots.......................... 18
Figure 2.1: The laparoscopic surgery robotic system............................... 24
Figure 2.2: The spherical mechanism laparoscope-holding robot............. 26
Figure 2.3: The schematics drawing of the spherical mechanism robot platform ................................................................. 29
Figure 2.4: The world coordinate frame \{W\}.............................................. 30
Figure 2.5: The angle of actuating coordinate frame \{A\} ......................... 31
Figure 2.6: The link coordinate frame \{L\}.................................................. 32
Figure 2.7: The platform coordinate frame \{P\}.......................................... 33
Figure 2.8: The camera coordinate frame \{C\}.......................................... 34
Figure 2.9: The branch link angles ............................................................ 35
Figure 2.10: Branch link radii .................................................................. 35
Figure 2.11: The actuating angles ............................................................. 36
Figure 2.12: The middle joint angles ....................................................... 37
Figure 2.13: The current and next platform and middle joint vectors........ 40
Figure 2.14: The drive and driven directions.......................................... 40
Figure 2.15: The transformation between camera frame and world frame... 43
Figure 2.16: Clearance between destination point and goal point............ 45
Figure 2.17: The demo1 graphical user interface ...................................... 46
Figure 2.18: The laptrack graphical user interface .................................... 47
Figure 2.19: 3D coordinates of the position marker on the surgical grasper shown on laptrack.

Figure 2.20: The relationship between world frame, camera frame, and the laparoscope-holding platform.

Figure 2.21: Motor control module.

Figure 2.22: Motor control hierarchy.

Figure 2.23: The motors and gear sets.

Figure 2.24: The motors and gear sets.

Figure 2.25: Comparison of O1 calculated by demo1 and Matlab.

Figure 2.26: Comparison of O2 calculated by demo1 and Matlab.

Figure 2.27: Comparison of O3 calculated by demo1 and Matlab.

Figure 2.28: Comparison of O4 calculated by demo1 and Matlab.

Figure 2.29: Test and measured point 1 comparison for the laparoscope-holding robot.

Figure 2.30: Test and measured point 2 comparison for the laparoscope-holding robot.

Figure 2.31: Test and measured point 3 comparison for the laparoscope-holding robot.

Figure 2.32: Test and measured point 4 comparison for the laparoscope-holding robot.

Figure 2.33: Test and measured point 5 comparison for the laparoscope-holding robot.

Figure 2.34: Test and measured point 6 comparison for the laparoscope-holding robot.

Figure 2.35: Test and measured point 7 comparison for the laparoscope-holding robot.

Figure 2.36: Test and measured point 8 comparison for the laparoscope-holding robot.

Figure 2.37: The driving shaft, platform, middle joint, platform joint, and the six link pieces of the three branches.

Figure 2.38: Branch 2 and the driving shaft.

Figure 3.1: The original experimental setup with plain white background and mock-up tool.

Figure 3.2: Real laparoscopic surgery environment.

Figure 3.3: The new experimental setup with plastic stomach model and real surgical tools.

Figure 3.4: Inside of the stomach model with real surgical tools.

Figure 3.5: The light source and camera unit.

Figure 3.6: The position marker design.

Figure 3.7: The z coordinate is the distance between the laparoscope and the surgical grasper.
Figure 3.8: Position marker x and y coordinate unit conversion flow chart
Figure 3.9: Setup for determining the pixel-to-millimetre conversion factor at various distances.
Figure 3.10: Relationship between separation, distance z, and focal length/plane.
Figure 3.11: Number of pixels appearing on screen representing 10 mm actual separation vs. distance.
Figure 3.12: Position marker x and y coordinate estimation and unit conversion experimental setup.
Figure 3.13: The laptrack surgical tool tracking z coordinate estimation errors.
Figure 3.14: Test done at the distance of 30 mm with the position marker placed at (-10, 10).
Figure 3.15: Wider field of view at the distance of 70 mm with the position marker at (0, 0).
Figure 3.16: Average errors in coordinate estimation and unit conversion at each distance.
Figure 3.17: Possible collision between laparoscope and stationary surgical tool while tracking the moving tool.
Figure 3.18: Two possible paths for the laparoscope to go around an obstacle surgical tool.
Figure 3.19: The roll, pitch and yaw angles of a surgical tool.
Figure 3.20: The 30 mm long obstacle tool markers with 20 mm gaps in between.
Figure 3.21: Obstacle marker on the left having larger size than tool tip and position marker.
Figure 3.22: The obstacle tool marker and quadrilateral obstacle tool model.
Figure 3.23: No potential collision if both the start and goal points are not more than 20 mm lower than the obstacle.
Figure 3.24: The worst case scenario of the laparoscope moving across the obstacle tool.
Figure 3.25: Potential collision if either or both of the start and goal positions are more than 20 mm lower than the obstacle.
Figure 3.26: Scenario 1 - Start point lower than the obstacle tool.
Figure 3.27: Scenario 1 - Goal point lower than the obstacle tool.
Figure 3.28: Scenario 1 - two path points being generated above the start point and goal point to avoid potential collisions.
Figure 3.29: Scenario 2 - start point higher than the obstacle tool.
Figure 3.30: Scenario 2 - goal point lower than the obstacle tool.
Figure 3.31: Scenario 2 - one path point being generated above the goal point to avoid collisions.
Figure 3.32: Scenario 3 - start point lower than the obstacle tool.
Figure 3.33: Scenario 3 - goal point higher than the obstacle tool.
Figure 3.34: Scenario 3 – one path point being generated above the start point to avoid collisions ................................................................. 96
Figure 3.35: Obstacle avoidance function testing results ................................................. 97
Figure 3.36: Not enough light reflected off the position marker and grasper tool tip to be recognized ....................................................... 98
Figure 3.37: From left to right, the obstacle marker, the grasper tool tip, and the position marker ................................................................. 99
Figure 3.38: Grasper tool tip being closer to the laparoscope than the obstacle tool ................................................................. 100
Figure 3.39: Obstacle tool marker being broken by the grasper ..................................... 101
Figure 3.40: Grasper tool tip blob being broken by the obstacle tool marker ................. 102
Figure 3.41: Parts of two obstacle markers being in the field of view .......................... 102
Figure 3.42: Only part of the obstacle marker being in the field of view ....................... 103
Figure 4.1: Laparoscopic surgery performed with long-stem surgical tools through laparoscope .................................................................... 107
Figure 4.2: The nine commonly used surgical tools ...................................................... 108
Figure 4.3: The eleven surgical tool gestures experimented ......................................... 109
Figure 4.4: The surgical tool tip marker ..................................................................... 110
Figure 4.5: The surgical tool gesture recognition stages ............................................. 111
Figure 4.6: Target tool segmentation with morphological operations and thresholding ......................................................................................... 115
Figure 4.7: Target tool segmentation stage flow chart .................................................. 117
Figure 4.8: Compactness of different shapes .................................................................. 119
Figure 4.9: Compactness of open grasper and closed grasper ..................................... 119
Figure 4.10: Roughness of different shapes ................................................................. 120
Figure 4.11: Roughness of double triangle and closed grasper .................................. 120
Figure 4.12: Elongation of open grasper and closed grasper ....................................... 121
Figure 4.13: The Feret diameters ................................................................................. 121
Figure 4.14: Feret elongation of open grasper and closed grasper ............................. 121
Figure 4.15: The ten scan lines drawn to determine the maximum and minimum colour change counts a surgical tool gesture .................. 122
Figure 4.16: Colour change count regions for the single tool gestures and the overlapping double-tool gestures ................................................. 123
Figure 4.17: Feed-forward neural network ................................................................... 127
Figure 4.18: Non-overlapping double-tool gesture open scissors and closed grasper being separated into two single tool gestures: open scissors and closed grasper ................................................................. 128
Figure 4.19: The seven orientations of single-tool gestures ........................................ 130
Figure 4.20: The seven orientations of overlapping double-tool gestures ................. 131
Figure 4.21: The NN training iteration ....................................................................... 133
Figure 4.22: The original images and final binarized images by the two methods at different distances.

Figure 4.23: A look at surgical tool gesture recognition function testing from outside.

Figure 4.24: The open grasper gesture being recognized at natural position.

Figure 4.25: Recognition rates of the seven open grasper gesture orientations.

Figure 4.26: The closed grasper gesture being recognized at natural position.

Figure 4.27: Recognition rates of the seven closed grasper gesture orientations.

Figure 4.28: The closed scissors gesture being recognized at natural position.

Figure 4.29: Recognition rates of the seven closed scissors gesture orientations.

Figure 4.30: The open scissors gesture being recognized at natural position.

Figure 4.31: Recognition rates of the seven open scissors gesture orientations.

Figure 4.32: The cross gesture being recognized at middle position.

Figure 4.33: Recognition rates of the seven cross gesture orientations.

Figure 4.34: The triangle gesture being recognized at middle position.

Figure 4.35: Recognition rates of the seven triangle gesture orientations.

Figure 4.36: The double triangle gesture often being recognized as open grasper.

Figure 4.37: Recognition rates of the seven double triangle gesture orientations.

Figure 4.38: One overlapping double-tool gesture being broken into two single-tool gestures by the shadow of the scissors cast on the grasper.

Figure 4.39: The open scissors and open grasper gesture being recognized.

Figure 4.40: The closed scissors and open grasper gesture being recognized.

Figure 4.41: Overall surgical tool gesture recognition rates for the seven single-tool and overlapping double-tool gestures.

Figure 5.1: The four 2D motion pattern projections.

Figure 5.2: The 3D movements of the position marker on the grasper.

Figure 5.3: The 3D position marker motion trajectory sample points.

Figure 5.4: The 2D projection of the position marker motion trajectory sample points.

Figure 5.5: The eight possible discrete orientations of displacement vectors.

Figure 5.6: The motion pattern represented by discrete displacement vectors.

Figure 5.7: The four possible discrete orientations of displacement vectors.

Figure 5.8: An open grasper gesture activates the surgical tool motion recognition function.

Figure 5.9: The 3D line segment motion trajectory of the position marker on the grasper.

Figure 5.10: The 3D line segment position marker motion trajectory sample points.

Figure 5.11: The 2D projection of the position marker line segment motion trajectory sample points.
Figure 5.12: The 3D two line segments and an angle motion trajectory of the position marker on the grasper.

Figure 5.13: The 3D position marker two line segments and an angle motion trajectory sample points.

Figure 5.14: The 2D projection of the position marker two line segments and angle motion trajectory sample points.

Figure 5.15: The 3D loop motion trajectory of the position marker on the grasper.

Figure 5.16: The 3D position marker loop motion trajectory sample points.

Figure 5.17: The 2D projection of the position marker loop motion trajectory sample points.

Figure 5.18: The line segment tool motion being recognized.

Figure 5.19: The line segment tool motion performed along the tool axis.

Figure 5.20: Recognition results of line segment tool motion performed along and not along tool axis.

Figure 5.21: The two line segments and an angle tool motion being recognized.

Figure 5.22: The curve tool motion being recognized.

Figure 5.23: The loop tool motion being recognized.

Figure 5.24: Surgical tool motion recognition testing results inside the plastic stomach model.

Figure 5.25: Surgical tool motion recognition testing results outside the plastic stomach model.

Figure 5.26: The tool gesture recognition activation area.

Figure 5.27: The 'Display MRI' function being selected with an open grasper tool gesture.

Figure 5.28: The 'Display XRAY' function being selected with an open grasper tool gesture.

Figure 5.29: The 'Display CAT' function being selected with an open grasper tool gesture.

Figure 5.30: The 'EXIT' function being selected with an open grasper tool gesture.

Figure 5.31: The function selection being confirmed with a loop tool motion.

Figure 5.32: The function selection being denied with a line segment tool motion.

Figure 6.1: Using smaller obstacle markers with a different colour to represent the z coordinates at different parts of the obstacle tool.

Figure 6.2: Contour of a surgical grasper tool tip.

Figure 6.3: The two consecutive frames for both optical flow scenarios.

Figure 6.4: Optical flow velocity fields.

Figure 6.5: The consecutive frames for both AD1 scenarios.

Figure 6.6: Positive, negative, and absolute AD1’s of the two scenarios.
LIST OF TABLES

Table 2.1: demo1 and Matlab inverse kinematics calculation results comparison........52
Table 2.2: Test and measured points comparison for the laparoscope-holding robot. .................................................................56
Table 3.1: Number of pixels appearing on screen of 10 mm actual separation vs. distance..............................................................73
Table 3.2: Position marker x and y coordinate unit conversion testing results. ..........79
Table 4.1: Tool gesture recognition confusion matrix........................................151
<table>
<thead>
<tr>
<th>Glossary Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active contour</td>
<td>Computer-generated curves that move within images to find object boundaries.</td>
</tr>
<tr>
<td>Blob</td>
<td>Connected regions within an image</td>
</tr>
<tr>
<td>Cholecystectomies</td>
<td>The surgical removal of the gallbladder.</td>
</tr>
<tr>
<td>Gesture recognition</td>
<td>Recognizing human gestures with mathematical algorithms.</td>
</tr>
<tr>
<td>Laparoscope</td>
<td>An instrument for examining visually the interior of a bodily canal or a hollow organ such as the colon, bladder, or stomach.</td>
</tr>
<tr>
<td>Inverse kinematics</td>
<td>The process of determining the parameters of a jointed flexible object in order to achieve a desired pose. Similar Terms: inverse kinematics problem, reverse kinematics.</td>
</tr>
<tr>
<td>Morphological filters</td>
<td>Operations or filters built upon lattice theory and topology.</td>
</tr>
<tr>
<td>Motion recognition</td>
<td>Includes sensing and analyzing of object motions.</td>
</tr>
<tr>
<td>Robot</td>
<td>Mechanical system under automatic control that performs operations such as handling and automation.</td>
</tr>
<tr>
<td>Spherical mechanism</td>
<td>Mechanism in which all points of its links describe paths located on concentric spheres.</td>
</tr>
<tr>
<td>Surgical tool/instruments</td>
<td>Tools used to perform surgical tasks such as knives, scissors, forceps, graspers, retractors, clamps and staplers.</td>
</tr>
<tr>
<td>Telerobotic</td>
<td>The area of robotics that is concerned with the control of robots from a distance, chiefly using wireless connections (like Wi-Fi and similar) or the Internet.</td>
</tr>
<tr>
<td>Telesurgery</td>
<td>Surgical procedures performed over a distance.</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>3D</td>
<td>3 Dimensional</td>
</tr>
<tr>
<td>ADI</td>
<td>Accumulative Difference Image</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>BCI</td>
<td>Brain Computer Interface</td>
</tr>
<tr>
<td>CCD</td>
<td>Charged-coupled Device</td>
</tr>
<tr>
<td>CR</td>
<td>Computed Radiography</td>
</tr>
<tr>
<td>DOF</td>
<td>Degree of Freedom</td>
</tr>
<tr>
<td>EGT</td>
<td>Eye Gaze Tracking</td>
</tr>
<tr>
<td>EMG</td>
<td>electromyogram</td>
</tr>
<tr>
<td>FDA</td>
<td>U.S. Food and Drug Administration</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HCl</td>
<td>Human Computer Interface</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>MIS</td>
<td>Minimally Invasive Surgery</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>OF</td>
<td>Optical Flow</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PMD</td>
<td>Precision MicroDynamics Inc.</td>
</tr>
<tr>
<td>PRM</td>
<td>Pattern Recognition Model</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

This chapter starts with a brief introduction to minimally invasive surgery. Section 1.2 describes the motivations of this research project by presenting reviews on the current technologies on three major issues of laparoscopic surgery: laparoscopic camera positioning, surgeon-computer interface, and telesurgery. Our proposed laparoscopic surgery robotic system, which focuses on improving the current form of laparoscopic surgery by addressing these three issues, is also introduced. Section 1.3 describes the objectives of this research project and the outline of this documentation. Section 1.4 lists my contributions to this research project.

1.1 Minimally Invasive Surgery

A minimally invasive surgery (MIS) is a medical procedure carried out by entering the body through the skin, a body cavity or an anatomical opening with the smallest damages possible to these structures. Compared with conventional open surgery, MIS is performed with less operative trauma for the patient. It is also less expensive, reduces hospitalization time (1-3 days vs. 7-10 days), causes less pain and scarring (tiny 10 mm marks vs. 160-250 mm scars), and reduces complications related to surgical trauma, speeding up recovery (1-2 weeks vs. 8 weeks or more) [1].

Many types of medical procedures are considered minimally invasive, such as hypodermic injection, air-pressure injection, subdermal implants, endoscopy,
percutaneous surgery, laparoscopic surgery, arthroscopic surgery, cryosurgery, microsurgery, keyhole surgery, endovascular surgery (such as angioplasty), coronary catheterization, permanent spinal and brain electrodes, and stereotactic surgery. Special medical equipment may be used, such as fiber optic cables, miniature video cameras and special surgical instruments handled via tubes inserted into the body through small openings in its surface [2]. Images of the interior of the body are transmitted to an external monitor for the surgeon to diagnose, visually identifying internal features and performing surgery on the patient as shown in Figure 1.1.

![Figure 1.1: Laparoscope surgery.](image)

Among many different types of MIS, laparoscopic surgery, pioneered by Dr. Camran Nezhat in 1985, is the focus of this research project. Allen [3] studied the economics of laparoscopic surgery and pointed out that the average direct cost of laparoscopic cholecystectomies per case is about $1,400, while the average cost of open cholecystectomies is $5,300 each. According to Poole et al. [4], MIS shows significant advances in operative and post-operative care of patients. The technical aspects of
laparoscopic surgery have taken a lot of credit for this. As a result, MIS procedures have been rapidly adopted in recent years and have become standard methods for cholecystectomy. More than 90% of cholecystectomy surgeries are now performed through MIS. Demand from patients has also accelerated the evolution of research and technology on this subject.

1.2 Problem Statement and Motivations

1.2.1 Camera Positioning

As beneficial as laparoscopic surgery may be for patients, it presents a certain group of challenges for surgeons. This includes the loss of direct viewing of the surgical site, the loss of tactile perception and unnatural hand-eye coordination. To make it even more difficult, the direct viewing is replaced by distorted endoscopic video camera, and manipulation is done through long stem surgical instruments. Surgeons have to operate by looking at monitors displaying cavities at awkward angles with a restricted range of views. Moreover, they can no longer touch tissues with their hands, as only those long and thin surgical instruments are small enough to go through the tiny incision holes. Therefore, surgeons have to develop new hand-eye coordination and instrument manipulation skills to cope with the less ergonomic operating environment and limited working space.

The process of minimally invasive surgery has been augmented by specialized tools for decades. However, in recent years, electronic and computer-aided tools have been developed to make the procedure more manageable and accessible for surgeons. Since the surgeons' eyes are replaced by cameras in laparoscopic surgery, camera
positioning has become a critical issue. Figure 1.2 shows the camera placement in various procedures such as the typical laparoscopic surgery performed on stomach, endoscopic spine surgery, endoscopic sinus surgery, and laparoscopic knee surgery. Most laparoscopic surgeries are performed with an assistant, shown on the right in Figure 1.1, who holds the laparoscope and moves it at the surgeon’s command. However, it is difficult for the assistant to maintain accurate and timely laparoscope adjustments in lengthy procedures due to fatigue.
Two robotic surgical systems have received FDA clearance to be marketed in the United States: The da Vinci® Surgical System, made by Intuitive Surgical, Inc. [5] of Sunnyvale, California, is cleared to perform surgery under the direction of a surgeon. The ZEUS Robotic Surgical System, made by Computer Motion, Inc. of Goleta, California, has also been cleared by the FDA to assist surgeons.

With the da Vinci® Surgical System shown in Figure 1.3, the surgeon operates while seated at a console viewing a 3D image of the surgical field, and the system translates the surgeon’s hand, wrist and finger movements into precise, real-time movements of surgical instruments held by the robot arms on the patient-side cart. The camera is moved smoothly by a robotic arm, using electromechanical damping to reduce vibrations that would otherwise be produced by shaky human hands.

![Figure 1.3: The Intuitive Surgical da Vinci® surgical system.](image)
ZEUS, shown in Figure 1.4, has three robotic arms that are mounted on the operating table. One robotic arm is called the Automated Endoscopic System for Optimal Positioning Robotic System (AESOP). AESOP is a voice-activated robot used to hold the endoscope. The FDA cleared AESOP to hold and position endoscopes in 1994, and voice activation was added later on. ZEUS differs from the da Vinci® system in that the AESOP part of ZEUS responds to voice commands. For example, a surgeon might say: "AESOP move right." The positioning arm then would move right until the "stop" command is given.

Like the da Vinci® system, the other two arms of ZEUS serve as extensions of the left and right arms of the surgeon. Surgeons sit at a console and wear special glasses that create a three-dimensional image. Unfortunately, the company, Computer Motion Inc., does not exist anymore.

![Figure 1.4: The Computer Motion ZEUS robotic surgical system.](image)

Different from the two commercial systems, our proposed solution to the camera positioning problem is to hold the laparoscope with a 4-DOF robot designed by Li [6] [7] [8] [9] as shown in Figure 1.5. Not only does the 4-DOF robot hold the laparoscope, but
it also automatically tracks the tip of a surgical tool through the camera image so that the operation site is always in the field of view. The robot is described in detail in Chapter 2.

![Control module](image)

**Figure 1.5:** The 4-DOF laparoscope-holding robot.

### 1.2.2 Intuitive Surgeon-computer Interface

Besides the camera-positioning problem, there exist many motivations for improvements to the current form of laparoscopic surgery. What if the surgeon wants to see the patient’s X-ray, MRI scan, or other medical records during surgery? What if the surgeon wants to superimpose a medical image on top of the organ of interest? What if the surgeon needs to start an emergency video conferencing with doctors on the other side of the planet? One way to grant the surgeon’s wishes in the middle of a surgery without him or her shouting to the assistants or releasing the surgical tools to press another button is by means of surgical tool gesture and motion recognition. Gestures can be classified into two types: one is the static gestures that are essentially static postures; the other is the dynamic gesture that involves motions. In our research, surgical tool gestures refer to static gestures, and tool motions refer to dynamic gestures.
We propose an automatic surgical tool gesture recognition system as an intuitive surgeon-computer interface. All the surgeon needs to do is to manipulate the two surgical tools in his or her hands under the camera, and the computer would recognize the tool gestures and motions. The system would then carry out the associated commands to open up a medical image on the monitor without the surgeon taking his or her hands and eyes off the monitor as shown in the imaginary operating room in Figure 1.6. Our surgical tool gesture and motion recognition functions, which are described in detail in Chapters 4 and 5, are capable of recognizing a number of surgical tool gestures and motions, and open an image on the monitor accordingly.

Figure 1.6: The imaginary operating room with surgical tool gesture and motion recognition.

In the future, the surgical tool and motion recognition functions could be extended to activate the augmented reality similar to that developed by IRCAD [10], which links the virtual world with the real world. The aim is to visualize anatomical or pathological structures that cannot be directly seen by superimposing 3D virtual models on the real
patient view as shown in Figure 1.7. To superimpose the virtual models on top of the live camera image, the virtual model has to be scaled to fit the actual size of the organ on the monitor. In addition, landmarks on the virtual model and organs on the monitor must be matched in order to correctly orient the virtual model on top of the camera image. This part is left for future work.

![Figure 1.7: Interactive superimposition of internal organs on a minimally invasive surgery view to see the tumour (in green) and veins (in blue) in transparency.](image)

According to Katkere [11], the use of gestures for telerobotic control is a fairly new human-centred method for man-machine communication that provides an expressive, natural, and intuitive way for humans to control robotic systems. Most researchers concentrate on the direct man-machine interface by human hand or body gesture recognition instead of any form of tool gesture recognition. Why use tools when you can use your hands? However, our case is unconventional. Our groundbreaking approach
recognizes tool gestures through actual manipulation of the surgical tools since the laparoscopic camera only takes images inside the abdominal.

Tool gesture recognition is easier than human hand gesture recognition as surgical tools have fixed colour and shape. Their surface colours do not differ from one another like human skin colours. Thus, colour calibration is not as big an issue in our case. Furthermore, surgical tools do not deform like human hands. In other words, the number of variations of one gesture in the tool gesture case is less than that of human hand gesture case.

However, surgical tool gesture recognition presents its unique challenges, the most difficult being extracting surgical tools from the background. In most human hand gesture recognition literatures, the background is often very simple and constant. In our case, the background, which is composed of body tissues and organs, is more complex. Moreover, the background changes with laparoscope movement and the reflections from body tissues and organs further complicate the surgical tool extracting process.

Various methods on static human gesture recognition and object detection have been developed by researchers over the years. Five common methods are as follows:

- **Morphological Filters and Active Contours**: Xuan et al. [12] employ a two-step scheme for automatic object detection in Computed Radiography (CR) images utilizing morphological filters to extract the foreign objects and active contour models to outline them. The first step is to use morphological filters with structuring elements to effectively distinguish the foreign object candidates from the complex background structures. The second step is to employ active contour models to accurately outline the morphological shapes.
of the suspicious foreign objects, such as tumours, to further reduce the rate of false alarms.

- **Template Matching:** Hu *et al.* [13] discuss a visual recognition system using template matching to interactively control a mobile robot. The robot first identifies the operator by human facial recognition using template matching. The template contains feature values for facial features such as eyebrows, mouth, and nose, and these candidate feature values are evaluated to find the match. Similarly, the robot then determines the actions by analysing the human hand gestures also with template matching. Ng *et al.* [14] develop a gesture recognition system that uses the condensation-based trajectory matching algorithm. The gesture data is collected using a pair of CyberGloves measuring hand-joint angles and three magnetic trackers that determine 3D hand positions. The multi-dimensional gesture data is subsequently recognized by matching against trajectory models using probability measures.

- **Sensor-Based:** Malassiotis *et al.* [15] use a novel 3D sensor that generates a dense range image of the scene to recognize complex hand postures such as those encountered in sign language alphabets. The major advantage of this method is its independence from colour information, which guarantees robust segmentation of the hand under various illumination conditions, and content of the scene.

- **Neural Network:** In the work of Shimada *et al.* [16], a neural network with the Pattern Recognition Model (NN-PRM) is applied to the Japanese Kana
finger spelling recognition. Using this method, feature quantities are extracted from the hand area and fed to the neural network for identification. Yin and Xie [17] create a fast and robust system that colors segments and recognizes hand gestures for human-robot interaction using a neural network. Yuan et al. [18] design a set of command-like gestures for multi-touch technology users with limited range and function in their digits and wrist. The multi-touch pad is a 2D parallelogram electrode array that produces proximity images of fingers and hands touching on the surface. Hand gestures are then recognized by a recurrent neural network. Molder et al. [19] create a system that maps hand gestures to musical parameters in an interactive music performance and virtual reality environment. Data obtained from a sensor glove is passed to a neural network for gesture classification, the gestures are then mapped onto equivalent musical parameters according to a dictionary created by the researchers. The system complements a 3D VRML environment that shows an animated hand model and behaving representations of musical structures. This 3D representation combines with the gesture processing module and the sound generation to allegedly Behaving Virtual Musical Objects.

- **Fuzzy Rule**: Fuzzy rule-based gesture classifiers are often combined with other methods such as neural network and HMM. Su [20] presents a fuzzy rule-based approach to spatio-temporal hand gesture recognition. This approach employs a neural network for selecting templates. Templates for each hand shape are represented in the form of crisp IF-THEN rules that are
extracted from the values of synaptic weights of the corresponding trained neural network. Each crisp IF-THEN rule is then fuzzified by employing a special membership function in order to represent the degree to which a pattern is similar to its corresponding antecedent part. When an unknown gesture is to be classified, each sample of the unknown gesture is tested by each fuzzy rule. The accumulated similarity associated with all samples of the input is computed for each hand gesture in the vocabulary, and the unknown gesture is classified as the gesture yielding the highest accumulative similarity. Fang et al. [21] propose a fuzzy decision tree with HMM for large vocabulary sign language recognition. A one- or two-handed classifier and a hand-shape classifier with little computational cost are first used to progressively eliminate many impossible candidates. Then, an HMM classifier is used to get the final results at the last non-leaf nodes of the fuzzy decision tree that only include a few candidates.

Six dynamic gesture recognition or motion classification methods are also reviewed as follows:

- **Hidden Markov Model**: A prominent dynamic gesture classifier is the HMM. The research of Min et al. [22] uses a Hidden Markov Model to recognize twelve different dynamic gestures in the Korean sign language. Perrin et al. [23] propose a finger gesture recognition system based on an active tracking mechanism. The laser-based system is able to obtain the position of a tracked finger, and recognize finger gestures using HMM. Joslin et al. [24] introduce a method for dynamic hand gesture recognition.
They use a camera to track the hand and finger motions in 2D. Then the information is passed to a 2D-to-3D module that transforms the 2D information into a full 3D hand model. Finally, HMM is used to identify and differentiate between different gestures.

- **Morphological Shape Decomposition**: Choi *et al.* [25] propose an algorithm that tracks a trajectory of centre points in primitive elements extracted by morphological shape decomposition. The trajectory of morphological centre points also includes the information on shape orientation.

- **Hand Configuration Recognition Algorithm**: Hu *et al.* [26] describe an intelligent automated algorithm for tracking the finger pads of a moving hand with movement videos captured by a common web camera. The algorithm applies the principle of HCRA (hand configuration recognition algorithm) and its refinement through a simple yet novel transitional appearance-based model.

- **Displacement Vectors**: A hand motion recognition algorithm (HMRA) is developed to recognize motion pattern. Hienz *et al.* [27] show a video-based analysis system for acquisition and classification of hand-arm motion concerning the German sign language. A colour coded glove and coloured markers at the elbow and shoulder are used, and the sequence of the marker position data is converted into discrete displacement vectors. Motion is then derived from the displacement vectors with rule-based classification.
Fuzzy Associative Inference: Ushida et al. [28] create a real-time human motion recognition method that uses fuzzy associative inference. It transforms space-time patterns into state-transition patterns, which are then recognized by means of fuzzy associative inference using associative memories.

Displacement Trajectories: Sappa et al. [29] focus on labelling a human movement as a walking or running displacement, which are the most frequent types of locomotion. The proposed technique first detects peaks and valleys of points’ trajectories, which are then used to discern whether the movement corresponds to a walking or running displacement.

1.2.3 Telesurgery

Telesurgery has been touted as a solution to medical problems in dangerous or inaccessible locations such as underdeveloped nations and remote territories, whereby surgeons located at a single central hospital can operate several remote machines at distant locations. For example, in western Canada, most medical experts are located in hospitals in major cities such as Vancouver, Calgary, or Edmonton. Telesurgery would enable them to operate on patients living in the remote northern territories such as Yukon Territory and Northwest Territories. The potential for telesurgery has had strong military interest as well, with the intention of providing mobile medical care while keeping trained doctors safe from the battlefield. In short, telesurgery is the future for minimally invasive surgery, and that future is not far away. On September 7, 2001, Professor Marescaux in New York removed the diseased gallbladder of a 68-year-old patient in
As shown in Figure 1.8, the surgeon used a ZEUS robotic surgical system with a high-speed network connection to complete the operation.

Our proposed laparoscope-holding robotic system could be easily expanded to build the conceptual telesurgery system as shown in Figure 1.9. Our current laparoscope-holding robotic system, shown on the upper-right-hand corner, will be the slave end of the complete telesurgery system. We could have two more similar robots holding surgical tools as shown on the bottom-right-hand corner. While the laparoscope-holding robot tracks the surgical tools automatically, the visual feedback from the laparoscope is transmitted to the master end computer through a high-speed network connection. Surgeon manipulates the two 4-DOF haptic devices, also designed by Li [6] [7] [8] [9], at the master end to control the two surgical-tool-holding robots at the slave end.
control signals are sent to the slave end computer through the high-speed network connection.

Figure 1.9: Conceptual master-slave telesurgery system schematic.

Figure 1.10 demonstrates how our conceptual master-slave system would work in telesurgery as if the surgeon were standing right next to the patient. On the right, the surgeon watches the camera image displayed on the monitor and manipulates the two haptic devices on the master end as if he or she were actually manipulating the real surgical tools. The control signals are then sent to the computer on the slave end, which controls the three robots mounted on the same base. On the slave end, the two surgical-tool-holding robots operate on the patient according to the control signals while the third laparoscope-holding robot tracks one of the surgical tools to keep the surgical site within the field of view.
Telesurgery is not the only robotics application in the field of surgical operations. IRCAD [30], in partnership with the CNRS (French National Scientific Research Centre) and the Louis Pasteur University in Strasbourg, is currently working on the development of a new generation robot. The objective is to automate the surgical gesture so that it becomes more secure in a way that surgery can be pre-programmed on the digital copy of the patient. This is achieved by applying the combination of augmented reality and robotics that enables surgeons to plan surgery on the virtual patient model. Such system allows first to detect and then to correct any manoeuvre error on the virtual patient model. The real patient could thus benefit from an optimal surgery with every cut or suturing carefully planned out and with minimized risks. Our conceptual telesurgery system could also be expanded to incorporate such functionalities in the future. With fast computers, advanced robots, and high-speed networks in hand, the boundary of computer-aided laparoscopic surgery is only limited by imagination.
1.3 Objectives and Outline

The main objective of this research project, which is an extension of my Bachelor thesis project [31], is to develop a laparoscopic surgery robotic system that not only automatically tracks a surgical tool and positions the laparoscope with a robot, but also provides a direct and intuitive HCI by recognizing surgical tool gestures and motions. This system is motivated by the idea of improving the existing laparoscopic surgery from both hardware and software perspectives, with the hardware being a novel parallel-mechanism laparoscope-holding robot, designed by Li [6] [7] [8] [9], and its control module, and the software being the multi-purpose image processing program, laptrack, originally developed by Zhang [32] [33], that estimates the 3D coordinates of a surgical tool tip, and recognizes surgical tool gestures and motions all at the same time. Since there are no stereo laparoscopic camera systems available, the depth information is estimated from 2D images. A TCP/IP network connection is established between the computer that controls the robot and the computer that performs the image processing to transmit surgical tool coordinates, tool gestures, and tool motions. This system would provide a solution to the laparoscope positioning problem, pioneer a novel surgical tool gesture and motion recognition as alternative surgeon-computer interface, and mark our first step in exploring the future of minimally invasive surgery - telesurgery.

The main objective is achieved by developing the following specific ideas:

1. Add a camera-frame-to-world frame transformation to transform surgical tool tip coordinates from camera frame to world frame for robot control.

2. Add a clearance between the laparoscope and surgical tool to maintain the size of the visible surgical site.
3. Examine the whole laparoscopic surgery robotic system to find out the source of the laparoscope positioning errors.

4. Modify laptrack to work with a new experimental setup that better mimics a real surgery environment. The current version of the laptrack program only tracks a mock-up tool with a black marker at the tip on a pure white background. The new experimental setup consists of real surgical tools and a mock-up plastic stomach model.

5. Convert the coordinates of surgical tool tip estimated by laptrack from pixels to millimetres.

6. Develop an obstacle avoidance function in laptrack to redirect the laparoscope around an obstacle tool to avoid potential collisions. When the laparoscope-holding robot is moving the laparoscope to track a surgical tool, there could be another surgical tool in its path, causing collisions.

7. Implement a surgical tool gesture recognition function that recognizes eleven single- or double-tool gestures.

8. Develop a surgical tool motion recognition function that recognizes four basic tool motions.

This thesis documentation is divided into seven chapters, of which this introductory chapter is the first. Chapter 2 starts with an overview of the system, followed by the camera-to-world-frame transformation, the clearance between the laparoscope and the surgical tool, and the investigation of the component that causes errors in laparoscope positioning. Chapter 3 describes the improvements made to the image processing.
laptrack program, including experimental setup change, coordinate unit conversion, and the obstacle avoidance function. Chapter 4 introduces and analyzes the surgical tool gesture recognition function, which is capable of recognizing a number of single-tool and double-tool gestures. In Chapter 5, the surgical tool motion recognition function is described and analyzed. Chapter 6 suggests future work that could improve the system performance, especially on the surgical tool motion recognition function. In Chapter 7, the conclusion of the research work is presented. An introduction to neural network and Matlab codes for the suggested surgical tool motion recognition methods in Chapter 6 are attached as appendices at the end.

1.4 Contributions

- Automatic laparoscope surgical tool tracking and laparoscope positioning: Building on my previous work [31], the image processing laptrack program has been improved. It can estimate 3D coordinates of a real surgical tool tip inside a plastic stomach model, convert the unit from pixel to millimetres, and send them to the robot control program demo1 over a TCP/IP network connection, which is the first step toward building a future telesurgery system. At the same time, laptrack also recognizes the locations of possible obstacles and plans alternative path for the laparoscope-holding robot to avoid potential laparoscope-instrument collisions. At the robot control side, demo1 has been improved to transform incoming surgical tool tip coordinates from camera frame to world frame. It also keeps a clearance between the laparoscope and the surgical tool being tracked so that a minimum viewing area of the surgical site is maintained.
Laparoscopic surgical tool gesture and motion recognition: Utilizing thresholding, blob analysis, and neural network, the surgical tool gesture recognition function added to laptrack is capable of recognizing a number of single-tool gestures and a number of overlapping or non-overlapping double-tool gestures. It enables the surgeon to send control commands to our surgical robotic system with surgical tools already in his or her hands. The surgeon does not even have to take his or her eyes off the monitor. This function provides an intuitive method for surgeon-computer interface. The tool gestures form a set of “vocabulary” the surgeon can use in different combinations to form different “sentences” to communicate with the system. In addition, the surgical tool motion recognition function recognizes four simple tool motions, adding more “words” to the set of “vocabulary”. As a result, more “sentences” are available to command the system to carry out more actions. An HCI using a combination of the surgical tool gesture and surgical tool motion recognition functions is implemented. It allows the surgeon to use the open grasper tool gesture and the loop tool motion to select and display different types of medical images on the monitor.
CHAPTER 2

MODELING AND CONTROL OF THE LAPAROSCOPE-HOLDING ROBOT

In tracking the marked surgical grasper, the laparoscope-holding robot is supposed to keep the grasper in the middle of the view at all times. However, serious errors in laparoscope positioning are causing the robot to point the laparoscope far off the designated direction. This chapter presents the investigation of the entire laparoscopic surgery robotic system regarding possible causes for the laparoscope positioning errors. The chapter starts with the system overview. Section 2.2 provides some background information on kinematics modelling and inverse kinematics of the laparoscope-holding robot. Section 2.3 describes two modifications to the robot control program demo1, including camera-frame-to-world-frame transformation and clearance between laparoscope and surgical tool. Section 2.4 discusses the examination of the network connection, the robot control module, the robot control program demo1, and the laparoscope-holding robot. The chapter concludes with the source of errors. A failed attempt to correct the system is also presented.

2.1 System Overview

Shown in Figure 2.1, the laparoscopic surgery robotic system consists of a white PC running the laparoscope tracking program (laptrack), and a black PC running the robot control program (demo1), a laparoscopic camera, a laparoscope-holding robot, and
a number of motor control cards and servo amplifiers that are combined into the motor control module.

Figure 2.1: The laparoscopic surgery robotic system.

The white PC running the C++ image processing program, \textit{laptopack}, is responsible for estimating the 3D coordinates of the position marker on the surgical grasper from the image captured by the laparoscope as well as recognizing surgical tool gestures and tool motions. The 3D coordinates, tool gestures, and tool motion are sent over to the black PC running the C++ motor control program, \textit{denoiol}, on the right through a TCP/IP network connection. The network connection is our first step in emulating telesurgery, in which surgery is performed through a high-speed network with the doctor and patient at two different sites. The motor control software then determines the angles each of the four motors on the laparoscope-holding robot has to turn by
solving the inverse kinematics of the robot presented in the next section. Finally, it sends appropriate control commands to the motor control module that drives the robot to position the laparoscope.

The motor control module consists of a number of motor control cards and servo amplifiers. Its tasks include taking signals from the motor control program, interpreting the signals, and outputting control signals to the individual motors on the robot. The motor control module also receives signals from corresponding motor encoders and feeds them back to the motor control program for feedback control.

The laparoscope-holding robot has four motors driving each of the four degrees of freedom it possesses. Three rotational DOF are for aligning the laparoscope along the vector pointing to the destination coordinates. With the three DOF, the robot can also rotate the laparoscope around its own axis. In other words, the robot can rotate the laparoscope to rotate the image displayed on the monitor. The fourth linear DOF provides a zoom in/out function by moving the laparoscope in and out of the abdominal. Together, the motors drive the robot to position the laparoscope tracking the surgical grasper.

2.2 Kinematics Modelling and Inverse Kinematics of the Robot

In comparison to serial mechanisms, parallel mechanisms have the characteristics of low inertia, high rigidity, compactness, and precise resolution. As shown in Figure 2.2, the patented spherical parallel mechanism laparoscope-holding robot is motivated by the Agile Eye, created by Gosselin et al. [34] [35]. Its most unique feature is that all three branches move on the surface of a virtual sphere. In other words, all of the rotational axes coincide at the centre of the virtual sphere, where the centre of rotation is located. When
Given the destination for the laparoscope to point to, the robot control program calculates individual turn angles and sends out control commands to corresponding motors to move the robot. Recent robotic devices are becoming more and more sophisticated, incorporating several innovative control methods. An interesting
design using a hemispherical device controlled by magnetic fields has been proposed by
Berlelman et al. [36]. Although this device has six degrees of freedom, its workspace is
rather limited (15–20 degrees in rotation, 25 mm in translation) and the control of
magnetic levitation is far from easy and also uncommon. The concept of neural networks
is another popular approach to robot control.

Novakovic [37] proposes an adaptive nonlinear robot control using a feed-forward
neural network. This neural network is trained to imitate an adaptive nonlinear robot
control algorithm, based on the dynamics of a full robot model of RRTR structure.
Sensor-based robot control systems can overcome many difficulties of uncertain models
and unknown environments which limit the domain of application of current robots used
without external sensory feedback. Three of the most significant types are visual, tactile,
and force/torque sensors.

Harashima [38] explores the idea of robot control with visual feedback and/or
cooperation of multiple sensors, and shows such control systems are very promising in
solving problems caused by uncertain modeling and/or unexpected changes in the task
environment.

The conventional way of controlling robots by calculating their forward or inverse
kinematics is still commonly used. The kinematics analysis of spherical 3-DOF parallel
manipulators has been studied extensively by Gosselin et al. [34] [35]. Since the
laparoscope-holding robot is similar to the Agile Eye developed by Gosselin et al. [34]
[35], we decide to control the robot by using the inverse kinematics method.

In general, kinematics analysis can be divided into direct (forward) kinematics
and inverse kinematics. Direct kinematics problems are about calculating the positions
and orientations of end-effectors from the known input joint variables. Similar to human arms, wherein the shoulder, elbow, and wrist joints decide the positions and orientations of hands, joint variables of robots decide the positions and orientations of the end-effectors. On the other hand, inverse kinematics problems involve solving for the joint variables from known positions and orientations of the end-effectors. In our case, we are given the coordinates of the surgical tool that the laparoscope-holding robot has to point the laparoscope to; therefore, solving the inverse kinematics problem is essential to controlling the motion and orientation of the laparoscope-holding robot.

Given the 3D coordinates of a destination point, the turning angles for each of the three branches supporting the platform have to be determined in order to position the laparoscope. Using those three branch angles and the gear ratios between the driven gears on the shafts and the driving gears, the turning angles for each motor can then be computed. Finally, the destination for each axis is set by using the motor control software, including the fourth axis that controls the height of the laparoscope. Command signals are then sent to the motor control module to move the robot.

Kinematics modelling of the robot, as explained in our previously published paper by Hsu et al. [39], is presented first. Figure 2.3 shows the schematic drawing of the laparoscope-holding platform of the robot shown in Figure 2.2. The platform is rotated by the three branches. The unit vector $V_o$ is directing along the concentric actuating axis, which is the drive shaft. The unit platform vectors $P_1, P_2, P_3$ are the axes of the revolute joints connected to the platform. The unit middle joint vectors corresponding to the revolute joints connecting the passive and active links on each branch are $M_1, M_2,$ and $M_3$. The link angles of passive links are defined as $\alpha_i$ ($i=1, 2, 3$). The link angle of
active links are \( \alpha_{ci}, i=1, 2, 3 \). Actuating angles at the joint between the active links and the driving shaft are defined as \( \theta_i, i=1, 2, 3 \).

![Diagram of spherical mechanism robot platform](image)

**Figure 2.3:** The schematics drawing of the spherical mechanism robot platform.

With a given coordinate \((x, y, z)\) for the laparoscope to reach, the three actuating angles \( \theta_i \)'s, which are the angle of the upper branch links that are attached to the drive shaft, are found through the inverse kinematics of the robot. The process is as follows.

Given a set of 3D coordinates, the three new platform vectors \( P_1, P_2, P_3 \) can be determined. Since the three middle joint vectors \( M_1, M_2, M_3 \) are related to the three platform vectors, the three new middle joint vectors can also be calculated. The last step is to calculate the three new actuating angles \( \theta_i \)'s, which can be done by taking the dot products of the three middle joint vectors with the vector of the drive shaft, \( V_a \). The
Actuating angles are also called active link rotation angles, and they are explained later in this section. Then the resulting three new actuating angles are compared with the previous three actuating angles; the differences are the angles each branch has to turn. Multiplying by the gear ratio between the motor gear and the drive shaft gear, the angles for the motors to turn are calculated. The kinematics modelling of the robot described in the following paragraphs would help explain the inverse kinematics process.

**World coordinate frame \( \{W\} \):** World coordinate frame is a fixed frame in which all other frames are referred to. Shown in Figure 2.4, the origin point \( O_\omega \) of the world coordinate frame is located at the pivot centre of the moving platform. This centre is also the incision point on the patient’s abdominal. The three coordinate axes are defined as \( X_\omega \), \( Y_\omega \), and \( Z_\omega \). Positive \( z \)-axis is toward the abdominal wall, and the origin is about 25 mm below the platform. All of the revolute joint vectors and actuating axes meet at the origin point, as this point is located at the centre of the sphere.

![Figure 2.4: The world coordinate frame \( \{W\} \).](image)
Actuating coordinate frame \{A\}: The actuating coordinate frame is a fixed frame. The origin point $O_a$ of the actuating coordinate frame is identical to the location of the origin point of world coordinate frame. In Figure 2.5, the angle of actuating coordinate frame, $\beta$, represents the angle ($3/4\pi$) between z-axis of the world frame and actuating frame. The rotation matrix of actuating coordinate frame can be expressed as:

$$
R_a^w = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \beta & -\sin \beta \\
0 & \sin \beta & \cos \beta
\end{bmatrix}
$$

(2.1)

Figure 2.5: The angle of actuating coordinate frame \{A\}.
Link coordinate frame \( \{L\} \): The link coordinate frame is attached to each link in order to calculate the local coordinate position shown in Figure 2.6. The z-axis is pointing to the spherical centre with the x-axis is parallel to the link arc.

![Figure 2.6: The link coordinate frame \( \{L\} \).](image)

Platform coordinate frame \( \{P\} \): As shown in Figure 2.7, the platform coordinate frame is attached to the moving platform and defined by three branch vectors of the platform. The platform coordinate frame shares its origin point with the world coordinate frame and rotates with respect to the world coordinate frame without any translation. The benefit of this design is that the third vector of the platform \( Z_p \) can be conducted easily by the cross product of \( X_p \) and \( Y_p \).
Camera coordinate frame \( \{C\} \): A moving coordinate frame as shown in Figure 2.8 is attached to the tip of laparoscope because it is convenient to describe camera motion with respect to camera frame rather than the world frame. Since the camera is attached to the moving platform, the camera frame has only one linear DOF with respect to platform and four DOF with respect to world frame. The origin of camera frame at home position is 10 mm below the origin of world frame.
Figure 2.8: The camera coordinate frame \( \{C\} \).

Branch link angles \((\alpha_{11}, \alpha_{21}, \alpha_{31}, \alpha_{12}, \alpha_{22}, \alpha_{32})\): As shown in Figure 2.9, these six angles represent the arc angles of the six links. The angles \(\alpha_{11}, \alpha_{21}, \) and \(\alpha_{31}\) are the arc angles of link \(L_{11}\), which are also the angles between the moving platform and the corresponding middle joints. \(\alpha_{12}, \alpha_{22}, \) and \(\alpha_{32}\) are passive link angles. On the other hand, \(\alpha_{12}, \alpha_{22}, \alpha_{32}\) are active link angles between the middle revolute joints and the actuating axis unit vector \(V_a\).
Figure 2.9: The branch link angles.

Branch link radii ($r_{11}, r_{21}, r_{31}, r_{12}, r_{22}, r_{32}$): These six radii are nested for the three branches. Shown in Figure 2.10, $r_{11}, r_{21}$ and $r_{31}$ are the radii of passive link $L_{11}$ and $r_{12}, r_{22}, r_{32}$ are those of active link $L_{12}$. The branch link radii of the first version laparoscope-holding robot are 120, 113, 106, 99, 92, and 85 mm. The branch link radii of the second version laparoscope-holding robot are 80, 75, 70, 65, 60, and 55 mm.

Figure 2.10: Branch link radii.
Active link rotation angles ($\theta_{a1}$, $\theta_{a2}$, $\theta_{a3}$): Shown in Figure 2.11, active link rotation angles are measured between the $x$-axis of the actuating frame and the $x$-axis of the link coordinate frames of $L_{21}$, $L_{22}$, and $L_{23}$ respectively. An active link rotation angle is positive for counter-clockwise and negative for clockwise. The angles are driven by motor with a gear ratio of 56:24.

![Figure 2.11: The actuating angles.](image)

Middle joint angles ($\theta_1$, $\theta_2$, $\theta_3$): Shown in Figure 2.12, middle revolute angles are measured between passive and active links. The angle between axis $X_i$ and $X_p$ about axis $Z_i$ is positive when rotation is counter-clockwise. Middle revolute angles are passive, which can be calculated from kinematics formulations or measured by potentiometer.
The following paragraphs present an analytical solution through the inverse kinematics of the laparoscope-holding robot. The world coordinate frame \( \{W\} \) and actuating coordinate frame \( \{A\} \) are defined in the previous section. The next step is to define home position and orientation of the camera and moving platform. Rotation matrix, \( R_h \), of the platform home position can be expressed by platform joint vectors \( v_j \), \( v_2 \) and \( v_3 \).

\[
R_h = \begin{bmatrix} v_1 \ & \ \v_2 \ & \ \v_3 \end{bmatrix}; P_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, P_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}, P_3 = \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix}
\]

(2.2)

Therefore,

\[
R_h = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix}
\]

(2.3)
Given a platform joint vector $p_i$, $(i=1, 2, 3)$, and the shaft vector $s$, the middle joint vectors $m_i = \langle x_{mi}, y_{mi}, z_{mi} \rangle$ can be solved by cosine laws. We have three equations

1) $\cos \alpha_i = p_i \cdot m_i = x_{pi} \cdot x_{mi} + y_{pi} \cdot y_{mi} + z_{pi} \cdot z_{mi}$

2) $\cos \beta_i = s \cdot m_i = x_i \cdot x_{mi} + y_i \cdot y_{mi} + z_i \cdot z_{mi}$

3) $x_{mi}^2 + y_{mi}^2 + z_{mi}^2 = 1$

There are two sets of solutions for the middle joint vector of each branch. Only one solution matches the physical mechanism configuration. We choose the set of solutions that is closer to the current configuration.

Then the initial rotate angles $(\theta_i)$ for home position can be solved by DH parameters.

\begin{equation}
\mathbf{m} = \mathbf{R}_i \mathbf{s}
\end{equation}

where, $\mathbf{s} = \begin{bmatrix} 0 \\ -\sin \beta \\ \cos \beta \end{bmatrix}$

\begin{equation}
\mathbf{R}_i = \begin{bmatrix} \cos \theta & -\sin \theta \cos \alpha_i & \sin \theta \sin \alpha_i \\ \sin \theta & \cos \theta \cos \alpha_i & -\cos \theta \sin \alpha_i \\ 0 & \sin \alpha_i & \cos \alpha_i \end{bmatrix}
\end{equation}

At home position, actuating angle $(\theta_i)$ can be acquired by middle joint vectors $m_i$, shaft vector $s$, and active link angle $\alpha_i$. 
Now we can rotate the mobile platform to reach a new orientation by defining a new camera vector \( \mathbf{v}_n = \langle x_{cn}, y_{cn}, z_{cn} \rangle \). We compare the home camera vector \( \mathbf{v}_h \), which is \( \langle 0, 0, 10 \rangle \), with the new camera vector \( \mathbf{v}_n \), which is \( \langle x_{cn}, y_{cn}, z_{cn} \rangle \) to find out the rotation angles about the axes. We can then form an Euler rotation matrix \( \mathbf{R}_{Euler} \) of the platform by Equation (2.12) from Craig [40]. We rotate about \( z \)-axis by \( \alpha \), about \( y \)-axis by \( \beta \), then about \( z \)-axis by \( \gamma \) to reach the new position.

\[
\mathbf{v}_n = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix}
\]

(2.11)

\[
\mathbf{R}_{Euler} = \begin{bmatrix}
\cos \beta \cos \gamma & -
\cos \beta \sin \gamma & \sin \beta \\
\cos \gamma \sin \beta \sin \gamma + \sin \gamma \cos \beta & 
\cos \gamma \sin \beta \cos \gamma - \sin \gamma \sin \beta & 
-\cos \beta \cos \gamma \\
-\sin \gamma \cos \beta \sin \gamma - \cos \gamma \sin \beta & 
\sin \gamma \sin \beta \cos \gamma + \cos \gamma \sin \beta & 
\cos \beta \cos \gamma
\end{bmatrix}
\]

(2.12)

We then multiply the home matrix \( \mathbf{R}_h \) with the Euler rotational matrix \( \mathbf{R}_{Euler} \) to get the \( \mathbf{R}_{new} \), which contains the three new platform joint vectors in its three columns. After finding the new platform joint vectors \( \mathbf{P}_i' \), we apply the same set of three Equations (2.4) to (2.6) to find the three new middle joint vectors \( \mathbf{M}_i' \), as shown in Figure 2.13. Then we are ready to compare the new and current middle joint vectors to determine the three driven gear angels, \( \theta_i \) using dot product.

\[
\mathbf{m}_i \cdot \mathbf{m}_i' = \cos \theta_i
\]

(2.13)
After finding the rotation angle of active link, it is necessary to determine the direction. CCW is defined to be the positive direction and CW is defined to be the negative direction as shown in Figure 2.14.
The last step is to convert the three driven gear angles to drive gear angles by multiplying the gear ratio for each branch. Then we get the corresponding motor rotating angles, $\theta_{\text{motor}}$, to reach the new position. Note that we have to multiply by -1 because the motor rotation direction is the opposite direction of the actuating branch.

$$\theta_{\text{motor}} = -\theta \times \text{gear ratio}$$

(2.14)

### 2.3 Robot Control Program Modifications

The two modifications to the robot control program *demo1* are presented in this section. In my Bachelor thesis [31], the robot was controlled by entering the coordinates to *demo1* in world frame. After the system is integrated, *demo1* is able to take coordinates of the marked surgical grasper through the network connection. However, the coordinates estimated by *laptrack* are in the camera frame, and thus a transformation from the camera frame to the world frame is added to *demo1*. A fixed clearance between the laparoscope and surgical grasper is also added to ensure the robot does not zoom in on the laparoscope too much that the surgical grasper blocks the light. When the light is blocked by a surgical tool, all the surgeon can see through the camera is a big surgical tool tip with the background tissues blacked out, making the surgery difficult to perform.

#### 2.3.1 Camera Frame to World Frame Transformation

The coordinate frame transformation process starts with *laptrack*. The first step of the frame transformation is to transform the coordinates from screen frame $\{S\}$ to camera frame $\{C\}$. The origin of $\{S\}$ is at the top left corner of the screen, while the origin of $\{C\}$ is in the middle of the image with the same orientation. Therefore, the transformation is a simple vector addition.
\[ \ell P = \ell^sP + \ell P_{\text{org}} \]  
(2.15)

where \( \ell P \) is the point in \( \{C\} \), \( \ell^sP \) is the point in \( \{S\} \), and the offset

\[ \ell P_{\text{org}} = \begin{bmatrix} x_{\text{offset}} \\ y_{\text{offset}} \end{bmatrix} \]  
(2.16)

There is no \( z \) component in this transformation because it is estimated by \textit{laptrack} in the world frame.

A transformation matrix \( ^wT \) is added at the beginning of the inverse kinematics calculations in \textit{den101} to transform a set of coordinates from camera frame \( \{C\} \) to world frame \( \{W\} \). According to Craig [40], the complete transformation from camera frame \( \{C\} \) to world frame \( \{W\} \), shown in Figure 2.15, is

\[ ^wP = ^wT \ell^sP, \text{ or } ^wP = ^wR \ell P + ^wP_{\text{org}} \]  
(2.17)

where \( ^wP \) is the set of coordinates in world frame, \( ^wR \) is the rotational matrix that describes \( \{C\} \) relative to \( \{W\} \), \( \ell^sP \) is the set of coordinates in \( \{C\} \), \( ^wP_{\text{org}} \) is the offset between the origins of \( \{C\} \) and \( \{W\} \). We first change \( \ell P \) to its description relative to an intermediate frame which has the same orientation as \( \{W\} \), but whose origin is coincident with the origin of \( \{C\} \). This is done by pre-multiplying by \( ^wR \). We then account for the translation between origins by simply adding \( ^wP_{\text{org}} \). \( ^wT \) can be expressed as

\[
^wT = \begin{bmatrix}
^wR & ^wP_{\text{org}} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  
(2.18)
At home position, \( \{C\} \) and \( \{W\} \) overlap with each other with a 10 mm offset along the z-axis as shown in Figure 2.20, which implies that the tip of laparoscope, which is the origin of camera frame, is at \((0, 0, 10)\) in the world frame. Therefore,

\[
^w R_C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

(2.19)

and

\[
^w p_{CORG} = \begin{bmatrix} 0 \\ 0 \\ 10 \end{bmatrix}
\]

(2.20)

Once the robot starts moving, \(^w R_C \) and \(^w p_{CORG} \) change. As stated before, \(^w R_C \) is the rotation matrix that relates \( \{C\} \) to \( \{W\} \). The rotation matrix \( R_{KUL} \) from the previous section rotates the camera frame from its home position, which overlaps the world frame, to the new camera frame orientation. That is the same as relating the new camera frame to the world frame. Consequently,
After the camera frame is rotated to its new orientation, the z-axis of the frame points to the destination point. This is when the clearance calculation, which is explained in the next section, takes effect. The idea is that the robot would move the laparoscope up or down along the z-axis until the distance between the tip of the laparoscope and the destination point is 50 mm. Thus, the offset between \( \mathcal{W} \) and \( \mathcal{C} \) becomes,

\[
\mathbf{P}_{\mathcal{W} \rightarrow \mathcal{C}} = \begin{bmatrix} x_{\text{lap-tip}} \\ y_{\text{lap-tip}} \\ z_{\text{lap-tip}} \end{bmatrix}
\]  

where \( x_{\text{lap-tip}}, y_{\text{lap-tip}}, \) and \( z_{\text{lap-tip}} \) are the coordinates of the tip of laparoscope in world frame.

### 2.3.2 Clearance between Laparoscope and Surgical Grasper

The 50 mm clearance between laparoscope and surgical grasper is added to ensure the surgeon’s clear view of the operating site. The surgeon would not be able to see the operating site if the laparoscope is right on top of it. The clearance is chosen to be 50 mm due to personal experiences that at a lesser distance, all the camera would see is a big grasper tool tip. The grasper tool tip would also block the light coming out of the tip of laparoscope, causing a lack of efficient lighting to the surgical site. Therefore, when a set of 3D destination coordinates \((x, y, z)\) in the world frame comes in, demo1 rotates the platform, which rotates the laparoscope and camera frame attached to it, such that the z-axis of camera frame points to the destination point, \(P_d\), as shown in Figure 2.16. Then, the control program calculates the length of vector \(\langle x, y, z \rangle\)
Next, the goal point

\[ p_g = \left( \frac{L-5}{L}, \frac{(L-5)x}{L}, \frac{(L-5)y}{L}, \frac{(L-5)z}{L} \right) \]  

which is 50 mm away from the destination point \((x, y, z)\) and along the \(z\)-axis of the camera frame is calculated. Finally, the laparoscope is driven either up or down along the camera frame \(z\)-axis, depending on the previous position of laparoscope tip, to reach the goal point, \(p_g\), and \(p_z\) becomes the new laparoscope tip point \((x_{lap\_tip}, y_{lap\_tip}, z_{lap\_tip})\). In the case where the laparoscope has to back up past the origin to keep the 50 mm clearance, it stops at the origin. Once the laparoscope reaches the goal point, it can be moved up or down independently along the same vector with the 'Up' and 'Down' control buttons on the demo1 GUI shown in Figure 2.17.

![Diagram of laparoscope movement](image)

**Figure 2.16: Clearance between destination point and goal point.**
2.4 System Main Components Examination Results

The three main components of the surgical robotic system that could contribute to the laparoscope positioning errors are the motor control module, the motor control program *demo1*, and the laparoscope-holding robot. The other two main components of the system, namely the *laptrack* program and the TCP/IP network connection, can be eliminated from the list of suspects because testing is done by entering the destination coordinates directly to *demo1*, not by sending coordinates from the *laptrack* program over the TCP/IP connection. In other words, without coordinate estimation errors from *laptrack* and possible lost packets during network transmissions, the isolated robot control side of the system is already producing more than tolerable errors in laparoscope positioning. Nevertheless, for the purpose of completeness, the precision of the TCP/IP network connection is reviewed, and the accuracy of the *laptrack* program will be discussed in the next chapter.
2.4.1 TCP/IP Network Connection

The TCP/IP network connection between the two computers is set up as a server-client model. Once the connection is established, the client, *laptrack*, shown in Figure 2.18, keeps sending messages containing the estimated 3D coordinates of the position marker on surgical grasper and the recognized surgical tool gestures and motions. The server, *demol*, shown in Figure 2.17, keeps receiving the messages and reads the contents.

![Figure 2.18: The laptrack graphical user interface.](image)

The 3D coordinates of the position marker in millimetres in camera frame are displayed on the upper-left corner of *laptrack* as shown in Figure 2.19. The 3D coordinates of the position marker in millimetres in the world frame are displayed on the bottom-right corner of *demol* GUI. The 10 mm difference in the z-coordinate is the offset between the world frame and the camera frame because the origin of the camera frame, located at the laparoscope tip, is 10 mm below the origin of the world frame as shown in...
Figure 2.20. The origin of the world frame is 25 mm below the laparoscope-holding platform. Fifty different coordinates are tested, and all of them are successfully and correctly received by demo1 from laptrack through the network connection. Consequently, it is concluded that the TCP/IP connection does not introduce any significant errors in transmitting the coordinates of the position marker on the grasper.

Figure 2.19: 3D coordinates of the position marker on the surgical grasper shown on laptrack.

Figure 2.20: The relationship between world frame, camera frame, and the laparoscope-holding platform.
2.4.2 Robot Control Module

The motor control module, shown in Figure 2.21, consists of a PMD (Precision MicroDynamics) MC400 motion control board inside the PC, two PMD Breakout60 boards enclosed in the metal box at left, one Maxon ADS 50/5 servo amplifier, and three Advanced Motion Controls 12A8 servo amplifiers enclosed in the metal box at right. The two switches on the box switch on or off two of the four amplifiers each. Figure 2.22 shows the control hierarchy of the motor control module. The MC4000 motion control card acts as the interface between the computer and control boards. It receives commands from the motor control software demo1, and sends signals to the two Breakout60 boards, which drives two servo amplifiers each, to control the four motors. All motor encoder feedbacks are fed back to the Breakout60 boards, and then back to the computer through the MC4000 motion control card.

Figure 2.21: Motor control module.
To verify that the motor control module works correctly, there are two things to check. First, does every motor rotate the angle it is commanded? Gear ratio also plays a role in this. Second, is the direction of direction correct? The motors have to turn in the opposite direction to the desired shaft rotation direction as shown in Figure 2.23.

The motor rotation direction can be easily verified. For the rotation angle, only visual checks can be done since precision angle-measuring instruments are not available. Each motor is commanded to turn from $0^\circ$ to $360^\circ$ with $10^\circ$ increments. Needles are attached to the motor gears, and visual checks are done by placing a protractor on top on
the motor gears as shown in Figure 2.24. It is confirmed that the motors do turn the angles they are instructed to, which implies that the gear ratios are correct. This proves that although the motor control module could contribute to the laparoscope positioning error, it is not one of the major contributors.

Figure 2.24: The motors and gear sets.

2.4.3 Robot Control Program

To check if the robot control program works correctly, especially the inverse kinematics calculations, a total of eight test points in world frame, two in each of the four quadrants, is entered to demo1. To simplify the testing, the 50 mm clearance between the
laparoscope and the surgical grasper is omitted so that the tip of the laparoscope would reach right where the specified testing point is. Ideally, we should manually move the robot to position the laparoscope to reach the test points, and measure the actuating angles of each link. Then the angles calculated by demo1 can be compared with the measured angles to verify if there are errors in the inverse kinematics calculations.

However, since precision measuring instruments are not available, and the laparoscope-holding robot is also one of the possible sources of errors, this method would not work. Instead, the four motor angles, $\theta_1$, $\theta_2$, $\theta_3$, and $\theta_4$, calculated by demo1 for each test point are compared with Matlab calculations in Table 2.1.

<table>
<thead>
<tr>
<th>Test Points (mm)</th>
<th>demo1 Calculation Results (°)</th>
<th>Matlab Calculation Results (°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(-8,-2,30)$</td>
<td>598.47 -18.15 58.73 -16.80</td>
<td>598.47 -18.14 58.73 -16.81</td>
</tr>
<tr>
<td>$(-10,-5,50)$</td>
<td>1168.9 -11.89 54.79 -18.56</td>
<td>1168.9 -11.89 54.79 -18.55</td>
</tr>
<tr>
<td>$(10,10,55)$</td>
<td>1326.3 14.90 -43.82 27.81</td>
<td>1326.3 14.89 -43.82 27.81</td>
</tr>
<tr>
<td>$(10,-5,50)$</td>
<td>1168.9 25.75 -1.26 41.22</td>
<td>1168.9 25.74 -1.26 41.21</td>
</tr>
<tr>
<td>$(8,-2,40)$</td>
<td>874.24 24.41 8.85 11.67</td>
<td>874.24 24.41 8.85 11.67</td>
</tr>
<tr>
<td>$(-15,15,55)$</td>
<td>1387.5 -39.38 -43.72 6.44</td>
<td>1387.5 -39.38 -43.71 6.43</td>
</tr>
<tr>
<td>$(-20,10,40)$</td>
<td>1015.5 -50.17 -40.80 13.07</td>
<td>1015.5 -50.16 -40.80 -13.06</td>
</tr>
</tbody>
</table>

The Matlab code attached in Appendix A was independently developed by Temei Li with the same inverse kinematics method to calculate the actuating motor angles. All angles calculated by demo1 and Matlab are within 0.01 degree of each other.
$\theta_1$ is the turning angle for the small up-and-down motor. $\theta_2$ is the turning angle for the link 1 driving motor. $\theta_3$ is the turning angle for the link 2 driving motor. $\theta_4$ is the turning angle for the link 3 driving motor. Figure 2.25, Figure 2.26, Figure 2.27, and Figure 2.28 show the comparison of the angles $\theta_1$, $\theta_2$, $\theta_3$, and $\theta_4$ calculated by demo1 and Matlab for the eight test points. The points in the graphs are almost all right on top of each other. To conclude, the motor control program demo1 is not a major contributor to the laparoscope positioning errors.

Figure 2.25: Comparison of $\theta_1$ calculated by demo1 and Matlab.
Figure 2.26: Comparison of 02 calculated by demo1 and Matlab.

Figure 2.27: Comparison of 03 calculated by demo1 and Matlab.
2.4.4 The Laparoscope-Holding Robot

The laparoscope-holding robot is tested with the same eight test points from the last section. The position of the tip of the laparoscope is measured and recorded in Table 2.2 after the robot moves the laparoscope from home position to the destination test point. Figure 2.29 to Figure 2.36 show the eight measured points are different from the original test points. The measured points are not even in the same quadrant as the test points. Since the motor angles calculated by the motor control program demo1 agree with the Matlab calculations, and the motor control module rotate the motors without significant errors, these results imply that the laparoscope-holding robot is the main source of laparoscope positioning errors.
Table 2.2: Test and measured points comparison for the laparoscope-holding robot.

<table>
<thead>
<tr>
<th>Test Points (mm)</th>
<th>Measured Points (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (-8, -2, 30)</td>
<td>(3, -4, 31)</td>
</tr>
<tr>
<td>2 (-10, -5, 50)</td>
<td>(12, 1, 50)</td>
</tr>
<tr>
<td>3 (10, 10, 55)</td>
<td>(-1, -8, 56)</td>
</tr>
<tr>
<td>4 (5, 8, 35)</td>
<td>(-2, -8, 36)</td>
</tr>
<tr>
<td>5 (10, -5, 50)</td>
<td>(9, 3, 52)</td>
</tr>
<tr>
<td>6 (8, -2, 40)</td>
<td>(-1, -1, 40)</td>
</tr>
<tr>
<td>7 (-15, 15, 55)</td>
<td>(9, -9, 57)</td>
</tr>
<tr>
<td>8 (-20, 10, 40)</td>
<td>(8, -1, 40)</td>
</tr>
</tbody>
</table>

Figure 2.29: Test and measured point 1 comparison for the laparoscope-holding robot.
Figure 2.30: Test and measured point 2 comparison for the laparoscope-holding robot.

Figure 2.31: Test and measured point 3 comparison for the laparoscope-holding robot.
Figure 2.32: Test and measured point 4 comparison for the laparoscope-holding robot.

Figure 2.33: Test and measured point 5 comparison for the laparoscope-holding robot.
Figure 2.34: Test and measured point 6 comparison for the laparoscope-holding robot.

Figure 2.35: Test and measured point 7 comparison for the laparoscope-holding robot.
Test and Measured Point 8 Comparison

Figure 2.36: Test and measured point 8 comparison for the laparoscope-holding robot.

The robot has been the main suspect of source of errors ever since it was built. First, it was not built according to the design specifications. The most significant differences are the arc angles of the six link pieces that make up the three branches holding and rotating the laparoscope-holding platform as shown in Figure 2.37. Each branch is made up of two arced link pieces, and their arc angles play key roles in the inverse kinematics calculations. Designer Li [6] [7] [8] [9] measured their arc angles and updated the specifications, but the measurements may not be accurate enough due to the lack of precision instruments. In addition, the angle between the driving shaft and platform is off, which causes the results of theoretical inverse kinematics calculations to be further off from the actual values.
Figure 2.37: The driving shaft, platform, middle joint, platform joint, and the six link pieces of the three branches.

Second, shown in Figure 2.37, the middle joints connecting the link pieces and the platform joints connecting the platform and the branches are loose. They could even come off occasionally. When all the gears are engaged, the platform is not supposed to move. However, as we discovered, the platform could be rotated to a certain degree manually when the gears are engaged due to the loose middle and platform joints. Moreover, before the system starts up, the robot has to be manually set to its designed home position. Since the robot is set by hand, and the home position is manually set without precision measurements, errors are unavoidable. As a result, before the robot starts moving, its actual home position is already off from the designed home position programmed in demo1. Another source of error comes from the payload. When the laparoscope is fixed to the platform, the weight of the camera and the length of the laparoscope combine to generate a torque that tilts the platform to one side due to the loose middle and platform joints. As a result, the robot is further away from the designed home position.
Third, when the robot is moving, the joints do not rotate smoothly. Adding to the complexities, the link pieces would collide with each other when they overlap. Branch 2 shown in Figure 2.38 is a prime example of this situation. The clearance between the upper and lower link pieces is quite tiny, causing them to frequently rub against each other. In addition, Figure 2.38 shows the space limitation for branch 2 to rotate before reaching singularity. Thus, sometimes the lower link would either be stuck with the end of driving shaft, which stalls the robot, or flip over, which causes singularity. To solve this problem, branch 2 will have to be redesigned and repositioned.

Fourth, when the robot stops, the platform may not immediately stop moving. The torque generated by the camera, along with the loose middle and platform joints, would often tilt the platform more, creating more errors. Every time the robot moves, the errors accumulate. After several moves, the robot would be completely out of position. In other words, the actual position of the robot would be far off from the position calculated. Consequently, when the robot is stopped and homed, it often goes back to a
different home position than where it started. If the robot were to move again, the positioning would be even further off.

A look-up table method was suggested to replace the inverse kinematics robot control approach. In our plan, the ranges for all three motors are defined to be -50° to 50°. A look-up table is built by rotating one of the three motors by increments of 10° while the other two motors remain still and recording the corresponding coordinates. After the look-up table is completed, demo1 would search the table to find the point that is closest to the destination point and retrieve the three corresponding motor angles.

However, this plan failed because of two reasons. First, in order to build the look-up table, the robot has to be homed many times so that each motor can take its turn to rotate. This presents a major problem since every time the robot is homed, it goes back to a different home position due to the reasons described above. Consequently, a reliable look-up table cannot be achieved due to the unreliable reference home position. Second, we observed that when one of the three motors is commanded to turn 10°, 20°, 30°, and 40° while the other two remain still, the robot always go to the same point. That means either some of the loose middle and platform joints buffer up the small changes in angle, or some of the link pieces are jammed, causing the robot to stall at certain orientations. Therefore, a meaningful look-up table cannot be built, and the look-up table method has been abandoned. It is also concluded that the robot has too many structural defects that it must be the major contributor to the laparoscope positioning errors.
CHAPTER 3

AUTOMATIC SURGICAL TOOL TRACKING IMPROVEMENTS

The laptrack automatic surgical tool tracking program, originally developed by Zhang [32] [33], tracks a black position marker on the tip of a stick with a pure white background as shown in Figure 3.1. The x and y coordinates of the position marker are determined in pixels by blob analysis using the Matrox Imaging Library (MIL). The z coordinate, which is the distance between the laparoscope and the surgical grasper, is estimated in millimetres. Three major improvements have been made since then, including an experimental setup change, a coordinate unit conversion, and an obstacle avoidance function. Section 3.1 describes the experimental setup change that better mimics the real surgical environment for system testing. Section 3.2 introduces the coordinate unit conversion that converts the x and y coordinates of the position marker from pixel to millimetres to retain unit consistency with the rest of the system. Section 3.3 describes the obstacle avoidance function that avoids potential collisions between the laparoscope and obstacle surgical tool that may be in its path.
3.1 Experimental Setup Change

Shown in Figure 3.1, the original experimental setup designed by Zhang [32] [33] for surgical tool tracking includes a white stick about 10 mm in diameter with a black marker near the end, and the background was purely white. The stick is larger in diameter than a real surgical tool so that the marker on it has bigger area and is easier to track. Obviously, this setup does not resemble the real laparoscopic surgical environment shown in Figure 3.2 at all. Therefore, a new experimental setup, which consists of a plastic stomach model and real surgical tools as shown in Figure 3.3, is used. Inside the plastic stomach model, the rubber organs and plastic tissue walls as shown in Figure 3.4 better mimic the background of real surgical site in Figure 3.2.

Figure 3.2: Real laparoscopic surgery environment.
Figure 3.3: The new experimental setup with plastic stomach model and real surgical tools.

Figure 3.4: Inside of the stomach model with real surgical tools.
The rest of the experimental setup, including the laparoscope, light source, and camera unit, remains the same. Due to the large amount of laparoscope-positioning errors, the laparoscope-positioning robot is ignored. Instead, the laparoscope is inserted into the plastic stomach model and held in place by a stand as shown in Figure 3.3. Images are captured by the Karl Storz *SuperCam* 9050B CCD colour camera. The CCD sensors inside the camera are electronic systems which transform the real image (photons) into electronic images which may be read on a screen. The *SuperCam* is a small, light weight, easy to operate, high resolution camera for video monitoring and recording of laparoscopic procedures.

Illumination is provided by the Karl Storz 615 xenon light source, which is the box on the top in Figure 3.5, through a fibre optic cable. The xenon lamp colour temperature approximates bright sunlight and is considered unmatched for visual and photographic colour rendition. The number “35” displayed on the light source indicates the brightness. Brightness can be adjusted by the switch on the control panel.

![Figure 3.5: The light source and camera unit.](image)
The captured images are transferred to the camera unit, which is the box at the bottom in Figure 3.5. The camera unit can only handle greyscale images, which is why our system only works with greyscale images. The camera unit then feeds the images to the Matrox Corona frame grabber in the image processing computer.

The Matrox Corona is a single-slot frame grabber built for colour imaging applications. It handles bandwidth transfer of up to 24-bit colour image over the PCI bus and, at the same time, displays 24-bit true-colour live video with true-colour non-destructive overlay (no influence on the original image). Its RGB digitizer’s sampling rate is up to 30MHz. Matrox Corona’s integrated display section delivers real-time display of captured video. Various resolutions up to 1600×1200 are supported at refresh rates up to 85 Hz. With dual frame buffer architecture, Matrox Corona also provides non-destructive overlay. Matrox Corona can deliver up to 24-bit colour image display with up to 24-bit colour overlay. However, because of the greyscale camera unit, the captured camera image is in greyscale.

The laptrack image processing program uses the Matrox Imaging Library (MIL) to process the camera image. MIL is a hardware independent, modular 32-bit imaging library that includes ActiveMIL, a collection for managing image capture, transfer, processing analysis and display. In general, MIL can manipulate binary, greyscale, or colour images. It enables fast application development use.

Instead of tracking a stick that is much bigger than a real surgical tool, real surgical tools with white markers are used in the new setup. Figure 3.4 shows the tool tips being marked white for surgical tool gesture and motion recognition. Since the camera system only outputs greyscale images, white markers are easier to segment from the
darker background. The white stripe marker on the surgical grasper on the right is the position marker, which is the marker laptrack tracks. The position marker design follows the guidelines set by Zhang [32] [33], but its size is smaller than the original marker due to the smaller diameter of the surgical grasper. As illustrated in Figure 3.6, \( M \) is the diameter of the surgical grasper cross section, and \( d \) is the width of the marker. The width \( d \) is 2 mm to satisfy the condition \( \frac{M}{3} < d < \frac{M}{2} \), where \( M \) is 5 mm.

![Figure 3.6: The position marker design.](image)

Since both the background and the position marker have been changed, the tracking algorithm in laptrack needs to be modified as well. The new experimental setup shown in Figure 3.4 is white-marker-on-dark-background, which is the opposite of the old black-marker-on-white-background setup shown in Figure 3.1. To extract the position marker from the background of the original experimental setup, the older tracking algorithm binarizes the camera image, and only pixels dark enough are recognized as part of the position marker blob. The 3D coordinates of the position tool blob are then estimated.

With the new setup, the blob extraction process is still the same. However, only pixels bright enough are recognized as part of blobs since now all markers are white instead of black. There could be two blobs if only the grasper is used to perform single-tool gestures, and three blobs if a pair of scissors is also used to perform double-tool gestures as shown in Figure 3.4. The position marker can be distinguished from the tool
tip blobs by its smaller size. Once the blobs are categorized, the tool tip blobs are used for tool gesture recognition, and the position marker blob is used for coordinate estimation and tool motion recognition.

3.2 X and Y Coordinate Unit Conversion

The original laptrack developed by Zhang [32] [33] estimates the z coordinate of the position marker in millimetres, but the x and y coordinates are estimated in pixels. Please refer to Appendix B for details about the coordinate estimation and camera calibration. The z coordinate is the distance between the laparoscope and the surgical grasper as shown in Figure 3.8. When laptrack shows the position marker at (150, 78, 50), it actually means the position marker is 150 pixels in the positive x direction and 78 pixels in the positive y direction from the origin of the screen frame, which is located at the top left corner of the window, and 50 mm below the laparoscope. Thus, a conversion factor is added to laptrack to convert the x and y coordinates from pixels to millimetres to match the units. The laparoscope-holding robot, which takes 3D coordinates in millimetres as inputs, can therefore position the laparoscope properly.

![Figure 3.7: The z coordinate is the distance between the laparoscope and the surgical grasper.](image)

Laparoscope

z coordinate

Surgical grasper
In order for the coordinate units to match, the unit has to be converted from screen pixel to physical millimetre with a conversion factor. The conversion factor represents the number of millimetres per pixel, and it depends on the z coordinate of the surgical grasper. For example, when the grasper is close to the laparoscope, it takes a large number of pixels to cover the position marker. In comparison, the position marker could be covered by fewer pixels when the grasper is further away from the laparoscope. In other words, the number of pixels representing one millimetre varies with the z coordinate of the surgical grasper. As a result, the z coordinate of the position marker needs to be determined first in order to choose the right conversion factor to convert the x and y coordinates from pixel to millimetre. The coordinate unit conversion process is summarized in Figure 3.8.

**Figure 3.8: Position marker x and y coordinate unit conversion flow chart.**

### 3.2.1 The Pixel-to-Millimetre Conversion Factor

To determine the pixel-to-millimetre conversion factor at various distances, a marker with a pair of black dots is placed under the laparoscopic camera at eighteen different distances. Figure 3.9 shows the experimental setup. The experiment is done outside the plastic stomach model so that the actual distance can be measured by a ruler. The centres of the two black dots on the marker are separated by 10 mm. At each of the eighteen different distances, the 10 mm separation between the black dot centres
displayed on screen would be different. The number of pixels representing the 10 mm separation is recorded for each distance.

Figure 3.9: Setup for determining the pixel-to-millimetre conversion factor at various distances.

Figure 3.10 shows the pinhole camera model of the experimental setup. The focal length, $f$, is the distance between the image plane and the camera. Assuming the marker with the pair of dots is parallel to the image plane, the further away the marker is to the camera, the smaller the separation appears on the screen, and vice versa. This is illustrated by separation$_{\text{current}}$ and separation$_z$, where separation$_{\text{current}}$ is the separation which appears on the screen when the marker is placed at $Z_{\text{current}}$, and separation$_z$ is the separation which appears on the screen when the marker is placed at $z$. When the marker is placed at $z = 41$ mm, separation$_z$ appears to be 100 pixels on the screen. Consequently, the conversion factor is 1 mm : 10 pixels. When the marker is placed at $Z_{\text{current}} = 100$ mm, the separation$_{\text{current}}$ appears to be 35 pixels on the screen. The resulting conversion factor is 1 mm : 3.5 pixels, which is much larger than that at $z = 41$ mm.
The numbers of pixels appearing on screen that represent the 10 mm actual separation at different distances between 30 mm to 200 mm are listed in Table 3.1. The table is only established between distances of 30 mm to 200 mm since the maximum vertical clearance inside the plastic stomach model is less than 200 mm. In theory, the number of pixels halves as distance doubles and the collected data reflects that relationship. The overall inverse-proportional relationship between separation and distance from the collected data is shown in Figure 3.11.

<table>
<thead>
<tr>
<th>Distance (mm)</th>
<th>10 mm Separation on Screen (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>125</td>
</tr>
<tr>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>80</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>70</td>
<td>50</td>
</tr>
<tr>
<td>80</td>
<td>44</td>
</tr>
<tr>
<td>90</td>
<td>38</td>
</tr>
<tr>
<td>100</td>
<td>35</td>
</tr>
</tbody>
</table>
Figure 3.11: Number of pixels appearing on screen representing 10 mm actual separation vs. distance.
3.2.2 The Pixel-to-Millimetre Unit Conversion Process

To convert the estimated x and y coordinates from pixel to millimetre, the appropriate pixel-to-millimetre conversion factor is retrieved from Table 3.1 according to the estimated z coordinate. The $x_{\text{pixel}}$ and $y_{\text{pixel}}$ coordinates, estimated by blob analysis in pixels, are converted to $x_{\text{new}}$ and $y_{\text{new}}$ in millimetres as

$$x_{\text{new}} = (x_{\text{pixel}} - x_{\text{offset}}) \times \text{conversion factor}$$  \hspace{1cm} (3.1)

$$y_{\text{new}} = (y_{\text{pixel}} - y_{\text{offset}}) \times \text{conversion factor}$$  \hspace{1cm} (3.2)

where $x_{\text{offset}}$ and $y_{\text{offset}}$ are offsets between the screen frame and camera frame in pixels. The origin of the screen frame is on the top-left-hand corner of the image window, while the origin of the camera frame is directly below the camera at the centre of the image; hence the offsets. In the case that an estimated z coordinate does not match one of the distances in the table, linear approximation is used. For example, if the estimated z coordinate is 74 mm, which is between 70 mm and 80 mm, then the approximate screen separation is:

Approx. screen separation = $50 + (40 - 50) \times \left(\frac{74 - 70}{80 - 70}\right) = 50 + (-10) \times 0.4 = 46$ pixels

Consequently, the conversion factor is 1 mm : 4.6 pixels.

3.2.3 Results and Analysis

Due to difficulties in measuring the actual coordinates inside the plastic stomach model, the unit conversion testing is done with the laparoscope outside the plastic stomach model with 10 mm x 10 mm grids on a piece of black paper as the background.
The surgical grasper is placed on the background paper such that the position marker on it sits directly on top of one of the grid intersections as shown in Figure 3.12.

![Figure 3.12: Position marker x and y coordinate estimation and unit conversion experimental setup.](image)

Although the conversion factor table works for surgical tool distances from 30 mm all the way up to 200 mm, the testing range is chosen to be between 30 mm and 70 mm. With the light source brightness set at 35 lux, the light becomes too dim at distances beyond 70 mm, and the position marker is not bright enough to be recognized. Lens distortion is ignored in all testing in this project for simplicity. Please refer to Appendix B for details on lens distortion correction.

First of all, the accuracy of the z coordinate estimation from Zhang [32] [33] is tested with the new position marker on the grasper. The grasper is placed from 30 mm to 70 mm below the laparoscope. The results in Figure 3.13 show that the z coordinate estimation is very accurate. The smallest error is 2.9% at 35 mm, and the largest error is 7.1% at 70 mm. The average error is only 4.5%, which is about a couple of millimetres.
Surgical Tool Z Coordinate Estimation Errors

Figure 3.13: The laptrack surgical tool tracking Z coordinate estimation errors.

The original laptrack developed by Zhang [32] [33] shows only the estimated x and y coordinates of the centre of the position marker estimated by blob analysis in pixels. A pixel-to-millimetre conversion factor is added to convert the estimated x and y coordinates from pixels to millimetres so that the unit matches with the unit of the estimated z coordinate. The accuracy of the x and y coordinate estimation and the pixel-to-millimetre unit conversion are examined next.

To account for the smaller field of view at lesser distances, the test points for the x and y coordinate estimations at distances 30 mm and 40 mm are chosen to be (10 mm, 10 mm), (-10 mm, 10 mm), (10 mm, -10 mm), and (-10 mm, -10 mm). Figure 3.14 shows one of the tests done at the distance of 30 mm with illumination brightness set at 35 lux. The position marker on the grasper is placed at (-10 mm, 10 mm). The converted x and y coordinates, which are the first two numbers of Tool Centre shown in Figure 3.14, are recorded. The third number, 31, is the estimated distance of the position marker. The
black crosses mark the centre of the position marker and tool tip blobs. The grey crosses mark the four blob boundary points \((x_{\text{min}} , y_{\text{max} @ x_{\text{min}}} , x_{\text{max} @ y_{\text{max}}} , y_{\text{max}})\), \((x_{\text{min} @ y_{\text{min}}} , y_{\text{min}})\), and \((x_{\text{max}} , y_{\text{min} @ x_{\text{max}}})\).

Figure 3.14: Test done at the distance of 30 mm with the position marker placed at \((-10, 10)\).

Test points \((0 \text{ mm}, 20 \text{ mm})\), \((0 \text{ mm}, -20 \text{ mm})\), \((20 \text{ mm}, 0 \text{ mm})\), and \((-20 \text{ mm}, 0 \text{ mm})\) are added for distances 50 mm to 70 mm due to the larger field of view. Figure 3.15 shows the wider field of view at a distance of 70 mm with the position marker at \((0 \text{ mm}, 0 \text{ mm})\). The complete x and y coordinates estimation testing results are listed in Table 3.2. The first row shows the test points, and the first column shows the different distances. The grasper is placed on the grid such that the centre of the position marker sits right on top of the test points.
The tests show the overall average x and y coordinate estimation and unit conversion errors are 19.3% and 15.2% for x and y, respectively. The average errors in x and y at each distance are shown in Figure 3.16. For example, the first two bars from the
left show that, for x and y at 30 mm, the average coordinate estimation and unit conversion errors are 11.1% and 14.1%. Recall in Figure 3.8, the choice of conversion factor depends on the estimated z coordinate; therefore, the x and y coordinate estimation and unit conversion errors contain inherited errors from the z coordinate estimation.

Table 3.2 shows that the converted x and y coordinates are only off by one or two millimetres. Given the diameter of the field of view is about 40 mm at the distance of 30 mm, and even larger at greater distances, the one or two millimetre error in either x or y coordinates does not drive the robot off target. In other words, the position marker would still be around the centre of the field of view, as well as the grasper tool tip. Therefore, the coordinate estimation and unit conversion have succeeded.
3.3 Obstacle Avoidance

Imagine a scenario where the surgeon is using one of the surgical tools to hold on to a certain part of the tissue, and he or she wants to use the other tool the robot is tracking to guide the laparoscope to examine the surrounding area. During the visual examination, the moving surgical tool, which is tracked by the robot, may go under the stationary one, causing the laparoscope tracking above to collide with the stationary tool as shown in Figure 3.17.

![Figure 3.17: Possible collision between laparoscope and stationary surgical tool while tracking the moving tool.](image)

If the laparoscope were held by a human assistant, the potential laparoscope-tool collision could be easily avoided. However, with the laparoscope-holding robot, the system has to be able to detect such potential collisions and plan alternative routes. With a full robotic system in which both the tools and the laparoscope are held by robots, a collision-free route can be determined by the system as all the tool positions are known.
through the robots. However, in our current image-based system, in which only the laparoscope is held by the robot, the system has to estimate the position of the stationary tool visually. In other words, the stationary tool, which will be referred to as the obstacle, has to be within the field of view of the camera. Otherwise, the system cannot detect if the obstacle is in the laparoscope's moving path. As a result, collisions cannot be avoided.

Assuming the system has the obstacle in sight, there are two ways for the laparoscope-holding robot to avoid collisions. It can direct the laparoscope to go either around the obstacle without vertical movements as shown in Figure 3.18(a), or above and over the obstacle as shown in Figure 3.18(b). For the first method to work, the obstacle tool cannot be attached to any tissue; otherwise, there is no way around the obstacle. In addition, the tip of the obstacle tool must be in the field of view of the camera for the system to plan a route around it. On the contrary, the second method does not require the tip of the obstacle tool to be in the view, and the obstacle tool could be attached to tissue. All the system needs to see is the part of the obstacle tool in the laparoscope's moving path. Thus, an obstacle avoidance function based on this method is added to laptrack.

Figure 3.18: Two possible paths for the laparoscope to go around an obstacle surgical tool.
3.3.1 Obstacle Tool Marker Design

For the system to recognize the obstacle tool, markers are placed on the trunk of the tool. The markers are wrapped around the trunk so that they can be seen by the camera regardless of the roll angle. The row, pitch, and yaw angles of a surgical tool are defined in Figure 3.19. The incision point is also the pivot point. Rotation around the vertical axis is called the yaw. Rotation around the side-to-side axis is called the pitch. Rotation around the front-to-back axis is called the roll. Positive rotation is defined as counter-clockwise rotation looking into the pivot point from the larger arrow on each axis.

![Figure 3.19: The roll, pitch and yaw angles of a surgical tool.](image)

To avoid misidentification, the lengths of the obstacle markers are significantly larger than that of position marker and tool tip. The lengths of the white obstacle markers are 30 mm, with 20 mm gaps between them as shown in Figure 3.20. In comparison, the length of the position marker is 2 mm, and the length of the tool tip is about 15 mm, which are both smaller than the obstacle markers. There are multiple markers placed on the trunk of the obstacle tool so that at least one of them is in the field of view and can be...
recognized by the system independent of how deep or shallow the obstacle tool goes into the abdominal.

![Image of obstacle tool markers](image)

Figure 3.20: The 30 mm long obstacle tool markers with 20 mm gaps in between.

The obstacle marker blobs are separated from the position marker blob and tool tip blobs by its larger size. As shown in Figure 3.21, the smallest white marker on the right is the position marker, and the largest white marker on the left is the obstacle tool marker. The medium size white marker in the middle is the grasper tool tip.

![Image of obstacle marker](image)

Figure 3.21: Obstacle marker on the left having larger size than tool tip and position marker.
3.3.2 The Obstacle Tool Model

The obstacle tool model, based on the obstacle tool marker shown in Figure 3.22a), is a quadrilateral as shown in Figure 3.22b). The length of the obstacle is not a concern in the obstacle avoidance algorithm because the algorithm directs the laparoscope to go above and across the long side of the obstacle as shown in Figure 3.18b). The algorithm works for obstacle tools of any length. Furthermore, the width of the obstacle does not need to be explicitly determined. When the grasper with the position marker moves under and across the obstacle tool, the last set of position marker coordinates captured before it goes under the obstacle and the first set of position marker coordinates captured after it has crossed the obstacle already contain the width information about the obstacle. The thickness of the obstacle tool model is the diameter of the trunk of the obstacle tool, which is 5 mm.

![Image](image.png)

Figure 3.22: The obstacle tool marker and quadrilateral obstacle tool model.

The most important attribute of the obstacle tool model is the estimated z coordinate, as it plays a key role in the obstacle avoidance function presented in the next
section. The z coordinate of the obstacle is estimated by the same method used to estimate the z coordinate of the position marker on the grasper by Zhang [32], [33]. This estimated z coordinate represents the average z coordinate of the obstacle tool model.

3.3.3 The Obstacle Avoidance Algorithm

The obstacle avoidance algorithm is as follows. While the grasper with position marker moves, the program checks if there is an obstacle in the laparoscope’s direct moving path between every set of old and new position marker coordinates. The old and new sets of position marker coordinates are referred to as the start and goal points. When the grasper moves across the obstacle as shown in Figure 3.23, the collision check compares the z coordinates of the start point (start_z), the goal point (goal_z), and the obstacle (obstacle_z). If both of the start and goal points are not more than 20 mm lower than the obstacle, the 50 mm clearance between the laparoscope and the position marker would allow the laparoscope to reach the goal point without any potential collisions. The 50 mm clearance, as described in Section 2.3.2, ensures that the camera’s field of view is large enough to accommodate the surgical tools and some of the background tissues.

Figure 3.23: No potential collision if both the start and goal points are not more than 20 mm lower than the obstacle.
The 20 mm vertical threshold, which is the maximum distance \( \text{start}_z \) and \( \text{goal}_z \) can be lower than \( \text{obstacle}_z \), is chosen based on the worst-case scenario illustrated in Figure 3.24. In the worst-case scenario, the start point is at the bottom of the plastic stomach model, which is about 200 mm from the incision hole, and the obstacle is 20 mm higher than the start point. Assuming the laparoscope is at the maximum 20° angle, the length of the part of the laparoscope inside the plastic stomach model would be 163 mm with a 50 mm clearance between the tip of the laparoscope and the start point. As the laparoscope rotates about the incision hole, it would have a 17 mm clearance over the obstacle. Taking into account the 5 mm thickness of the obstacle tool, there would still be a clearance of more than 10 mm between the laparoscope and the obstacle. This is the reason why the vertical threshold is chosen to be 20 mm.

Figure 3.24: The worst case scenario of the laparoscope moving across the obstacle tool.
On the other hand, if either or both of start\(_z\) or goal\(_z\) are more than 20 mm lower than obstacle\(_z\), a collision between the laparoscope and the obstacle is likely to happen. In such case, one to two extra path points are generated to guide the laparoscope to go above and over the obstacle. If the start point is more than 20 mm lower than the obstacle, a path point is generated with the same x and y coordinates as the start point. The z coordinate of the path point is set to 20 mm lower than the obstacle so that the laparoscope, which would be 50 mm above the path point, could clear the obstacle with minimal vertical movement. If the goal point is more than 20 mm lower than the obstacle, a second path point is generated with the same x and y coordinates as the goal point. The z coordinate of the second path point is also set to 20 mm lower than the obstacle. Finally, the path point(s) and the goal point are sent to the robot control program in sequence as if the position marker actually goes through the path points to reach the goal point.

An example of the goal position being more than 20 mm lower than the obstacle is shown in Figure 3.25(a). In this case, a path point is generated with the same x and y coordinates, \(g_x\) and \(g_y\), as the goal point. The z coordinate of the path point is set to 80 mm, which is 20 mm lower than the obstacle, to avoid laparoscope-obstacle collisions. Figure 3.25(b) shows an example of both the start and goal points being more than 20 mm lower than the obstacle. Two path points are generated in this example. The first path point, path point 1, has the same x and y coordinates, \(s_x\) and \(s_y\), as the start point, and the z coordinate is set to 80 mm, which is 20 mm lower than the obstacle. Similarly, the second path point, path point 2, has the same x and y coordinates, \(g_x\) and \(g_y\), as the goal point, and the z coordinate is set to 80 mm as well. The path points would guide the laparoscope-holding robot to move the laparoscope above and over the obstacle to avoid...
potential collisions. Algorithm 3.1 shows the pseudo code for the obstacle avoidance algorithm.

Algorithm 3.1: The obstacle avoidance algorithm.

START

IF the grasper moves across the obstacle THEN

    IF (start_z < (obstacle_z + 20 mm)) and (goal_z < (obstacle_z + 20 mm)) THEN
        No path planning required
    ELSE IF (start_z > (obstacle_z + 20 mm)) THEN
        Generate one path point with x = start_x, y = start_y, and z = obstacle_z + 20 mm
    ELSE IF (goal_z > (obstacle_z + 20 mm)) THEN
        Generate one path point with x = goal_x, y = goal_y, and z = obstacle_z + 20 mm
    END IF

END IF

IF path point(s) are generated THEN

    Send the path point(s) in sequence to the robot control program

END IF

END IF

Send the goal point to the robot control program

END
3.3.4 Results and Analysis

Although the optimal vertical threshold is determined to be 20 mm, it is found that the obstacle would block too much light such that the position marker on the grasper would be too dark to be recognizable if it were placed 20 mm below the obstacle. Turning up the brightness does not help because most of the light would still be blocked by the obstacle tool. Therefore, in the obstacle avoidance function testing, the vertical threshold is set to 5 mm so that the position marker is bright enough to be recognized by laprack.

There are three testing scenarios. The first one is that the start and goal points are both lower than the obstacle tool by more than 5 mm. As an example, Figure 3.26 shows the position marker on the grasper at a start position of 73 mm, which is more than 5 mm lower than the obstacle at 62 mm. Figure 3.27 shows the position marker at the goal position of 73 mm deep, which is also more than 5 mm lower than the obstacle at 64 mm.

![Figure 3.26: Scenario 1 - Start point lower than the obstacle tool.](image)
In this scenario, two path points are created to avoid potential collisions with the obstacle as shown in the laparoscope tracking simulation in Figure 3.28. The virtual laparoscope-holding robot holds and moves the virtual laparoscope to track the position marker on the grasper, which is represented by a dot. Inside the virtual abdominal, the quadrilateral represents the obstacle tool. The dot at the bottom-left is the start point, which is the last point the camera sees before the position marker on the grasper disappears underneath the obstacle tool. The dot at the bottom-right is the goal point, which is the first point where the position marker on the grasper appears again on the other side of the obstacle tool. Above the start point on the left is path point 1, which has the same x and y coordinates as the start point; its z coordinate is obstacle z coordinate plus 5 mm. Thus, the coordinate of the path point 1 is (12, 5, 67). Similarly, path point 2, which is above the goal point at right, has the same x and y coordinates as the goal point, and its z coordinate is obstacle z coordinate plus 5 mm. Consequently, the coordinates of
path point 2 \((-18, 1, 69)\). From the start point, the robot would direct the laparoscope to point to path point 1, path point 2, and finally, the goal point to avoid potential collisions with the obstacle tool.

The second scenario is that the start point is higher than the obstacle, and the goal point is lower than the obstacle tool by more than 5 mm. As an example, Figure 3.29 shows the position marker on the grasper at the start position is at 42 mm, which is higher than the start point.
than the obstacle at 66 mm. Figure 3.30 shows the position marker at the goal position of 78 mm, which is more than 5 mm lower than the obstacle at 63 mm.

Figure 3.29: Scenario 2 - start point higher than the obstacle tool.

Figure 3.30: Scenario 2 - goal point lower than the obstacle tool.
In the second scenario, only one path point, which is above the goal point on the right, is generated to avoid potential collisions as illustrated in Figure 3.31. The path point has the same x and y coordinates as the goal point, and its z coordinate is obstacle z coordinate plus 5 mm. Consequently, the coordinates of the path point is (-19, 14, 68). From the start point, the robot would move the laparoscope around the obstacle by pointing it to the path point, then to the goal point.

Figure 3.31: Scenario 2 – one path point being generated above the goal point to avoid collisions.

The third scenario is that the start point is lower than the obstacle tool by more than 5 mm, and the goal point is higher than the obstacle. As an example, Figure 3.32
shows the position marker on the grasper at the start position of 69 mm, which is more than 5 mm lower than the obstacle at 59 mm. Figure 3.33 shows the position marker at the goal position of 44 mm, which is higher than the obstacle at 62 mm.

Figure 3.32: Scenario 3 - start point lower than the obstacle tool.

Figure 3.33: Scenario 3 - goal point higher than the obstacle tool.
In this scenario, only one path point, which is above the start point on the left, is generated to avoid collision as shown in Figure 3.34. The path point has the same x and y coordinates as the start point, and its z coordinate is obstacle z coordinate plus 5 mm. Consequently, the coordinates of the path point is (25, 4, 64). From the start point, the laparoscope would be directed by the robot to point to the path point, then to the goal point to avoid potential collisions with the obstacle tool.

Figure 3.34: Scenario 3 – one path point being generated above the start point to avoid collisions.

Each scenario is tested fifty times with the experimental setup described in Section 3.1, where the grasper with position marker is moved underneath and across the
obstacle tool from one side to the other each time. If the proper path point(s) is generated and shown on the tracking simulation, the test is a success. The testing results are shown in Figure 3.35. Scenario one has a success rate of 32%. Scenario two has a success rate of 24%. Scenario three has a success rate of 22%. The overall success rate is 26%.

![Obstacle Avoidance Function Testing Results](image_url)

Figure 3.35: Obstacle avoidance function testing results.

Scenario one has a slightly higher success rate than the other two scenarios because the grasper movement in the first scenario is purely horizontal. The grasper always stays below the obstacle tool. The mixed horizontal and vertical grasper movement involved in moving the grasper higher or lower than the obstacle tool in the second and third scenario is harder to manipulate. For example, sometimes the grasper would hit the obstacle tool and drag the obstacle tool with it, causing the test to fail.

Overall, the 26% success rate of the obstacle avoidance function means only one out of four potential collisions could be avoided, which is too low for actual surgical
application. The non-satisfactory performance is due to two main reasons: the lighting condition inside the abdominal, and the marker design.

Inside the abdominal, the whole space is illuminated by a single light source coming from the laparoscope at the top. The obstacle tool often blocks the light source. As a result, there is too little light reflected off the position marker on the grasper underneath for the system to recognize the position marker as illustrated in Figure 3.36. The lack of crosses around the position marker and the grasper tool tip in Figure 3.36 implies that the position marker and the grasper tool tip are not being recognized because their pixel intensities do not reach the threshold. Consequently, the system would lose track of the position marker and the grasper tool tip. Not only would the obstacle avoidance function fail, but the surgical instrument tracking would also malfunction. One possible solution to solve this problem is to upgrade a coloured camera system, and use different colours for different markers. Identification of the markers would then depend on colours rather than pixel intensities.

Figure 3.36: Not enough light reflected off the position marker and grasper tool tip to be recognized.
The second problem is marker design. Due to the nature of the greyscale camera system, white is the only choice of colour for the markers. As a result, different markers can only be distinguished by size and shape, both of which constantly vary with the movement of surgical tools. Since the position marker, closed grasper tool tip and obstacle marker are all quadrilateral, the only attribute that can be used for categorization is size. Currently, there can only be three markers in the field of view as shown in Figure 3.37, and the blob with the largest area is labelled as the obstacle marker, the blob with medium area is labelled as the grasper tool tip, and the blob with the smallest area is labelled as the position marker. The obstacle avoidance function and surgical instrument tracking function would then work according to the above blob labelling. If the blobs were labelled incorrectly, the x and y coordinate estimation, z coordinate estimation, and coordinate conversion would all be incorrect, causing both functions to fail.

Figure 3.37: From left to right, the obstacle marker, the grasper tool tip, and the position marker.

A number of scenarios can cause the above-mentioned confusions to the system. First, if the grasper were closer to the laparoscope than the obstacle tool, the grasper tip blob would be larger than the obstacle marker blob. Shown in Figure 3.38, the grasper tip blob would be larger than the obstacle marker blob.
is far closer to the laparoscope and blocking the light such that the part bright enough to be recognized on the obstacle marker (enclosed by the four grey crosses) is far smaller than the grasper tool tip blob. In this example, the area of the recognizable obstacle marker is even smaller than the area of the position marker. Consequently, the grasper tool tip would be labelled as the obstacle, the position marker would be labelled as the tool tip, and the obstacle would be labelled as the position marker. The obstacle avoidance function would malfunction as a result.

Figure 3.38: Grasper tool tip being closer to the laparoscope than the obstacle tool.

Three other scenarios would also create confusions in properly identifying the blobs. Figure 3.39 shows the scenario that the obstacle tool marker is broken into two by the grasper, creating a total of four blobs in the field of view. In this particular example, the grasper tool tip and the two blobs of the broken obstacle marker have similar areas. Depending on which one is the largest, the obstacle avoidance function could malfunction in many different ways. laptrack could label one of the broken obstacle marker blobs to be the grasper tool tip blob, and the real grasper tool tip blob as the obstacle. As a result, the position of the obstacle would be incorrect. laptrack could also
label one of the broken obstacle marker blobs as the obstacle. However, the estimated z coordinate of the obstacle would be incorrect because the z coordinate estimation depends on the blob size. The estimated z coordinate of the obstacle would be much lower than it actually is because the size of the broken obstacle marker blob is less than half of the actual size.

Figure 3.39: Obstacle tool marker being broken by the grasper.

When the grasper tool tip is broken into two by the obstacle tool as shown in Figure 3.40, \textit{laptrack} might mistakenly label one of the broken grasper tool tip blobs as the position marker if its area is smaller than the position marker’s. Consequently, the x, y, and z coordinates of the position marker would be misinterpreted, and the laparoscope would be tracking the wrong marker. \textit{laptrack} would falsely conclude that the position marker has not crossed the obstacle tool. The smaller size of the broken grasper tool tip blob could also cause the z coordinate of the position marker to be estimated higher than it actually is, causing \textit{laptrack} to mistakenly decide that no potential collisions would take place.
Figure 3.40: Grasper tool tip blob being broken by the obstacle tool marker.

Figure 3.41 shows the scenario where part of a second obstacle tool marker is also in the field of view. This could cause similar confusions to the program as in the scenario in Figure 3.39. laptrack would label the grasper tool tip blob as the obstacle, and one of the obstacle tool markers as the grasper tool tip. laptrack could also label the partial second obstacle tool marker as the obstacle, but the estimated z coordinate of the obstacle would be incorrect. Consequently, the obstacle avoidance function would malfunction.

Figure 3.41: Parts of two obstacle markers being in the field of view.
When only part of an obstacle tool marker is present in the field of view as shown in Figure 3.42, the obstacle avoidance function could also malfunction. First, *laptrack* could mistakenly label the obstacle tool marker blob as the grasper tool tip, and the grasper tool tip blob as the obstacle tool marker. Even if the blobs were correctly labelled, the estimated $z$ coordinate of the obstacle tool would be wrong because only part of the obstacle tool blob is in the field of view. Possible changes to improve the performance of the obstacle avoidance function are suggested in Chapter 6.

![Figure 3.42: Only part of the obstacle marker being in the field of view.](image)
CHAPTER 4

SURGICAL TOOL GESTURE RECOGNITION

The surgical tool gesture recognition function of our laparoscopic robotic system provides a convenient and intuitive HCI for surgeons. Literature reviews on different forms of HCI are presented at the beginning of this chapter. Section 4.1 introduces the eleven single-tool and double-tool gestures being experimented in our project, and provides an overview of the surgical tool gesture recognition process. Section 4.1.1 describes how the target tools are being segmented from the background. Section 4.1.2 defines the feature quantities extracted from the target tool blobs for classification, and discusses the extraction method. Section 4.1.3 describes the gesture classifier, a feed-forward neural network. Finally, Section 4.2 presents the testing results and analysis of the surgical tool gesture recognition function.

Since the conception of computers, HCI has become the centre of a never-ending evolution. From the early punch cards to the modern day keyboard, mouse, touch screen, hand-writing pad, and joystick, many different computer input and control devices have been developed and popularized. Some of the leading edge HCI research include speech recognition, eye tracking, facial feature tracking, electromyogram signal, brain signal, data glove, and hand gesture recognition.

In the project codenamed Dr. Who at Microsoft Research, Deng et al. [41] present the speech-centric multimodal human-computer interaction application MiPad that
address the mobile user interaction scenario. The MiPad is a wireless mobile PDA prototype that enables users to accomplish many common tasks using a multimodal spoken language interface and wireless-data technologies. It fully integrates continuous speech recognition and spoken language understanding, and provides a novel solution to the current prevailing problem of pecking with small styluses or typing on minuscule keyboards in today’s PDAs or smart phones.

Many eye tracking, facial feature tracking, head movement tracking, electromyogram signal, or brain signal based HCI have been developed to help paralyzed patients or people with significant physical disabilities gain access to computers. Ko et al. [42] propose a robust, fast and low-cost scheme for locating the eyes, lip-corners, and nostrils for the Eye-Head Controlled Human Computer Interface on a facial image with non-constrained background. First, the algorithm computes the similarity between all pairs of objects after the image is thresholded. The two objects with the greatest similarity are then selected as eyes. The algorithm has been tested on several sequential facial images with different illuminating conditions and varied head poses, and the average computing time of a 360 × 240 image is within 0.2 seconds.

Lyons et al. [43] propose a hybrid hands-off HCI that uses infrared video eye gaze tracking (EGT) and electromyogram (EMG) signals. This system combines the advantages of both sub-systems, providing quick cursor displacement in long excursions and steady, accurate movement in small position adjustments. The hybrid system also provides a reliable clicking mechanism. The testing results also show that the hybrid system is, on average, faster than the EMG-only system by a factor of two or more.
Brain-computer interface (BCI) is a communication channel from a human brain to a computer by only brain signals, and it does not resort to the usual human output pathways such as muscles. Progress and realization of practical BCI applications depend on systematic evaluations and comparisons of different brain signals, recording methods, processing algorithms, output forms, and operating protocols. However, the typical BCI system is designed specifically for one particular method and is, therefore, not suited to the systematic studies essential for further progress. Schalk et al. [44] have developed a general-purpose BCI research and development platform called BCI2000. BCI2000 can be incorporated alone or in combination with any brain signal, signal processing method, output device, and operating protocol, which makes it suitable for continued studies and development in the field of BCI.

Wearable HCI such as a data glove also provides a natural and convenient means for a human being to communicate with a computer. Chou et al. [45] developed a data glove with tactile feedback and an exoskeleton arm type haptic interface. The data glove has eleven degrees of freedom, and vibrators are mounted on the tip of five glove fingers. The arm type haptic interface has five degrees of freedom. It can be worn on an operator’s arm, and the operator can obtain real-time force feedback. The combination of data glove and arm type haptic device provide both external forces of contact and internal forces of grasping, which are great for completing complex works such as teleoperation and interacting with virtual environments.
4.1 The Surgical Tool Gesture Recognition Function

To provide a direct and intuitive surgeon-computer interface, our robotic surgical system employs an innovative surgical tool gesture recognition function. This function enables surgeons to command the laparoscopic surgery robotic system by performing tool gestures with the surgical tools already in hand during time-critical surgeries.

Minimally invasive surgery is performed through multiple small incisions (1/4" to 1/2" long) using specially-designed surgical instruments and viewed through a laparoscope as shown in Figure 4.1. The goal of the surgical tool gesture recognition function is to recognize tool gestures performed with the long-stem surgical tools.

![Figure 4.1: Laparoscopic surgery performed with long-stem surgical tools through laparoscope.](image)

Shown in Figure 4.2 are nine commonly used surgical tools, including grasper, scissors, dissector, hook scissors, spoon forceps, claw forceps, needle holder, cauteries, and fan retractors. All the surgical tools have an open state and a closed state, except for the cauteries and fan retractors. The multi-state surgical tools could create more tool gestures than the cauteries and fan retractors. Thus, we have chosen grasper and scissors, the two most commonly used multi-state surgical instruments, to perform tool gestures in...
our experiments. In addition to the open and closed states, the multi-state surgical tools
could also have intermediate states such as half-open, in the process of opening, or in the
process of closing. For simplicity, only the open and closed states are considered in our
experiments.

Figure 4.2: The nine commonly used surgical tools.
In Hsu and Payandeh [46], we experimented with four single-tool gestures performed with either a grasper or a pair of scissors shown in Figure 4.3a) to d), four non-overlapping double-tool gestures performed by combining grasper and scissors as shown in Figure 4.3e) to h), and three overlapping double-tool gestures performed with the combination of grasper and scissors as shown in Figure 4.3i) to k).

Figure 4.3: The eleven surgical tool gestures experimented.

In reality, not all eleven tool gestures would have satisfactory recognition rates, and not all eleven tool gestures would be needed. However, we would like to explore the
limits of surgical tool gesture recognition by trying to recognize as many tool gestures as possible. By experimenting more different tool gestures, we can also find out which ones would work and which ones would not work. The few tool gestures with high recognition rates can be chosen in the future to form a reliable HCI for surgeons to control various equipments or system functions.

The stainless steel tool tip of the surgical scissors shown in Figure 4.4a) does not provide even light reflection due to the curved surfaces. The darker areas would break up the tool tip blob into multiple blobs as shown in the binarized image in Figure 4.4b). Even light reflection from a surgical tool tip is critical in segmenting the tool tip blobs. Therefore, to provide even and strong light reflection, the stainless steel tips of the grasper and scissors are covered in white as shown in Figure 4.4c). The shape of the tool tip is much better preserved in the binary image shown in Figure 4.4d).

![Figure 4.4: The surgical tool tip marker.](image)

110
The surgical tool gesture recognition function is activated by manually pressing the 'Insert' key on the keyboard, and terminated by performing a cross tool gesture. The total sampling time of the function, which includes the time the surgical tools have to be held still, and the time it takes for the system to recognize the tool gesture, is within one second. Every acquired image, as shown in Figure 4.5a), undergoes three different stages of processing, including tool segmentation, feature quantity extraction, and neural network gesture classification. In the first stage, Figure 4.5b), tool blobs are segmented with morphological filters and thresholding. In the second step, Figure 4.5c), feature quantities are extracted from tool blobs using blob analysis. Next, the feature quantities are fed to a neural network, Figure 4.5d), for gesture classification. Finally, the surgical tool gesture is recognized as shown in Figure 4.5e). Since lens distortion is ignored in the input image capturing stage, some errors could already be present in the image, which could affect the tool gesture classification result.

Figure 4.5: The surgical tool gesture recognition stages.
4.1.1 Target Tool Segmentation

Segmentation refers to the process of extracting meaningful regions from images. Such regions typically correspond to objects of interest or to their parts. Numerous researchers have studied and developed segmentation methods to rid of undesired background information and to retain the objects of interests in images. The most popular approaches to target segmentation may be the thresholding technologies studied by Sahoo et al. [47]. In simple implementations, the segmentation is determined by a single parameter known as the intensity threshold. In a single pass, each pixel in the image is compared with this threshold. If the pixel's intensity is higher than the threshold, the pixel is set to white in the output. If it is lower than the threshold, it is set to black.

Contour-based algorithms are also commonly used in segmentation. Contour-based segmentation methods use edge or boundary information to detect features of interest in an image. Kass et al. [48] propose the model of snakes based on the deformation of an initial contour or surface towards the boundary of the detected object. The minima of deformation energy function is obtained at the boundary of the object.

Colour can also be used to segment objects from the background. Tai and Song [49] apply a histogram method for background removal before segmenting moving vehicles in a vision-based traffic monitoring system via thresholding. Based on the concept of probability, the background image can be constructed from the histogram of individual pixels in an image sequence. It is assumed that, in a greyscale image, the pixel intensity that is most frequently recorded within a certain intensity range in the image sequence is the intensity of the background pixel. Removing the pixels with that certain intensity would remove the background.
According to Gonzalez and Woods [50], mathematical morphology is also a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries, skeletons, and the convex hull. The language of mathematical morphology is the set theory. As such, morphology offers a unified and powerful approach to numerous image processing problems, including object segmentation. Xuan et al. [12] employ a two-step scheme for automatic object detection in Computed Radiography (CR) images utilizing morphological filters to extract the foreign objects and active contour models to outline the foreign objects. First, various structuring elements of the morphological filters are applied to effectively distinguish the foreign object candidates from the complex background structures. Second, active contour models are employed to accurately outline the morphological shapes of suspicious foreign objects such as tumours to further reduce the rate of false alarms.

According to Salenbier and Pardas [51], mathematical morphology is indeed very attractive for the purpose of object segmentation because it is a geometrical approach to signal processing and easily deals with criteria such as shape, size, contrast, connectivity, etc. Moreover, morphological operations can be efficiently implemented in both software and hardware. This point is of prime importance because the major bottleneck in segmentation is the complexity and computational load of the segmentation step. In other words, the simple and efficient nature of mathematical morphology could keep the processing time low, which is important in the time-critical surgery environment. With efficiency and processing time in mind, thresholding and mathematical morphology are our selected methods for target tool segmentation. The performance of simple thresholding and morphology with thresholding are compared at the end of this chapter.
In our laparoscope-captured image, the target of segmentation is the surgical tool tip, and the background to be removed is the plastic and rubber tissues and organs inside the plastic stomach model. The first target tool segmentation method experimented is thresholding. Every pixel in the greyscale input image, Figure 4.5(a), is compared with a fixed threshold value. Pixels with intensity values higher than the threshold value are set to white, and pixels with intensity values lower than the threshold value are set to black. The threshold value is chosen based on preliminary testing results with the brightness of the light source set to 35 lux. The output is a binary image with the target tool segmented as shown in Figure 4.5(b).

The second method, the mathematical morphology method with thresholding, takes the same greyscale input image, and suppresses the background while retaining the size and location information of the target tool with morphological filters. The image then goes through thresholding, leaving only the target tool in the binary output image as shown in Figure 4.5(b). The morphological operation used in this method is the background reduction described by Xuan et al. [12], which removes background objects larger than the target tool, and a modified background reduction, which removes background objects smaller than the target tool. Background reduction is defined as:

\[
\text{Background reduction} = A - (A \ominus B_I)
\]

(4.1)

where \( A \) is the input image shown in Figure 4.6(a), \( B_I \) is a square structuring element whose size is larger than the target tool of any shape, and "\( \ominus \)" is the greyscale opening operation, which is defined as:

\[
A \ominus B = (A \ominus B_I) \ominus B
\]

(4.2)
where $\oplus$ and $\ominus$ are grayscale dilation and erosion operations, respectively. Please refer to Appendix C for the definitions of grayscale dilation and erosion operations.

Figure 4.6(a) is the greyscale input image that contains the target tool, some small bright spots in the background, and a large piece of white paper. The result of the first step of background reduction, $A \ominus B_1$, is shown in Figure 4.6(b), where the target tool and small white spots are removed, and only the large piece of white paper remains. Then the complete result of background reduction is shown in Figure 4.6(c), which is the difference between Figure 4.6(a) and Figure 4.6(b). In Figure 4.6(c), the large piece of white paper has been removed, leaving only the target tool and small bright spots.

Figure 4.6: Target tool segmentation with morphological operations and thresholding.
To remove the small white spots in the background shown in Figure 4.6c), we changed the subtraction in Equation (4.1) to addition to form a modified background reduction as follows:

\[
\text{Modified background reduction} = A + (A \circ B_z) \quad (4.3)
\]

where \( A \) is now the image of Figure 4.6c), and \( B_z \) is a square structuring element that is smaller than the target tool.

\( A \circ B_z \) produces a greyscale image of the target tool as shown in Figure 4.6d), where all the background white spots are removed and only the target tool is left. Then, instead of subtracting Figure 4.6d) from image Figure 4.6c) as in background reduction, Figure 4.6d) is added to Figure 4.6c) as in Equation (4.3). The result is Figure 4.6e) with almost all pixels of the target tool saturated, that is, the intensity values of the pixels are at 255. In other words, the target tool is purely white, which makes it stand out among all other objects in the image.

The greyscale image is then converted to binary image by thresholding. Since the intensity values for almost all the target tool pixels are at the maximum 255 after the modified background reduction, the threshold value can be set close to the maximum 255. The higher threshold value would eliminate all the small background bright spots, leaving only the target tool in the image as shown in Figure 4.6f). This completes the target tool segmentation stage, and binary image Figure 4.6g) is ready for the next feature quantities extraction stage.

Figure 4.7 shows the flow chart for the target tool segmentation stage. The input greyscale image first undergoes background reduction to remove background objects
larger than the target tool. The resulting image is shown in Figure 4.6c). To eliminate background objects smaller than the target tool, modified background reduction is applied to the image, and the output image is Figure 4.6e). The resulting intensity values of the target tool pixels are at the maximum 255, which allows the threshold value to be set high to eliminate the leftover smaller background objects. Finally, the target tool is segmented, and the final output binarized image only contains the target tool blob ready for the feature quantities extraction stage. At this stage, the threshold value, lighting conditions, surgical tool orientations, and background tissue reflections could all affect how well the shape of the segmented target tool is preserved. Less well-preserved target tool blobs would result in less accurate feature quantities. Consequently, the tool gesture classification would be less accurate. Algorithm 4.1 shows the pseudo code for the target tool segmentation stage.

Figure 4.7: Target tool segmentation stage flow chart.
Algorithm 4.1: The target tool segmentation algorithm.

START
// Background reduction
COPY input image A to image buffer C
ERODE C with structuring element B₁
DILATE C with structuring element B₁
A - C
STORE the result in A
// Modified background reduction
COPY A to C
ERODE C with structuring element B₂
DILATE C with structuring element B₂
A + C
STORE the result in A
BINARIZE A by thresholding
END

4.1.2 Feature Quantities Extraction

Feature quantities are quantitative representations of target tool blobs in the final binarized image of the target tool segmentation stage shown in Figure 4.6g). They are distinctive information extracted from the target tool blobs that, together, allow the surgical tool gesture classifier to classify the surgical tool gestures. Six feature quantities are used in distinguishing the surgical tool gestures. Four of them are determined by the MIL blob analysis, and two of them are determined by a function developed by us. The feature quantities form the input vector to the surgical tool gesture classifying neural network that is presented in the next section.

The Matrox Imaging Library (MIL) blob analysis calculates more than thirty blob feature quantities. Among them, four are useful in distinguishing the shapes of target tool blobs in our case. The four feature quantities are compactness, roughness, elongation, and Feret elongation. The fifth and sixth feature quantity, the maximum color change count and the minimum color change count, are determined with a function we developed.

118
The maximum colour change count is the maximum number of black-to-white and white-to-black colour changes in the final binary image shown in Figure 4.6g). On the other hand, the minimum colour change count is the minimum number of black-to-white and white-to-black colour changes in the final binary image.

The definitions of the six feature quantities are as follows:

**Definition 1** Compactness is a measure of how close all particles in the blob are from one another. It is derived from perimeter and area. A circular blob such as Figure 4.8a) is most compact and is defined to have a compactness measure of 1.0 (the minimum); more convoluted shapes such as the hexagon in Figure 4.8b) have higher values. In our case, the open grasper shown in Figure 4.9a) has a compactness of 4.5, which is higher than the 2.2 of the closed grasper shown in Figure 4.9b).

![Compactness of different shapes.](image)

**Definition 2** Roughness is a measure of the unevenness or irregularity of a blob’s surface. It is a ratio of the perimeter to the convex perimeter of a blob. Smooth convex
blobs such as Figure 4.10a) have a roughness of 1.0, whereas rough blobs such as Figure 4.10b) have a higher value because their perimeter is bigger than their convex perimeter. For example, the roughness of the double triangle shown in Figure 4.11a) is 1.6, which is higher than the 1.1 of the closed grasper shown in Figure 4.11b).

![Figure 4.10: Roughness of different shapes.](image)

\[ \text{Roughness} > 1 \]

**Figure 4.10:** Roughness of different shapes.

![a) Double triangle roughness = 1.6 b) Closed grasper roughness = 1.1](image)

**Figure 4.11:** Roughness of double triangle and closed grasper.

**Definition 3**

Elongation is equal to the ratio of the blob length to the blob breadth. These features are derived from area and perimeter, using the assumption that the blob area is equal to \([\text{length} \times \text{breadth}]\) and the perimeter is equal to \([2(\text{length} + \text{breadth})]\). Breadth is also known as width. This quantity is a better measurement for long and thin blobs. The open grasper shown in Figure 4.12a) has an elongation of 11.8, while the closed grasper shown in Figure 4.12b) has a much smaller elongation of 4.5. From the definition, the open grasper shown in Figure 4.12a) should have a smaller elongation than the closed grasper in Figure 4.12b) due to the larger breadth. The miscalculation may be resulted from the fact that the grasper tool tip blob is not as long and thin as the feature.
extraction function prefers. However, it is found that the miscalculation is consistent, suggesting that this feature quantity could still be used in classifying the tool gestures.

Figure 4.12: Elongation of open grasper and closed grasper.

Definition 4 As shown in Figure 4.13, Feret elongation is equal to the ratio of the maximum Feret diameter to the minimum Feret diameter, which are determined by testing the diameter of the blob at several angles. This quantity is a better measurement for short and thick blobs. An open grasper shown in Figure 4.14a) has a Feret elongation of 1.3, while a closed grasper shown in Figure 4.14b) has a larger Feret elongation of 2.5.

Figure 4.13: The Feret diameters.

Figure 4.14: Feret elongation of open grasper and closed grasper.
Definition 5 Maximum and minimum colour change counts are determined with the same function. The number of white-to-black or black-to-white colour changes is counted along ten scan lines as shown in the binary image of an open grasper gesture in Figure 4.15. The centre of the scan is where the ten scan lines intersect, which is also the blob centre determined by blob analysis. The blob centre moves with the blob, so missing the centre is not an issue.

Figure 4.15: The ten scan lines drawn to determine the maximum and minimum colour change counts in a surgical tool gesture.

Surgical tools can come in at different yaw angles. Although performing a 360° scan would guarantee finding the maximum and minimum number of colour changes, it would cost extra processing time. To minimize the processing overhead, the minimum number of scan lines is decided based on the analysis shown in Figure 4.16.
Figure 4.16: Colour change count regions for the single tool gestures and the overlapping double-tool gestures.

Figure 4.16a) shows an open grasper gesture. The two yellow lines intersecting at the blob centre divide the plane into four regions. Any scan line drawn from the region labelled 2 that intersects the blob centre would encounter two black-to-white or white-to-black colour changes. Similarly, any scan line drawn from the region labelled 4 that intersects the blob centre would encounter four black-to-white or white-to-black colour changes. In the open grasper case, the low colour change count is two, and the high colour change count is four. The two yellow lines are almost perpendicular to each other. To accurately count the maximum and minimum number of colour changes regardless of the yaw angle of the surgical tool, the angles between each scan lines have to be smaller.
than 90°. Consequently, drawing three scan lines 60° to each other would guarantee finding the maximum and minimum number of colour changes.

Both of the maximum colour change count and minimum colour change count for the closed grasper gesture in Figure 4.16b), the closed scissors gesture in Figure 4.16d), and the cross gesture in Figure 4.16e) are two. As long as the scan line intersects the blob centre, the colour change count is always two regardless of the angle. As a result, only one scan line is need for these three cases.

The open scissors gesture shown in Figure 4.16c) is similar to the open grasper case except the angle between the two dividing lines is about 45°. Thus, it would require at least five scan lines 36° to each other to get the accurate maximum and minimum colour change counts.

There are three possible colour change counts, two, four, and six, for the triangle gesture shown in Figure 4.16f). To get the maximum colour change count of six, the angles between the scan lines have to be smaller than 40°. Furthermore, to get the minimum colour change count of two, the angles between the scan lines have to be smaller than 30°. Therefore, a minimum of seven scan lines 25.7° to each other is required.

The double triangle gesture shown in Figure 4.16g) also has three possible colour change counts. To get the minimum colour change count of two, the angles between the scan lines have to be smaller than 58°. To get the maximum colour change count of six, the angles between the scan lines have to be smaller than 20°. Therefore, a minimum of ten scan lines 18° to each other is required.
The non-overlapping double-tool gestures shown in Figure 4.3e) to h) are different combinations of single tool gestures in Figure 4.16a) to d), so they are treated as two single tool gestures. The minimum number of scan lines required for the non-overlapping double-tool gestures, therefore, are the same as the single tool gestures they consist of. To conclude, ten is the minimum number of scan lines to determine the maximum and minimum colour change counts in all eleven surgical tool gestures experimented.

The maximum and minimum colour change counts effectively divide the surgical tool gestures into three groups. The first group consists of the open grasper and open scissors. Their maximum colour change count is four, and their minimum colour change count is two. The second group includes the closed grasper, closed scissors, and cross. Their maximum colour change count and minimum colour change count are both two. The third group consists of the triangle and double triangle. Their maximum colour change count is six, and their minimum colour change count is two. The surgical tool gesture classifier could use these two feature quantities to classify a given tool gesture into one of the three groups. It can then further classify the tool gesture based on the other four feature quantities.

However, finding the right number of maximum and minimum number of colour changes is not always guaranteed. If the tool gesture is performed with extreme roll, yaw, or pitch angles, or if the shape of the target tool blob is not well-preserved, the blob centres of the target tool blobs would be far away from the anticipated blob centres shown in Figure 4.16. As a result, the number of maximum and minimum colour changes could be wrong. If the background is not completely removed, leaving small blobs
around the target tool blob, the maximum and minimum number of colour changes could also be wrong. Moreover, if the boundary of the target tool blob is not smooth, the maximum and minimum colour change count could also be more than what they are supposed to be.

4.1.3 Neural Network

The heart of a gesture recognition system is the gesture classifier. The gesture classifier is responsible for interpreting the information gathered from the input gesture and classifying the gesture. Neural network’s ability to learn by example makes them very flexible and powerful; furthermore, there is no need to devise an algorithm to perform a specific task. Neural network has the ability to make the best decision or approximation without understanding the complex underlying mechanism. Due to its parallel architecture, its fast response and computational times also makes it very well suited for real time systems. In addition, traditional numerical methods can be applied to feed-forward network training as well as algorithms invented especially for neural networks, so there is a wide range of algorithms that can be considered. Therefore, neural network is the chosen gesture classifier in our surgical tool gesture recognition function.

Neural networks, shown in Figure 4.17, are made of units that can be described by single numbers, their “activation” values. Each unit generates an output signal based on its activation. Units are connected to each other very specifically, each connection having an individual “weight” (again described by a single number). Each unit sends its output value to all other units to which they have an outgoing connection. Through these connections, the output of one unit can influence the activations of other units. The unit receiving the connections calculates its activation by taking a weighted sum of the input
signals (i.e. it multiplies each input signal with the weight that corresponds to that connection and adds these products). The output is determined by the activation function based on this activation (e.g. the unit generates output or "fires" if the activation is above a threshold value). Networks learn by changing the weights of the connections [2]. Appendix D provides an introduction on neural network structures, how neural network works, neural network learning process, and different types of neural networks.

![Feed-forward neural network](image)

**Figure 4.17: Feed-forward neural network.**

An open-source C++ neural network library, *nn-utility*, is added to *laptrack* for surgical tool gesture classification. The type of NN used is the feed-forward neural network with one hidden layer as shown in Figure 4.17. This type of structure, also used in the work by Shimada *et al.* [16], is the most commonly adopted structure in gesture recognition. In addition, a number of studies have shown that a three-layered feed-forward network has the ability to approximate any non-linear continuous function to an arbitrary degree of exactness [52]. Thus, feed-forward neural network is one of the most popular, effective and accurate classifiers for gesture recognition.

Our feed-forward neural network has six neurons in the input layer, twenty neurons in the hidden layer, and seven neurons in the output layer. The input vector
consists of the six feature quantities: compactness, roughness, elongation, Feret elongation, maximum colour change count, and minimum colour change count. The six feature quantities, which are six real numbers, are fed into the corresponding neurons in the input layer. The hidden layer neurons have sigmoid transfer functions. Twenty is determined to be the number of neurons that gives better results through testing. The output vector consists of the seven real numbers coming out of the seven neurons of the output layer. The seven numbers representing the seven single-tool and overlapping double-tool gestures, including open grasper, closed grasper, open scissors, closed scissors shown in Figure 4.3a) to d), and cross, triangle, and double triangle shown in Figure 4.3i) to k). The remaining four non-overlapping double-tool gestures shown in Figure 4.3e) to h) are basically different combinations of open and closed grasper and scissors. They are simply treated as two separate single-tool gestures as shown in Figure 4.18, and they require the system to make two tool gesture recognition calls.

![Figure 4.18: Non-overlapping double-tool gesture open scissors and closed grasper being separated into two single tool gestures: open scissors and closed grasper.](image)

The NN is trained under supervised learning with back-propagation, in which the training data is fed into the network for many iterations, so that the network learns the relationship between the input and output. The goal is to determine a set of weights that minimizes the errors between the neural network output and the training output. For the
four single-tool gestures, there are seven sets of training data acquired from performing the tool gestures with different roll, pitch, and yaw angles. Since the laparoscope is fixed on top of the plastic stomach model, these seven orientations simulate the laparoscope coming inside the plastic model from different angles. In other words, the surgical tool surface is not always parallel to the image plane. Shown in Figure 4.19, the seven orientations include a) natural position, b) positive roll angle, c) negative roll angle, d) smaller negative pitch angle, e) larger negative pitch angle, f) positive yaw angle, and g) negative yaw angle. The roll, pitch, and yaw angles of a surgical tool are defined in Figure 3.19. For the three overlapping double-tool gestures, there are seven sets of training data acquired from performing the tool gestures at different positions. As shown in Figure 4.20, the seven orientations are a) middle, b) up, c) down, d) left, e) right, f) close, and g) away.
Figure 4.19: The seven orientations of single-tool gestures.
Figure 4.20: The seven orientations of overlapping double-tool gestures.
Each set of training data contains an input vector and an output vector. Each surgical tool gesture has seven sets of training data corresponding to the seven orientations. A total of forty-nine training data sets for the seven surgical tool gestures are used in training. An example training input vector for the open grasper gesture would be $<4.34635, 1.2493, 1.3568, 1.38424, 4.0, 2.0>$. The six numbers correspond to compactness, roughness, elongation, Feret elongation, maximum colour change count, and minimum colour change count. An example training output vector for the open gesture would be $<0.9, 0.1, 0.1, 0.1, 0.1, 0.1>$. The seven numbers of the output vector represent open grasper, closed grasper, open scissors, closed scissors, cross, triangle, and double triangle. The first number, 0.9, which is significantly larger than the others, indicates that this is an open grasper gesture.

For each training iteration, one of the forty-nine training data sets is chosen as shown in the first step of Figure 4.21. Then the input vector of the chosen training data set is fed to the NN in the second step. In the third step, the errors between the output vector from the NN and the output vector of the chosen training data set is determined. In the fourth step, the set of weights are adjusted to minimize errors. In the above open grasper gesture example, weights would be adjusted such that when the same training input vector is fed into the NN again, the first number in the NN output vector would be closer to 0.9, while the other six numbers would be closer to 0.1.
After the NN is trained offline, the tool gesture recognition function is ready to recognize surgical tool gestures. If an open grasper gesture were performed, the surgical tool gesture function would segment the target tool and extract the feature quantities as described in the sections above. Then, an input vector of the form \(<\text{compactness}, \text{roughness}, \text{elongation}, \text{Feret elongation}, \text{maximum colour change count}, \text{minimum colour change count}>\) is fed to the NN for classification. The errors accumulated from the previous stages could cause the extracted feature quantities to deviate from their normal values. However, according to Appendix D, the NN’s ability to approximate nonlinear relationships would compensate for those errors. The NN can still correctly classify the tool gestures as long as the input feature quantities do not deviate outside a certain range.

An example input vector for the open grasper tool gesture would be \(<4.28054, 1.30689, 1.3597, 1.47503, 4.0, 2.0>\). Finally, the NN would output a vector such as \(<0.53, 0.15, 0.13, 0.056, 0.15, 0.18, 0.42>\). The largest number in the output vector corresponds to the tool gesture the input is most likely to be. The seven numbers represent open grasper, closed grasper, open scissors, closed scissors, cross, triangle, and double triangle. In this example, the largest of the seven numbers is 0.53, which is the first number. As a result, the input tool gesture is classified as an open grasper.
4.2 Results and Analysis

4.2.1 The Target Tool Segmentation Methods

Two target tool segmentation methods are proposed and tested with the experimental setup described in Section 3.1: 1) the background reduction and modified background reduction with thresholding and 2) simple thresholding. With the light source brightness set at 35 lux, the performance of the two methods are compared at three different distances: 30 mm, 50 mm, and 70 mm. The distance between the laparoscope and the grasper is defined in Figure 3.7. The threshold value is chosen based on a brightness of 35 lux. The original images and final binary images at different distances are shown in Figure 4.22. Figure 4.22a), b) and c) show the original image at 30 mm, 50 mm, and 70 mm, respectively. Figure 4.22d), e) and f) show the final binarized image of the target tool segmented with thresholding at 30 mm, 50 mm, and 70 mm, respectively. Figure 4.22g), h) and i) show the final binarized image of the target tool segmented with morphological filters and thresholding at 30 mm, 50 mm, and 70 mm, respectively.

When comparing the preservation of the shape of segmented grasper tool tip in the binarized images shown in Figure 4.22d) to i), both methods work equally well. However, the background reduction and modified background reduction with thresholding method is found unsuitable for our application for two reasons. The first reason is that the morphological operations take too much processing time. Using a computer with Pentium II 400 MHz CPU and 128 MB of SDRAM, it takes about fifteen seconds to perform the target tool segmentation with this method. In comparison, it takes only a fraction of a second to perform the target tool segmentation with simple thresholding. Within those fifteen seconds, the surgeon could have performed additional
tool gestures. Those following tool gestures would have been ignored by the system as a result. Therefore, this method is too slow to meet the real-time requirement of the surgical tool gesture recognition function.

![Image of original and binarized images at different distances](image)

**Figure 4.22:** The original images and final binarized images by the two methods at different distances.
The second reason is that the size of the grasper tool tip changes with the z coordinate, and the sizes of the structuring elements $B_1$ and $B_2$ have to change accordingly; otherwise, the method would fail. In Equation (4.1), the background reduction, the structuring element $B_1$ has to be larger than the target tool in order to remove background objects larger than the target tool. In Equation (4.3), the modified background reduction, the structuring element $B_2$ has to be smaller than the target tool to remove background objects smaller than target tool. Figure 4.22(g) to (i) are produced with the size of $B_1$ and $B_2$ fixed. Although the grasper tool tips are segmented successfully in all three cases, some of the background spots are being enhanced at the same time. The two linkages near the grasper tool tip in Figure 4.22(g) are enhanced. One of the linkages and two spots behind the tool trunk are enhanced in Figure 4.22(h). Two spots behind the grasper are enhanced in Figure 4.22(i). Those background spots could cause confusion for laptrack in labelling the position marker blob and grasper tool tip blob. If the blobs were labelled incorrectly, both or either one of the instrument tracking function and surgical tool gesture recognition function would fail.

The segmentation results of the background reduction and modified background reduction with thresholding method could be improved by varying the sizes of structuring elements $B_1$ and $B_2$. However, it is hard to overcome the significantly longer processing time. It is impractical to wait for fifteen seconds for a tool gesture to be recognized during real surgeries; therefore, the thresholding method is used for target tool segmentation instead of the background reduction and modified background reduction with thresholding.
The thresholding method has its own weaknesses. First, the threshold value is fixed, implying that this method could only work for a certain range of light source brightness. In our case, the threshold chosen value works for brightness in the range of 25 lux to 45 lux. To work under higher brightness, the threshold value will have to be raised, and vice versa. The second weakness of the thresholding method is its inability to remove background spots that are as bright as or brighter than the target tool. In real surgeries, there are moist tissues in the background that can cause strong reflections. Since the thresholding method only considers the pixel intensity values, it would not be able to distinguish between those bright background spots and the target tool. However, with the plastic stomach model constructed with plastic and rubber, and the surgical tools covered in white, there are no such concerns in our case. The background will always be darker than the target tool, and there are no background bright spots brighter than the target tool. Therefore, the thresholding target tool segmentation method works well with our experimental setup.

4.2.2 Surgical Tool Gesture Recognition

With the brightness of the light source set at 35 lux, the four single-tool gestures shown in Figure 4.3a) to d) are performed ten times at each of the seven orientations and ten times at ten different random orientations inside the plastic stomach model with the experimental setup described in Section 3.1. The tool gestures are performed at distances, or the z-coordinate as shown in Figure 3.7, between 30 mm to 70 mm underneath the laparoscope. As shown in Figure 4.19, the seven orientations include natural position, positive roll angle, negative roll angle, smaller negative pitch angle, larger negative pitch angle, positive yaw angle, and negative yaw angle. As defined in Figure 3.19, a smaller
negative pitch angle means the surgical tool is closer to the laparoscope. A larger pitch angle means the surgical tool is further away from the laparoscope. In other words, the distance between the laparoscope and the surgical grasper is larger. The three overlapping double-tool gestures (shown in Figure 4.3i) to k) are also performed ten times at each of the seven orientations and ten times at ten different random orientations inside the plastic stomach at distances between 30 mm to 70 mm. As shown in Figure 4.20, the seven orientations are middle, up, down, left, right, close, and away.

An image from outside the plastic stomach model during testing is shown in Figure 4.23. For the remaining four non-overlapping double-tool gestures (shown in Figure 4.3e) to h), since they are basically combinations of the four single tool gestures, their recognition rates are roughly the same as the lower one of the single tool gestures they consist of.

Figure 4.23: A look at surgical tool gesture recognition function testing from outside.

Figure 4.24 shows the open grasper gesture being recognized inside the plastic model. Shown in Figure 4.25, the recognition rates for the seven orientations are all
100%. The recognition rate for random orientations, which involves different combinations of various roll, pitch, and yaw angles, is 90%. Overall, the open grasper gesture can be recognized at a success rate of 98.75%. Being one of the basic single tool gestures, the open grasper gesture is expected to have a high recognition rate.

Figure 4.24: The open grasper gesture being recognized at natural position.

Figure 4.25: Recognition rates of the seven open grasper gesture orientations.
Figure 4.26 shows the closed grasper gesture being recognized inside the plastic stomach model. Shown in Figure 4.27, the recognition rates for the natural, positive roll angle, negative roll angle, smaller negative pitch angle, larger negative pitch angle, positive yaw angle, negative yaw angle, and random orientations are 90%, 100%, 70%, 100%, 80%, 70%, 100%, and 80%, respectively. This gesture is better recognized at natural, positive roll angle, smaller negative pitch angle, and negative yaw angle orientations. The overall recognition rate for the closed grasper gesture is 86.25%.

Although the overall recognition rate of this gesture is not as high as the recognition rate for open grasper gesture, the closed grasper gesture is still a reliable gesture, as it can be recognized almost nine out of every ten times.

Figure 4.26: The closed grasper gesture being recognized at natural position.
Figure 4.27: Recognition rates of the seven closed grasper gesture orientations.

Figure 4.28 shows the closed scissors gesture being recognized inside the plastic stomach model. The surgical grasper is also present in the image because the position marker on the grasper is required to keep the surgical instrument tracking function running properly. Figure 4.29 shows that the recognition rates for the natural, positive roll angle, negative roll angle, smaller negative pitch angle, larger negative pitch angle, positive yaw angle, negative yaw angle, and random orientations are 90%, 80%, 100%, 100%, 100%, 70%, 70%, and 80%, respectively. The recognition rates are lower at either positive or negative yaw angles. Overall, the closed scissors gesture can be recognized at a rate of 86.25%, making it a reliable tool gesture.
Despite being one of the basic single tool gestures, the 48.75% overall recognition rate for the open scissors gesture shown in Figure 4.30 is far lower than the other three single tool gestures. Shown in Figure 4.31, the recognition rate for the natural, positive roll angle, negative roll angle, smaller negative pitch angle, larger negative pitch angle, positive yaw angle, negative yaw angle, and random orientations are 20%, 70%, 90%,
50%, 0%, 50%, 60%, and 50%, respectively. The recognition rates are extremely poor at natural position (20%) and large negative pitch angle (0%). However, the recognition rates are significant higher when the open scissors gesture is performed at either a positive roll angle (70%) or a negative roll angle (90%).

The open scissors gesture is often misidentified as the triangle gesture shown in Figure 4.3j, resulting in a low overall recognition rate. One possible contributing factor to the low recognition rate is that the open scissors is part of the triangle gesture. Consequently, the feature quantities extracted from an open scissors gesture may confuse the neural network because they are similar to those of a triangle gesture. The higher recognition rates at positive and negative roll angles also support this assumption. When the open scissors gesture is performed at a roll angle, the features quantities extracted would be less similar to those of a triangle gesture that contains an open scissors at natural orientation.

![Figure 4.30](image-url) The open scissors gesture being recognized at natural position.
Figure 4.31: Recognition rates of the seven open scissors gesture orientations.

Figure 4.32 shows the overlapping double-tool cross gesture being recognized inside the plastic stomach model. Figure 4.33 shows that the recognition rates for the middle, up, down, left, right, close, away, and random orientations are 50%, 70%, 30%, 60%, 40%, 40%, 80%, and 70%, respectively. The overall recognition rate of this gesture is 55%. The recognition rate are above average at the up, left, and away orientations. At 80%, the recognition at the away orientation is the highest of all seven orientations, which means the cross gesture is actually better recognized when it is performed further away from the laparoscope.
Figure 4.32: The cross gesture being recognized at middle position.

Figure 4.33: Recognition rates of the seven cross gesture orientations.

The overall recognition rate of the triangle gesture shown in Figure 4.34 is 38.75%. Figure 4.35 shows that the recognition rates for the middle, up, down, left, right, close, away, and random orientations are 20%, 40%, 30%, 40%, 50%, 50%, 40%, and
40%, respectively. The recognition rates are higher at the right and close orientations, where the scissors are in the middle of the image and are better illuminated.

Figure 4.34: The triangle gesture being recognized at middle position.

![Triangle Gesture Recognition Testing Results](image)

Figure 4.35: Recognition rates of the seven triangle gesture orientations.

The double triangle gesture shown in Figure 4.36, which is the hardest to perform, is unrecognizable. Figure 4.37 shows that the recognition rates for all the orientations are
0%. It is usually recognized as the single-tool open grasper gesture. Visually, the double triangle gesture is consists of an open grasper gesture and an open scissors gesture, so it is not surprising that the double triangle gesture is often recognized as an open grasper gesture.

Figure 4.36: The double triangle gesture often being recognized as open grasper.

![Figure 4.36: The double triangle gesture often being recognized as open grasper.](image)

Overall, the overlapping double-tool gestures are more difficult to perform than single-tool gestures, so their recognition rates are expected to be lower than those of
single-tool gestures. When performing an overlapping double-tool gesture, sometimes it is difficult to manipulate the two surgical tools to form a single blob. Sometimes the positions of the two surgical tools with respect to each other could be too far off from any one of the designed tool gestures in Figure 4.3. Consequently, the performed overlapping double-tool gestures are misidentified either as two single-tool gestures, or as the closest tool gesture the neural network interprets them to be.

In addition, for an overlapping double-tool gesture to be recognized, the two tool tips are supposed to overlap and form only a single blob. This requirement is often not easy to achieve due to the shadows of the surgical tools. In some cases, the shadow could actually help in obtaining the depth information. However, in our case, the shadow of one surgical tool cast on the other could often break up the single blob into two. As a result, the overlapping double-tool gestures are often misidentified as two single-tool gestures as shown in Figure 4.38, which further lowers the recognition rates.

![Figure 4.38: One overlapping double-tool gesture being broken into two single-tool gesture by the shadow of the scissors cast on the grasper.](image)
For the four non-overlapping double-tool gestures, their recognition rates roughly equal to the lower one of the two single-tool gestures they are consisted of. The closed scissors and closed grasper non-overlapping double-tool is shown in Figure 4.28. The open scissors and closed grasper non-overlapping double-tool is shown in Figure 4.30. Figure 4.39 shows the open scissors and open grasper non-overlapping double-tool being recognized. Figure 4.40 shows the closed scissors and closed grasper non-overlapping double-tool being recognized.

Figure 4.39: The open scissors and open grasper gesture being recognized.

Figure 4.40: The closed scissors and open grasper gesture being recognized.
The overall surgical tool gesture recognition rates for the seven single-tool and overlapping double-tool gestures are shown in Figure 4.41. The recognition rates for the open grasper gesture, closed grasper gesture, open scissors gesture, closed scissors gesture, cross gesture, triangle gesture, and double triangle gesture are 98.75%, 86.25%, 48.75%, 86.25%, 55%, 38.78%, 0%. The open grasper, closed grasper and closed scissors gestures can all be recognized at over 86%, which suggests that they are reliable tool gestures. Although the overall recognition rate of the open scissors gesture is only 48.75%, it can still be recognized about eight out of ten times when the gesture is performed at either positive or negative roll angles. Therefore, this gesture could still be considered a reliable tool gesture, as long as the operator keeps in mind that the gesture has to be performed with a roll angle.

Figure 4.41: Overall surgical tool gesture recognition rates for the seven single-tool and overlapping double-tool gestures.
The overlapping double-tool gestures are not as recognizable as the single-tool gestures. At 55%, the recognition rate of the cross gesture is barely over 50%. However, when performed at a distance from a laparoscope, this gesture can actually be recognized at 80%, which is a respectable percentage. The triangle and double triangle gestures, recognized at 38.75% and 0% overall, are the two gestures with the lowest recognition rates. The tool gesture recognition function has to be improved to raise their recognition rates in the future for these two tool gestures to be useful.

Table 4.1 shows the confusion matrix of the tool gesture recognition function. For example, the second row shows that the open grasper tool gesture is correctly recognized seventy-nine times out of eighty trials. Only one trial is misidentified as the double triangle tool gesture. The confusion matrix shows that most of the tool gestures are misidentified as the triangle gesture for a significant amount of trials, suggesting that the recognition rates of these tool gestures, especially the open scissors and cross gestures, could be improved by removing the triangle tool gesture.

Table 4.1: Tool gesture recognition confusion matrix.

<table>
<thead>
<tr>
<th>Recognized Performed</th>
<th>Open Grasper</th>
<th>Closed Grasper</th>
<th>Open Scissors</th>
<th>Closed Scissors</th>
<th>Cross</th>
<th>Triangle</th>
<th>Double Triangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Grasper</td>
<td>79</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Closed Grasper</td>
<td>0</td>
<td>69</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Open Scissors</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>10</td>
<td>0</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>Closed Scissors</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>69</td>
<td>2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Cross</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>44</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>Triangle</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>28</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>Double Triangle</td>
<td>46</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>29</td>
<td>0</td>
</tr>
</tbody>
</table>
Although the illumination brightness as tested is 35 lux, the gesture recognition still works quite well when the brightness is as low as 25. When the brightness is increased, the gesture recognition works well up until about 50. However, it is not suggested to turn up the brightness too high because the background objects would be too bright and pass the threshold value, causing confusions in the target tool segmentation process. In addition, the tool gestures are suggested to be performed as close to the centre of the image as possible, where the illumination is strong and even, and without extreme roll, pitch, or yaw angles.

Overall, the high recognition rate of single-tool gestures and non-overlapping double-tool gestures show that the surgical tool gesture recognition function is a practical and reliable HCI, and is up for the challenge of executing vital commands during surgery. With the development of a more accurate tool gesture classifier that could classify overlapping double-tool gestures more accurately and recognize more different surgical tool gestures, this function would definitely impact the evolution of surgical surgeon-computer interface.
CHAPTER 5

SURGICAL TOOL MOTION RECOGNITION

Surgical tool motion recognition is another method of direct and intuitive HCI. For example, in Apple computers, moving the mouse cursor across the screen to the right would bring up an overview of all the windows currently running. The user could then easily find the window he or she is looking for and display it on the screen in full size.

Several motion recognition methods have been studied or commercialized, including sensor tracking, eye-gaze tracking, and video tracking. The Aurora System by Northern Digital Inc. [53] uses an electromagnetic measurement technology that has been designed for applications requiring precise, real-time spatial measurements without having to worry about line-of-sight. The 8 mm x 0.8 mm sensors can be placed on the surgical tools to track tool motions.

Eye-gaze tracking is a non-contact, non-restrictive and intuitive HCI. Hennessey et al. [54] develop an eye-gaze tracking system using a single high-resolution camera with a fixed field of view. The point of gaze of a user is closely related to user intention. By tracking the eye-gaze of a user, valuable insight may be gained into what the user is thinking of doing, resulting in more intuitive interfaces and the ability to react to the user’s intentions rather than explicit commands. Although eye-gaze tracking has not been applied in real laparoscopic surgeries, Law et al. [55] have studied the eye movements of five experts and five novices performing a one-handed task on a computer-based
laparoscopic surgery simulator. They conclude that performance and eye movement differ between the two groups. With further development, the eye-gaze tracking system can become an alternative HCI in laparoscopic operating rooms.

Video taken from a single monocular camera is the most common means of recording human action; thus, video tracking is another popular and well-studied method for motion tracking. Park et al. [56] present a semiautomatic method for synthesizing human motions from a single video stream by using a motion library. Choosing a 3D motion in the library as a reference, they resolve the inherent depth ambiguity in motion synthesis. Their experimental results show that the method can synthesize highly dynamic motions such as kicking and header motions of soccer players. Efros et al. [57] introduce a novel motion descriptor based on optical flow measurements that recognizes human actions recorded in videos at a distance, at resolutions where a whole person may be only 30 pixels tall. They demonstrate the use of that descriptor, in a nearest-neighbour querying framework, to classify actions, transfer 2D/3D skeletons, as well as synthesize novel action sequences.

Our goal is to develop a HCI without any special hardware beyond the laparoscopic camera system and the image processing computer. The sensor tracking method requires expensive sensors and control units, which does not suit our goal. The eye-gaze tracking method is excellent, but it requires an extra external camera to track the eye movements. Tracking surgeons’ hands through videos is another alternative method, but it also requires an extra external camera. Moreover, the hands could move out of the camera field of view or be blocked by other instrument of equipment during surgeries. The cluttered operating room also makes it hard to maintain an undisrupted line-of-sight...
between the camera and the surgeon. As a result, this method would not be reliable enough in the real surgery environment.

We propose an alternative video motion tracking method. Instead of tracking the surgeon’s eyes or hands, our method tracks the surgical tool motions inside the stomach. By utilizing the existing laparoscopic camera image, no extra external cameras or sensors are required. In addition, the same surgical tools and laparoscope camera system can be retained since our method is purely software. Thus, there are no extra costs in replacing the current surgery equipment.

For our laparoscopic surgery robotic system, the surgical tool motion recognition function could compliment the surgical tool gesture recognition to add more “words” to the set of “vocabulary”. The more “words” there are, the more functions the surgeon could command the computer to execute through the surgical tool gesture and motion recognition. Section 5.1 describes the three steps of the surgical tool motion recognition function used in our system. Section 0 presents the test results and analysis of the surgical tool motion recognition function. Section 5.3 presents a preliminary laparoscopic robotic system HCI using a combination of surgical tool gesture recognition and surgical tool motion recognition.

5.1 The Surgical Tool Motion Recognition Function

As shown in Figure 5.1, our system is capable of recognizing the 2D projections of four simple motion patterns the surgical grasper performs. The 2D projections of the four motion patterns are line segment (Figure 5.1a), two line segments and an angle (Figure 5.1b), curve (Figure 5.1c), and loop (Figure 5.1d). After the surgical tool gesture
The recognition function is activated by manually pressing the ‘Insert’ key on the keyboard. An open grasper gesture would trigger the surgical tool motion recognition function. The surgical tool motion recognition function would analyze the subsequent grasper tool motion and send a message to the motor control computer over the network. The motor control computer will then display the corresponding image on the monitor.

Figure 5.1: The four 2D motion pattern projections.

5.1.1 Motion Trajectory Sample Point Collection

There are three stages in the motion recognition function, including the position marker motion trajectory sample point collection, the displacement vector discretization, and the motion trajectory classification. In the first stage, sample points of the position marker motion trajectory from the start position to the end position, as shown in Figure 5.2, are collected. Utilizing the surgical tool tracking function, our system is capable of collecting 3D sample points. When the surgical tool motion recognition function is activated, 3D sample points from each frame is stored until the motion trajectory collection is terminated by either performing a closed grasper gesture or reaching the
maximum number of fifteen sample points. Figure 5.3 shows the fifteen collected 3D sample points of the position marker motion trajectory in Figure 5.2. For simplicity, the 3D sample points are projected onto the x-y plane as illustrated in Figure 5.4. Next, the 2D sample points are passed on to the displacement vector discretization stage.

Figure 5.2: The 3D movements of the position marker on the grasper.

Figure 5.3: The 3D position marker motion trajectory sample points.
5.1.2 Displacement Vector Discretization

From the 2D sample points shown in Figure 5.4, displacement vectors, \( \nu = <dx, dy> \), between each sample point are calculated. To classify the motion pattern, the displacement vectors are assigned with one of the eight discrete orientations as shown in Figure 5.5. A displacement vector would be assigned with orientation 0 if the angle it makes with respect of the horizontal x-axis is between \(-\pi/8\) and \(\pi/8\). Similarly, orientation 1 to 7 are between \(\pi/8\) and \(3\pi/8\), \(\pi/4\) and \(5\pi/8\), \(3\pi/8\) and \(7\pi/8\) and \(5\pi/8\) and \(7\pi/8\), and \(-\pi/8\) and \(-3\pi/8\), respectively.

In Figure 5.4, the angles of the first seven displacement vectors are all between \(-\pi/8\) and \(\pi/8\), so they are assigned with orientation 0 (pointing in the positive x-axis.)
direction), and shown as v1 to v7 in Figure 5.6. The eighth displacement vector in Figure 5.4 is assigned with orientation 7 because its angle is between $\frac{3\pi}{8}$ and $-\frac{\pi}{8}$. It is shown as v8 in Figure 5.6.

![Figure 5.5: The eight possible discrete orientations of displacement vectors.](image)

![Figure 5.6: The motion pattern represented by discrete displacement vectors.](image)

The angles of the ninth and tenth displacement vectors in Figure 5.4 are between $-\frac{5\pi}{8}$ and $-\frac{3\pi}{8}$, so they are assigned with orientation 6 (pointing in the negative y-axis).
direction), and shown as v9 and v10 in Figure 5.6. The angle of the eleventh displacement vector in Figure 5.4 is between $\frac{7\pi}{8}$ and $\frac{5\pi}{8}$, so it is assigned with orientation 5 and shown as v11 in Figure 5.6. The angles of the last three displacement vectors in Figure 5.4 are between $\frac{5\pi}{8}$ and $-\frac{7\pi}{8}$, so they are assigned with orientation 4 (pointing in the negative x-axis direction), and shown as v12 to v14 in Figure 5.6. The motion pattern represented by discretized displacement vectors is ready for classification.

5.1.3 Motion Pattern Classification

Algorithm 5.1, presented by Hienz et al. [27], is applied to classify the four motion patterns shown in Figure 5.1. A neural network is not required for surgical tool motion classification because there is a direct relationship between the number of orientation changes and the motion pattern. The algorithm works as follows. If there is no change in orientations, then the motion pattern must be a line segment. If there is only one change in orientations, then the motion pattern must be two line segments and an angle. The angle is calculated by taking the difference between the two different orientations. For example, if there is an orientation change from orientation 1 to orientation 3, then the angle between the two orientations is

\[(3 - 1) \times 45' = 90'\]

If there are multiple changes in orientations without visiting all possible eight orientations, then the motion pattern is classified as a curve. Everything else would be classified as a loop. Following the classification algorithm, the motion pattern in Figure 5.6 would be classified as a curve motion.
Algorithm 5.1: The motion pattern classification algorithm.
START
  IF no change in orientations THEN
    Motion pattern is a straight line
  ELSE IF only one change in orientations THEN
    Motion pattern is a straight line + angle
  ELSE IF more than one change of orientations, but not in all 8 possible orientations THEN
    Motion pattern is a curve
  ELSE
    Motion pattern is a circular path
  END IF
END

After preliminary testing, it is discovered that dividing the plane into eight regions far exceeds the requirement. Given the difficulties in manipulating a surgical grasper precisely in the desired motion, motion patterns would be classified as curves more than 80% of the time. For example, one little shake of the wrist would cause vibration in the motion pattern, causing a line segment or two line segments and an angle to be classified as a curve. In addition, a loop motion pattern is almost impossible to achieve because it is too difficult to cover all possible eight orientations in seconds. As a result, the eight possible orientations are reduced to four as shown in Figure 5.7. The motion pattern classification algorithm remains the same.
The new workflow of laptrack is outlined as follows. After surgical tool tracking has been started, manually pressing the “Insert” key on the keyboard activates the surgical tool gesture recognition function. Whenever an open grasper gesture is recognized, the surgical tool motion recognition function is activated as shown at the bottom of Figure 5.8. laptrack will start recording the motion trajectory sample points. If the motion recognition function were not terminated by performing a closed grasper gesture, the allowable time to perform tool motions would depend on the processing speed of the computer. Our image processing computer takes about two seconds to collect the fifteen motion trajectory sample points. The full surgical tool motion must be performed within two seconds to be correctly recognized. After the sample points are collected, the tool motion can be recognized in a fraction of a second. A message containing the recognized tool motion will be sent to the demo1 program running on the
server computer. *demo1* will then display the image corresponding to the recognized tool motion on the monitor.

![Figure 5.8: An open grasper gesture activates the surgical tool motion recognition function.](image)

The following figures show the example 3D position marker motion trajectory (Figure 5.9, Figure 5.12 and Figure 5.15), the 3D motion trajectory sample points (Figure 5.10, Figure 5.13 and Figure 5.16), and the 2D projection of the motion trajectory sample points (Figure 5.11, Figure 5.14 and Figure 5.17) for the line segment motion, two line segment and an angle motion, and loop motion, respectively.
Figure 5.9: The 3D line segment motion trajectory of the position marker on the grasper.

Figure 5.10: The 3D line segment position marker motion trajectory sample points.
Figure 5.11: The 2D projection of the position marker line segment motion trajectory sample points.

Figure 5.12: The 3D two line segments and an angle motion trajectory of the position marker on the grasper.
Figure 5.13: The 3D position marker two line segments and an angle motion trajectory sample points.

Figure 5.14: The 2D projection of the position marker two line segments and an angle motion trajectory sample points.
Figure 5.15: The 3D loop motion trajectory of the position marker on the grasper.

Figure 5.16: The 3D position marker loop motion trajectory sample points.
5.2 Results and Analysis

With the light source brightness set at 35 lux, each of the four surgical tool motions patterns, including line segment, two line segments with an angle, curve, and loop are performed fifty times with the grasper within 30 mm to 70 mm below the laparoscope. The experimental setup is the same as described in Section 3.1. As long as the position marker on the grasper can be recognized by laptrack, the orientation of the grasper does not affect the performance of this function because only the coordinates of the position marker is used to determine the surgical tool motion. All tool motions are performed and completed within two seconds as the computer takes about two seconds to collect the maximum number of fifteen motion trajectory sample points.

The line segment tool motion can be recognized at a rate of 36%. Figure 5.18 shows the line segment tool motion being recognized by laptrack. The icon representing
the line segment motion is also displayed underneath the recognized tool motion. The low recognition rate is expected given the difficulties in manipulating a surgical tool constrained by the incision hole through the rubber stomach wall. A small bump in the grasper movement could cause the motion trajectory to become a curve.

![Figure 5.18: The line segment tool motion being recognized.](image)

In general, there are two ways of performing a line segment tool motion: 1) along the tool axis as shown in Figure 5.19, and 2) not along the tool axis. Evidently, the higher recognition rate of 56% shown in Figure 5.20 suggests that to perform a line segment tool motion by moving the grasper along the tool axis is easier. The recognition rate (16%) is much lower when the grasper is not moved along the tool axis. Line segment motion not along the tool axis is more difficult to perform due to the fact that the grasper needs to be carefully pulled out or pushed in while the yaw angle of the grasper is changing. In contrast, line segment motion along the tool axis can be easily performed by pulling out or pushing in the grasper.
Figure 5.19: The line segment tool motion performed along the tool axis.

Figure 5.20: Recognition results of line segment tool motion performed along and not along tool axis.

The recognition rate of the two line segments and an angle motion is 40%. Figure 5.21 shows the two line segments and an angle tool motion being recognized. This tool
motion is often misidentified as the curve tool motion, which implies that there is at least one orientation change too many. This motion is basically a combination of two shorter line segments. As it turns out, performing two shorter line segments is easier than manipulating the grasper in a longer line segment because it requires better skills and higher concentration to keep the grasper moving in a longer straight line. Consequently, the recognition rate of this tool motion (40%) is slightly higher than that of the line segment motion (36%).

Figure 5.21: The two line segments and an angle tool motion being recognized.

The recognition rate of the curve tool motion is 44%. Figure 5.22 shows the curve tool motion being recognized by laptrack. It is often misidentified as the two line segments and an angle tool motion. The difference between the curve motion and two line segments and an angle tool motion is just one orientation change. In performing this motion, user has to be careful with how sharp and where the turns have to be. Not turning sharply enough would result in a two line segments and an angle tool motion. At the
same time, turning too sharply or making too many turns would turn the motion into a loop tool motion.

![Figure 5.22: The curve tool motion being recognized.](image)

The recognition rate of the loop tool motion is 50%. Figure 5.23 shows the loop tool motion being recognized. It is the easiest tool motion to perform since it does not require careful tool manipulations other than moving the tool around to generate displacement vectors in all four orientations. Consequently, it has the highest recognition rate of all four surgical tool motions. However, there are still times when the grasper does not make enough turns to generate displacement vectors in all four orientations, resulting in the tool motion being misidentified as the curve tool motion.
The overall surgical tool motion recognition testing results are shown in Figure 5.24. The recognition rates for the line segment tool motion, two line segments and an angle tool motion, curve tool motion, and loop tool motion are 36%, 40%, 44%, and 50%. The first two tool motions, which involve manipulating the grasper in straight lines, have lower recognition rates. However, when the line segment is performed along the tool axis, the recognition rate is much higher (56%) because it is much easier to manipulate the grasper in a straight line along the tool axis than not along the tool axis, which has a low recognition rate of 16% as shown in Figure 5.20. On the other hand, the curve and loop tool motions have higher recognition rates because they do not require as many skills and as much concentration in performing the surgical tool motions.
Emphasis must be drawn on the fact that the effectiveness of this surgical tool motion recognition function cannot be judged based on its recognition rate only. Other than the tool motion classification algorithm, the actual performance of the tool motion recognition function is affected by two other major factors, the surgical environment and user skills.

In terms of the surgical environment, there is limited room inside the plastic stomach model for surgical tool motion performance. Any accidental contacts between the surgical grasper and surrounding tissue could turn the intended tool motion into another one. In addition, the surgical tool is constrained by the incision hole, which presents further challenges in making precise tool motions.

Figure 5.25 summarizes the results of testing the surgical tool motion recognition function outside the plastic stomach model. The line segment, two line segments and an angle, curve, and loop tool motions can be recognized at 66%, 76%, 82%, and 88%.
respectively. These much higher recognition rates suggest that most of the time, a surgical tool motion is misidentified because the tool motion is mis-performed inside the plastic stomach model, not because the tool motion is misclassified by the surgical tool motion recognition function.

Figure 5.25: Surgical tool motion recognition testing results outside the plastic stomach model.

Due to precision limitations of hand-held tools and the lack of direct visual guidance to the target, user skills in manipulating the surgical tools is another major factor in tool motion recognition. Users experienced in manipulating these long and thin surgical tools through a monitor should do better than novice users. Practice could also improve the performance of tool motion recognition.

In order for the surgical tool motion recognition function to be a practical and reliable HCI, not only do the users have to improve their skills, but the function itself also has to be improved with a different algorithm or classifier. A couple of motion recognition algorithms are described in Chapter 6 for possible improvements.
5.3 Hybrid HCI

After developing and testing both the surgical tool gesture recognition and surgical tool motion recognition functions, we propose a hybrid HCI utilizing a combination of the two functions. The original surgical tool gesture recognition function is triggered by manually pressing the ‘Insert’ key on the keyboard, and terminated by performing a cross tool gesture. The original tool motion recognition function is triggered by performing an open grasper and terminated by either performing a closed grasper gesture or reaching the maximum number of motion trajectory sample points. In real surgeries, it may not be possible for the surgeon to empty one hand and manually press a key on the computer keyboard to activate the surgical tool recognition function. In addition, some of the tool gestures and tool motions have recognition rates lower than 50%, which do not guarantee reliable performance.

With the new hybrid HCI, the surgeons only need to use one tool gesture and one tool motion to access three different types of medical images without taking their hands off the surgical tools to press a button. Furthermore, only one tool gesture, the open grasper gesture, and one tool motion, the loop motion, is required to control the HCI. The open grasper tool gesture and the loop tool motion are chosen because of their higher recognition rates compared with the rest of the tool gestures and tool motions. Here is how to trigger, use, and terminate the tool gesture and motion recognition hybrid HCI:

1. Interface Activation: The tool gesture and motion recognition HCI is activated by keeping the scissors stationary within a shaded activation area at the bottom of the camera image as shown in Figure 5.26 for three seconds. The system uses the 2D projection of the centre of the scissors...
tool tip blob to determine if the scissors are within the activation area. The grasper is not used because it can never rest in the activation area with the automatic tracking laparoscope-positioning robot trying to keep it in the middle of the field of view.

![Tool gesture recognition activation area](image)

**Figure 5.26: The tool gesture recognition activation area.**

2. **Function Selection:** When the tool gesture and motion recognition hybrid HCI is activated, the laparoscope-holding robot is stopped and four function buttons are overlaid on the camera image. The user can now use the grasper to choose one of the functions by performing an open grasper tool gesture within the areas of the buttons. When an open grasper gesture is recognized, the system checks if the centre of the tool gesture blob is within the areas of the function buttons. Figure 5.27 shows the ‘Display MRI’ function being selected. A message pops up in the middle of the camera image asking for user confirmation. Similarly, Figure 5.28, Figure 5.29, and Figure 5.30 shows the ‘Display XRAY’, ‘Display CAT’ and ‘EXIT’ functions being selected.
Figure 5.27: The 'Display MRI' function being selected with an open grasper tool gesture.

Figure 5.28: The 'Display XRAY' function being selected with an open grasper tool gesture.
Figure 5.29: The 'Display CAT' function being selected with an open grasper tool gesture.

Figure 5.30: The 'EXIT' function being selected with an open grasper tool gesture.
3. Function Selection Confirmation: To confirm the function selection, the user has to perform a loop tool motion within the next two seconds. Upon confirmation, the system prints a confirmation message as shown in Figure 5.31 and executes the function. Any tool motions other than the loop tool motion would deny the confirmation as shown in Figure 5.32. The user can select another function again when the selected function is either executed or denied.

![Function Selection Confirmation](image)

**Figure 5.31:** The function selection being confirmed with a loop tool motion.
4. Exit: To exit the interface, the user will only need to select the ‘EXIT’
function by performing an open grasper tool gesture and then perform a
loop tool motion to confirm. The robot will resume action when the tool
gesture and motion recognition hybrid HCI is terminated.

Here is a breakdown of the time it takes to select a function with the proposed
tool gesture and motion recognition HCI. To activate the interface, the scissors have to be
stationary within the activation area for three seconds. The time it takes to select a
function with an open grasper tool gesture is equal to the sum of the time the grasper has
to be held still at the open orientation and the time it takes for the system to recognize the
tool gesture. This process can be done with a second with our current image processing
computer. To confirm a function selection, it takes two seconds to perform and recognize
the tool motion. As a result, the best scenario for activating the interface, selecting a
function, and confirming the selection through the proposed interface, excluding the time it takes to manipulate the grasper, is about six seconds. Another three seconds is required to exit the interface. In total, the surgeon can execute a function through our proposed hybrid HCI and return to the surgery within ten seconds.
CHAPTER 6

FUTURE WORK

There are a number of alternative methods that can be considered to enhance the performance of the laparoscopic surgery robotic system. To improve the laparoscope positioning accuracy, it has been concluded that the laparoscope-holding robot has to be rebuilt. In addition, Sections 6.1, 6.2, and 6.3 in this chapter highlight a few suggestions in each of the three areas: the obstacle avoidance function, the surgical tool gesture recognition function, and the surgical tool motion recognition function. Section 6.4 highlights a couple of recommendations for the hybrid HCI using the surgical tool gesture and motion recognition functions. Section 6.5 suggests the idea of estimating the distance between the laparoscope and the surgical tools using the shadows of the surgical tools. A few recommendations are also made in Section 6.6 for future field study and testing.

6.1 The Obstacle Avoidance Function

To improve the performance of the obstacle avoidance function, the camera system has to be upgraded to a coloured system. The position marker, surgical tool tip, and obstacle tool markers have to be of different colours to prevent misidentifications. The obstacle tool markers and obstacle tool model also need to be redesigned to avoid the malfunctioning scenarios described at the end of Chapter 3. One possible new obstacle tool marker design is shown in Figure 6.1. The new obstacle markers are of similar size.
as the position marker. They can be distinguished from the position marker by colour instead of blob size, which eliminates blob misidentification problems.

The new obstacle tool model consists of a series of smaller quadrilateral instead of one big quadrilateral. This design has a major advantage over the old design. The $z$ coordinate estimation of the obstacle tool is greatly affected by the pitch angle of the tool. If the obstacle tool has a large pitch angle, the average estimated $z$ coordinate used would not be able to represent the different $z$ coordinates at different parts of the obstacle tool. The new obstacle tool model with the new thinner obstacle tool markers would be able to represent the different $z$ coordinates ($z_1$ to $z_7$) of different parts of the obstacle tool as illustrated in Figure 6.1. As a result, the program can more accurately determine if path planning is necessary to avoid potential collisions depending on which part of the obstacle tool the surgical grasper crosses over.

Figure 6.1: Using smaller obstacle markers with a different colour to represent the $z$ coordinates at different parts of the obstacle tool.
6.2 The Surgical Tool Gesture Recognition Function

There are three suggested methods to improve the current surgical tool gesture recognition. The first way is to extract or compute additional feature quantities that are more distinctive for the overlapping double-tool gestures. Distinctive feature quantities would increase the chance of successfully recognizing the overlapping double-tool gestures. The second way is to revise the surgical tool gesture classifier. Different neural network architectures could be experimented as there are dozens of different neural network architectures available. In addition, the number of layers, number of neurons, number of training iterations, learning rate, and feedback or not, etc. can all be adjusted to improve the surgical tool gesture recognition outcome. The current rule of classifying the tool gesture as the one corresponding to the largest number in the output vector can also be changed. For example, in case there are two very close numbers, an additional comparison can be carried out to classify between the two corresponding tool gestures.

Using other types of classifier such as fuzzy-rule based classifier, or some kind of hybrid classifiers is another alternative.

Another approach to improve the surgical tool gesture recognition function is using the active contour. With this approach, a contour of the target tool tip similar to the one shown in Figure 6.2 is determined and traced. While tracing the contour, the changes in tangent lines are recorded. Depending on how distinctive the changes are, they could be used independently to classify surgical tool gestures, or they could be used as one of the additional feature quantities that are fed to the surgical tool gesture classification neural network.
6.3 The Surgical Tool Motion Recognition Function

Two of the alternative approaches in motion recognition are optical flow and accumulative difference image. These two concepts are briefly summarized and experimented with our laparoscopic surgical tools in this section.

6.3.1 Optical Flow

Optical flow is a concept for estimating the motion of objects within a visual representation. The motion is typically represented as vectors originating or terminating at pixels in a digital image sequence [2]. Optical flow (OF), first published by Horn and Schunck [58], is a well-established method for calculating the velocity field $(u(x, y), v(x, y))$ of the apparent 2D motion of pixels in a dynamic image $I(x, y, t)$ due to the 3D motion of imaged objects, by examining the spatial and temporal changes in intensity values. Classical OF is based on two main constraints. The first states that the brightness of any object point is constant over time. This can be written as:

$$I(x + dx, y + dy, t + dt) = I(x, y, t)$$

(6.1)

Using the Taylor series expansion and neglecting higher order terms give the first OF constraint equation:
\[ I_s u + I_s v + I_t = 0 \]  

(6.2)

where \( u = \frac{df}{dt} \), \( v = \frac{df}{dt} \) are the desired velocity field components, \( I_s \) and \( I_t \) are the spatial image derivatives, and \( I_t \) is the temporal image derivative. Equation (6.1) by itself is insufficient to calculate \((u,v)\); hence, a second constraint, the velocity field smoothness constraint, is introduced. The velocity field can now be calculated as what best satisfies both constraints by minimizing the following square error function

\[ \xi^2(x,y) = (I_s u + I_s v + I_t)^2 + \lambda (u_x^2 + u_y^2 + v_x^2 + v_y^2) \]  

(6.3)

where \( \lambda \) is a Lagrange multiplier. The following iterative algorithm detailed in Horn and Schunck [58] is used to find the velocity field

Initialize: \( u(x,y) = v(x,y) = 0 \) for all \( x,y \)

do: \( u = \overline{u} - I_s \frac{P}{D} \), \( v = \overline{v} - I_s \frac{P}{D} \) while \( \xi^2(x,y) > \varepsilon \)  

(6.4)

where \( P = I_s u + I_s v + I_t \), \( D = \lambda^2 + I_s^2 + I_t^2 \) and \( \varepsilon \) is a small number.

The optical flow Matlab code attached in Appendix E is tested with two scenarios. The first is with a pure black background. The two consecutive frames are shown in the Figure 6.3a) and Figure 6.3b). The second scenario is inside the plastic stomach model. The two consecutive frames are shown in Figure 6.3c) and Figure 6.3d). In both cases, the surgical grasper is moving from left to right.

The resulting optical flow images on the position marker are shown in Figure 6.4. Figure 6.4a) shows the results of the pure black background scenario. Figure 6.4b) shows the results of the plastic stomach model background scenario. In both images, it is obvious that the grasper is moving to the right as the vectors generally point to the right.
However, in Figure 6.4b) where the background is not pure black, the vector field is messier, and there are some small vectors generated by the background objects. Therefore, it is recommended that the position marker needs to be segmented from the image before calculating the optical flow vector field so that the resulting image will look cleaner as shown in Figure 6.4a).

Figure 6.3: The two consecutive frames for both optical flow scenarios.
6.3.2 Accumulative Difference Image

The idea of accumulative difference image (ADI) is to perform a pixel-wise comparison of a reference frame, where only permanent stationary objects are present, and a moving frame, where moving objects are present. The changes are accumulated to extract object motions. As described by Gonzalez and Woods [50], this is done by creating a new image—within the same dimensions as those in the sequence being examined—in which each pixel value is zero. Furthermore, the first image in the sequence will be designated as the reference image, and each subsequent frame will be compared to it. Each time a difference is found, the pixel value in the corresponding location of the ADI will be incremented. Therefore, at any given time a pixel in the ADI stores the number of frames that, when compared to the reference image, were found to differ at that particular pixel location. Different types of ADI’s may be defined by different equations. The following three are of particular interest and are labeled as positive, negative, and absolute ADI’s.
\[ P_i(x, y) = \begin{cases} P_{i-1}(x, y) + 1 & \text{if } R(x, y) - f_i(x, y) > T \\ P_{i-1}(x, y) & \text{otherwise} \end{cases} \quad (6.5) \]

\[ N_i(x, y) = \begin{cases} N_{i-1}(x, y) + 1 & \text{if } R(x, y) - f_i(x, y) < -T \\ N_{i-1}(x, y) & \text{otherwise} \end{cases} \quad (6.6) \]

\[ A_i(x, y) = \begin{cases} A_{i-1}(x, y) + 1 & \text{if } |R(x, y) - f_i(x, y)| > T \\ A_{i-1}(x, y) & \text{otherwise} \end{cases} \quad (6.7) \]

where \( f_i \) is the \( i \)th frame being compared to the chosen reference frame \( R \), \( T \) is a threshold value, \( P_i, N_i, \) and \( A_i \) are the accumulative difference images after \( i \) frames have been compared with the reference, and where \((x, y)\) indicates the corresponding pixel location in each image. Please refer to Gonzalez and Woods [50] for more details.

The consecutive frames used to generate the accumulative difference images are shown in Figure 6.5. Figure 6.5a), which contains the first three rows of Figure 6.5, shows the frames of the black background scenario. The order is from left to right and top to bottom. Figure 6.5b) shows the frames inside the plastic stomach model.
Figure 6.5: The consecutive frames for both ADI scenarios.

The corresponding three different types of ADI’s generated by the Matlab code attached in Appendix E are shown in Figure 6.6. The positive, negative, and absolute ADI’s for the black background scenario are shown in Figure 6.6a), Figure 6.6c), and Figure 6.6e). The figures show that absolute ADI is the most effective method in capturing the surgical grasper movements. The ADI’s for the plastic stomach background scenario are shown in Figure 6.6b), Figure 6.6d), and Figure 6.6f). In this case, the bright background pixels have similar values as the target tool pixels. Consequently, the ADI’s
do not show the surgical grasper movements as well as in the black background case. Thus, it is recommended to segment the surgical grasper tool first before applying absolute ADI to achieve the best results.

Figure 6.6: Positive, negative, and absolute ADI's of the two scenarios.
6.4 The Hybrid HCI

To execute more functions, the number of function buttons can be increased. However, to maintain the recognition rates of the tool gesture and tool motion, the grasper cannot be positioned too far below the laparoscope. In other words, the grasper has to be kept close to the laparoscope and it has to appear as large as possible on the screen. Therefore, the function buttons cannot be too small; otherwise, the grasper has to be positioned further away from the laparoscope in order to perform the open grasper tool gesture within the button area, which may lower the tool gesture and tool motion recognition rates. Six function buttons would probably be able to fit within the camera image with reasonably large button area. Two of the buttons can be used to flip forward and backward between different pages of functions while the other four buttons can take on different functions on different pages. With this approach, the number of executable functions through the interface is virtually unlimited. The first page could contain frequently accessed functions such as displaying medical images. Subsequent pages could each contain functions for a certain class of tasks.

With a faster image processing computer, the time it takes to select and confirm a function can be shortened. The function selection and confirmation process could cut down to within a second. As a result, the time it takes to activate the interface, select and confirm a function selection on the first page, and exit the interface could be shortened from ten seconds to five seconds.
6.5 Distance Estimation using Shadow

An alternative method to estimate the distance between the laparoscope and the surgical tools, or the $z$ coordinate, using the pinhole model of the camera, as it is currently implemented in laptrack, is to use the information contained in the shadows of the surgical tools. Irvin and McKeown [59] describe a procedure to estimate building heights using information provided by cast shadows. If implemented, the shadow method and the pinhole camera method could then be compared for accuracies in estimating the distance between the laparoscope and the surgical tools.

6.6 Field Study and Testing

Conducting a field study to see how effective our system is in assisting surgeons in real operation is a key step in the future development of this laparoscopic surgery robotic system. We can start by asking surgeons to try out the various functions of the system and gather their feedbacks, which could help us determine whether a certain function is effective, helpful but needs to be improved, or redundant. Their feedbacks might also stimulate new ideas that we have never thought of.

Running a field testing with videos of actual operations would also be beneficial, especially for testing the surgical tool gesture and motion recognition functions. However, the camera equipment has to be upgraded to output colour images, and the image processing program laptrack has to be modified to process colour images. The surgical tool tracking, tool gesture recognition, and tool motion recognition functions will also have to be altered. This field test would provide insights on how well those two functions work in real surgical environment, and if the addition or removal of certain surgical tool gestures or surgical tool motions would be necessary.
CHAPTER 7

CONCLUSION

The work presented in this thesis project is a continuous development on the automatic tool tracking, surgical tool gesture and surgical tool motion recognition laparoscopic surgery robotic system integrated in my Bachelor thesis project. With telesurgery in mind, the system is developed in such way that the image processing computer and the robot control computer communicates through a TCP/IP network connection. The laptrack program running on the image processing computer is responsible for automatically tracking a surgical grasper, estimating the 3D coordinates of the tool position marker, and sending the coordinates over to the robot control computer through the network connection. Upon receiving the set of new tool position coordinates, the demo1 program running on the robot control computer goes through the inverse kinematics calculations of the laparoscope holding robot, and commands the robot to position the laparoscope such that the surgical grasper is always at the center of the field of view. The novel design of the spherical parallel mechanism robot occupies less space than existing robot arms. It could replace the camera holding assistant, and provide precise, dexterous, steady, and instant camera movements for the entire duration of operation.

The system is also capable of recognizing a number of single- and double-tool gestures and tool motions, making available a convenient and intuitive surgeon-system interface. During operations, if the surgeon would like the system to execute certain
commands, such as displaying the patient’s CT scan on the monitor, he or she could use the surgical tools already in his or her hands to perform gestures and motions as “words” to form command “sentences”. The system would recognize them and execute the commands immediately. This system is highly promising as a standard surgical assistance device in the near future.

Chapter 2 investigates modeling and control of the laparoscope-holding robot. It starts with a brief system overview. The kinematics modeling and inverse kinematics of the robot are then introduced, followed by the robot control program modifications. In the last section, a number of the major components of the system, including the TCP/IP network connection, robot control module, robot control program, and the laparoscope-holding robot are closely examined to determine the source of the laparoscope positioning errors. It is concluded that the laparoscope-holding robot was built with major defects that contribute to the majority of the laparoscope positioning errors.

Chapter 3 describes the improvements made to the surgical tool tracking program laptrack. The first improvement is switching from the white background to black background, and replacing the old mock-up stick with a real surgical grasper. The program can now track the real surgical grasper inside a plastic stomach model that better mimics the real surgical environment. A conversion factor is added to convert the estimated x and y coordinates of the position marker on the surgical grasper from pixel to millimeters.

When the surgical grasper is placed within 30 mm to 70 mm below the laparoscope, testing results show that the average error for the z-coordinate estimation developed by Zhang [32] [33] is 4.5%, which is excellent. The estimation errors for the x
and y coordinates are around 2.8% to 24.5%. However, in reality, the testing results show that the x or y coordinates are only off by one or two millimetres, which means the surgical grasper would still be at the centre of the field of view. Overall, the performances of the coordination estimation and conversion function are satisfactory. An obstacle avoidance function is added in case another surgical tool is in the way of the laparoscope while the robot is moving. If potential collisions between the laparoscope and the obstacle tool are detected, the program would generate one to two extra path points to direct the robot to avoid such collisions. However, due to the limitations of the grayscale camera system, the performance of the obstacle avoidance function is not satisfactory.

Chapter 4 details the surgical tool gesture recognition function. The function is capable of recognizing eleven single- and double-tool gestures in total. It utilizes morphological operations and thresholding to segment the target tool blob. Two target tool segmentation methods are compared. One is the background reduction method and modified background reduction with thresholding method. The other is the simple thresholding method. The thresholding method is chosen due to its significantly shorter processing time. However, the thresholding method has its limitations. It only works within a certain range of brightness. It also cannot remove background bright spots that are as bright or brighter than the target tool. After the target tool is segmented, feature quantities are extracted from the target tool blob and fed to a feed forward neural network for tool gesture classification. Neural network's ability to approximate nonlinear relationships helps in covering for the errors accumulated from the previous stages.
Testing results show that the single tool gestures, except the open scissors gesture, can be recognized with high recognition rates, and so do the non-overlapping double-tool gestures. The open scissors gesture has to be performed with a roll angle to achieve high recognition rates. For the overlapping double-tool gestures, the recognition rates for the cross and triangle gestures are around or below 50%, and the double triangle gesture is unrecognizable. Overall, the surgical tool gesture recognition function could be a reliable and practical surgeon-computer interface in future laparoscopic surgery assistance systems if the recognition rates for the overlapping double-tool gestures could be improved.

Chapter 5 presents the surgical tool motion recognition function that is capable of recognizing four surgical tool motions, including line segment, two line segments and an angle, curve, and loop. The system collects sample points along the motion trajectory of the position marker and computes the displacement vectors. One of the four discrete orientations is then assigned to each displacement vector before the motion classification algorithm is applied to classify the motion pattern.

Testing results show that the four motion patterns have recognition rates ranging from 36% to 50%. Those recognition rates are too low for the surgical tool motion recognition function to be considered reliable and practical in real life surgeon-computer interface applications. However, the high recognition rates achieved by testing the surgical tool motion recognition function outside the plastic model suggest that the performance of this function is also greatly affected by the surgical environment and user skills. Improved user skills and an advanced surgical tool motion algorithm could
enhance the performance of this function, making it a worthy compliment to the surgical tool gesture recognition function.

A hybrid HCI using a combination of the surgical tool gesture and surgical tool motion recognition functions is proposed. The interface is activated by keeping the scissors within an activation area for three seconds. After the interface is activated, the user can use the open grasper tool gesture to select one of the four functions overlaid on the camera image and confirm the selection with a loop tool motion. The system will then execute the selected function. The hybrid HCI is exited with the same selection and confirmation steps. The user can activate the interface, select and confirm a function, and exit the interface in approximately ten seconds.
APPENDICES
Appendix A – demo1 Verification Matlab Code

Created by Temei Li

clear all;
cf; format short;

%xi=0.685079; yia=0.377469; zia=-0.623044;
wxia=-0.025524; yxia=0.315824; zxia=-0.391342;
wx3a=-0.307039; y3a=-0.454637; z3a=-0.187403;

fid = fopen('.\scope-mapping-table.txt', 'wt');
alpha=90;
theta=45;
% old link=[45 37; 50 63; 35 75];
link=[25 65; 30 22; 35 48];

% xi=0.68507895327194; yia=0.3774886932547; zia=-0.62304430726345; 
link=[25 65; 30 22; 35 48];

% xia=0.02524297159395; y2a=-0.21989409880578; z2a=-
% 0.39134188393594; 
% xia=0.025242397159395; y2a=-0.21989409880578; z2a=- 
% 0.39134188393594; 
% xia=0.025242397159395; y2a=-0.21989409880578; z2a=- 
% 0.39134188393594; 
% x3a=-0.97073853481007; y3a=-0.45463880726991; z3a=-0.18740271466030;

alpha11=25;
alpha12=65;
alpha21=50;
alpha22=22;
alpha31=35;
alpha32=48;
theta1_home=(-80.579010290+63.75592619);
theta2_home=(-80.579010290+163.14273909);
theta3_home=(-80.579010290+36.17086101);
theta3m_home=55.8008346;
theta3m_home=-(180-69.5580977);

v1=[];
w1=[];
w2=[];
w3=[];
x1=0; y1=0; z1=0;
x2=zeros(3,1);
x3=zeros(1,3);
x4=eye(4);
x5=eye(4);

tx=1*pi/12;
ct = cos(tz);
st = sin(tz);
rz = [ct -st 0;
st ct 0; 0 0 1];
m1z=[.4082482905 0 0 .4082482905 -.8164965809 ;
.7071067812 -.7071067812 0 ;
-.5773502692 -.5773502692 -.5773502692 ];
mlz=rz*mlzO;
size(ml);
erasemode='xor';
wrist=1;
repeat=1;
base=0;

% lfs=[1 2 6 5;2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];
%hdll=patch('faces',lfs,'vertices',[lll(1,:); lli(2,:); llli(3,:)],'FaceColor','C');
figure(1);
Rset(gcf, 'renderer','zbuffer');
grid on;
view([0,0,5]);
view([10,10,10]);
title('Motion of 3 DOF RRR second tool holder platform');
xlabel('X');
ylabel('Y');
zlabel('Z');
axis('equal');
axis('square');
axis([-1 1 -1 1 -1 1]);
reach=l;
line('xdata', [0;0], 'ydata', [0;0], 'color', 'g');
view([1.1,1.1,1.1]);
rotate3d on;
line('xdata', [0;0], 'ydata', [0;0], 'zdata', [-reach;base],
'color', 'g');
view([0.12,0.125,0.125]);
%create a line which we will
%subsequently modify. Set erase mode to xor for fast
%update
% hr = line('color', 'r', 'erasemode', 'erasemode);
% if wrist,
% hr = line('xdata', [0;0], 'ydata', [0;0], 'zdata', [0;base], ...
'color', 'red', 'erasemode', 'xor');
% hy = line('xdata', [0;0], 'ydata', [0;0], 'zdata', [0;base], ...
'color', 'green', 'erasemode', 'xor');
% hz = line('xdata', [0;0], 'ydata', [0;0], 'zdata', [0;base], ...
'color', 'blue', 'erasemode', 'xor');
end

202
% create a line which we will
% subsequently modify. Set erase mode to xor for fast
% update
%
hr = line('color', 'r', 'erasemode', 'erasemode);

motor_axes=[1,-1/2,-1/2;0,3,.5/2,-3.5/2;0,0,0,0];

% new start vector for B R G branch

%draw the platform
lfs=[1 2 6 5;2 3 7 6;4 8 3 4;1 5 0;1 2 3 4;5 6 7 8];

13i(1:1,1:3)=ml(1:3,1:3);
13i(1,4)=ml(1,1)+.01;
13i(1,5:7)=ml(1:3,1:3)+.05;
13i(1,8)=ml(1,1)-.01;
13i(2:3,8)=ml(2:3,1);

hdls=patch('faces',lfs,['vertices',13i(1,:),13i(2,:)]);
13i(3,:)=[';','FaceColor','r']);

drawnow

% hold on;

%new platform position
set(hdls,'vertices',[13n(1,:)',13n(2,:)',13n(3,:)'])]

ml4t=[ml(L1:) 0; ml(2,:) 0; ml(3,:) 0; 0 0 0 1];

d1van=tlxrn (ml4 t)

% draw the moving platform (no erase yet)

for i=1:1:?
ml4t=x2trn([d1van(l:7); (d1van (8)-(pi*i/18)) I);]

13i(1:3,1:3)=ml4t(l:3,1:3);
13i(1,4)=ml4t(1,1)-.001;
13i(2:3,4)=ml4t(2:3,1);
13i(1:3,5:7)=ml4t(1:3,1:3)+.001;
13i(1,8)=ml4t(1,1)-.001;
13i(2:3,8)=ml4t(2:3,1);

hdls=patch('faces',lfs,['vertices',13i(1,:),13i(2,:)]);
13i(3,:)=[';','FaceColor','y']);

set(hdls,'vertices',[13n(1,:)',13n(2,:)',13n(3,:)'])]

set(thx,'xdata',[0 13i(3,1)]);

set(thz,'xdata',[0 13i(3,1)]);

set(thz,'xdata',[0 13i(3,3)]);

set(thz,'xdata',[0 13i(3,3)]);

drawnow
hold on;

and
aframe = rt3dh(0, 135, 0, eye(4));
vlh = aframe = rt3dh(0, -1*alpha2, 0, theta1_home, eye(4));
wlh = aframe = rt3dh(0, -1*alpha22, 0, theta1_home, eye(4));
w2h = aframe = rt3dh(0, -1*alpha3, 0, theta2m_home, eye(4));
w3h = aframe = rt3dh(0, -1*alpha3, 0, theta3m_home, eye(4));
vlh = vlh * rt3dh(0, 90, 0, -160.75, eye(4));
vlh = (vlh(1, 3), vlh(2, 3), vlh(3, 3));
dot(vlh(1, 3), vlh(2, 3));
cross(vlh(1, 3), vlh(2, 3));

old R_h1 = [vlh(1:3, 3), v2h(1:3, 3), v3h(1:3, 3)];
R_h1 = [vlh(1:3, 3), v2h(1:3, 3), v3h(1:3, 3)];

for k = kmin:kint:kmax
    for j = jmin:jint:jmax
        for i = imin:iint:imax
            imin = 0;
imax = 30;
ilint = 2;
jint = 10;
kmin = 0;
kmax = 0;
kint = 1;
count = (imax-imin)/lint;
jcount = (imax-jmin)/jint;
kcount = (kmax-kmin)/kint;
jk = 1;
thetaxy = 0;
i = 1;
j = 1;
k = 1;
xcount = 1;
ycount = 1;
for v = v3h(1:3, 3)
    delh = 1;
    tcount = 0;
    for kwmin = kmin:kekmax
        for ji = jmi:jint:jmax
            for ik = imi:iint:imax
                for k = kmin:kint:kmax
                    for j = jmin:jint:jmax
                        for i = imin:iint:imax
                            ...

204
\begin{verbatim}
count=count+1;
\end{verbatim}
if m==l
0.
O

wl=[wl;xll,yll, 2111;
wll=[w11;x12,y12,z12];
cE = EprintE(Eid,'\n 212.hE %12.6f %12.6f 212.6E

"d2.6f
o
%12.6E "612.6E %12.6f 912.6E %12.6E %12.6f
%12.6f1,i,j,k,vscope(lj , v s c o p e 2 v c o p e ( 3 ) ,xll,yll, 211) ;
cf = fprintf(fidI1 \n middle joint vector vl xl yl
zlCb12.6f %12.6E %12.6f ',xll,yll, zll) ;
,
cf = fprintf(fid,'\n X12.6f Z12.6E %12.6f %12.6f
2.12.6f 212.6E Yr12.6E %12.6f 9d12.6f' , i, j, k,x12, y12, 212) ;

Z (rpyhome(1)+ixpi/180), (rpyhome (2)+jxpi/180), (rpyhome (3)+k*pi/l80j

elseif m==2
w2=[w2;x11, yll, 2111;
w21=[w%l;x12, y12, z12] ;
cE = Epriritf (fid,'\n middle joint vector v2 x2 y2 22%12.6E
%12.6f %12.6f1,xLl,yll, zllj;
%
cf = Eprir1t.f(Eid, 'B12.6E %12.6[ R12. 6 f 1 ,xll, y11, zll) ;
else

,


w3=[w3:x11,y11,z11];
w31=[w31:x12,y12,z12];
cf = fprintf(fid,'middle joint vector v3 x3 y3 z3%12.6f\n%12.6f%12.6f\n',x11,y11,z11);
cf = fprintf(fid,'middle joint vector 2 v3 x3 y3 z3%12.6f%12.6f%12.6f\n',x12,y12,z12);
end
end %m
end %l
end %j
end %k

w=[w1(:,1),w1(:,2),w1(:,3),w2(:,1),w2(:,2),w2(:,3),w3(:,1),w3(:,2),w3(:,3)];
w=[w1(:,1),w1(:,2),w1(:,3),w2(:,1),w2(:,2),w2(:,3),w3(:,1),w3(:,2),w3(:,3)];
fclose(fid);
if tcount>=2
for i=2:tcount
    zltmpold=w(i-1,3);
z2trnpold=w(i-1,6);
z3tmpold=w(i-1,9);
zltmp=w(i,3);
z2tmp=w(i,6);
z3tmp=w(i,9);
    if zltmp-zltmpold < 0
c1(i)=-1;
else
c1(i)=1;
end
    if w(i,4) > 0
if z2tmp-z2tmpold < 0
c2(i)=1;
else
c2(i)=-1;
end
else if z2tmp-z2tmpold < 0
c2(i)=-1;
else
c2(i)=1;
end
    if z3tmp-z3tmpold < 0
c3(i)=-1;
else
c3(i)=1;
end
    if zll-z3a < 0
c3(i)=1;
else
c3(i)=-1;
end
end
end
end
end
\[
t_1 = \cos(\text{dot}([w(i,1), w(i,2), w(i,3)], [xla,yla,zla])); \\
t_2 = \cos(\text{dot}([w(i,4), w(i,5), w(i,6)], [xla,yla,zla])); \\
t_3 = \cos(\text{dot}([w(i,7), w(i,8), w(i,9)], [xla,yla,zla])); \\
\]

\[
t_1 = \cos(\text{dot}([w(i,1), w(i,2), w(i,3)], [w(i-1,1), w(i-1,2), w(i-1,3)])); \\
t_2 = \cos(\text{dot}([w(i,4), w(i,5), w(i,6)], [w(i-1,4), w(i-1,5), w(i-1,6)])); \\
t_3 = \cos(\text{dot}([w(i,7), w(i,8), w(i,9)], [w(i-1,7), w(i-1,8), w(i-1,9)])); \\
\]

deltang = [deltang; (t1*c1(i)), (t2*c3(i)), (t3*c3(i))];

end

motorang3 = \cos(\text{dot}([w(tcount,7), w(tcount,8), w(tcount,9)], [xla,yla,zla]));

motorang3 = \cos(\text{dot}([w(tcount,7), w(tcount,8), w(tcount,9)], [xla,yla,zla]));

end

motorang3 = \cos(\text{dot}([w(tcount,7), w(tcount,8), w(tcount,9)], [xla,yla,zla]));

\%
cf = fprintf(fid, ' \nn\n motor rotate angle 1 2 3 %12.6f %12.6f
12.6f\', motorang1, motorang2, motorang3));
\%
cf = fprintf(fid, ' \nn\n motor rotate angle 1 2 3 %12.6f %12.6f\', motorang1, motorang2, motorang3));

close(fid); %
plot3(vscopeall(:,1), vscopeall(:,2), vscopeall(:,3), 'k');
plot3(w(:,1), w(:,2), w(:,3), 'b');
plot3(w(:,1), w2(:,2), w2(:,3), 'r');
plot3(w(:,1), w3(:,2), w3(:,3), 'g');
plot3(0, 0, 1, 'k*'); NCLID on;

212
Appendix B – Camera Calibration, Lens Distortion Correction and Tool Tracking

The following are excerpted from Chapter 3 of Zhang [32]. To realize automatic image tracking in Laparoscopic surgery, several important issues need to be solved. Among them are marker design, endoscopic camera calibration, endoscopic image distortion correction, tracking methods selection and scope holder design. The proposed solutions for these tasks are presented in this chapter. (Before solving these problems, proper image processing techniques need to be selected for our application.)

In this chapter, first the image processing techniques used in our research are presented. Second, procedures needed to achieve automated instrument tracking, such as, marker design, camera calibration, endoscopic image distortion correction are discussed and proposed the solutions. Finally, the scope holder design and simulation are explicated.

3.1 Image Processing techniques used in our Image Tracking system

Since computers can process only digital images, and physical images were in non-discrete forms, a prerequisite for digital image processing is the conversion of images into digital form: digitization. Digitalization comprises three steps: a) Scanning or digitizing, is the process of sequentially addressing small area on the film. b) Sampling, it means measuring the gray level of an image at each pixel location. c) Quantization is the representation of a measured value by an integer[20]. These three steps are sufficient to generate a numerical representation of an image.
The digitized image can be represented by a two dimensional gray level matrix, for each pixel located at \((u, v)\), its gray level is \(f_{u,v}\). For an image \([F]\) with \(n \times m\) pixels.

\[
[F] = \begin{bmatrix}
  f_{0,0} & f_{0,1} & \ldots & f_{0,m-1} \\
  f_{1,0} & f_{1,1} & \ldots & f_{1,m-1} \\
  \vdots & \vdots & \ddots & \vdots \\
  f_{n-1,0} & f_{n-1,1} & \ldots & f_{n-1,m-1}
\end{bmatrix}
\]

(3.1)

Thus, applying different operations to matrix \([F]\) enable us to process the image in different image processing methods. Gray-scale resolution is defined as the number of gray levels per unit of measure of image gray amplitude. Storing a digital image in 8 bits bytes, for example, yields a 256-level gray scale. Generally larger size of image have higher gray-scale resolution and clearer image than small size image. But at the same time the processing time will also increase. Considering both processing speed and image quality, the default image size comes with Matrox Imaging Library is 640 \(\times\) 480 with 8 bit gray-scale resolution in our system.

An efficient data structure is also a key factor that influence the image processing speed. The common used data structure for image processing include two dimensional and one dimensional matrix. Though the two dimensional matrix data structure is a popular one due to its clear and direct expression. However, two dimensional matrix require fixed memory allocation, while using one dimensional matrix in the pointer mode enable us to use dynamic allocated memory. A point \((u, v)\) at a \(m\) row \(n\) column two dimensional matrix can be represented as \((v - 1) \times n + u\) in a one dimensional matrix.

Using pointers to represent this one dimensional matrix, we can allocate, delete, resize memory as needed.
Most of the image processing tasks in this application can be handled by Matrox Imaging Library. But for the endoscope image distortion correction method, an algorithm is developed for handling image processing tasks. Image data are stored in the one dimensional matrix data structure described above for efficient manipulation.

Recently there has been an increase in robotic applications that require image tracking of one sort or another. Many researchers proposed different techniques for visual tracking tasks. As stated in the introduction, techniques used for visual tracking include optical flow, feature point correspondence, edge or contour correspondence, active contour models, and model-based matching. For our particular application in laparoscopic surgery, due to the diversity of the instruments and different shapes of the same instrument if inserted into the camera field of view in different angles, model-based pattern matching is not feasible. In this thesis, we proposed to attach uniform markers to the tip of the instruments and using image segmentation method to locate the marker in successive image frames. Matrox Imaging Library provides an operation called blob analysis. This operation fits for our application well. The first step for any imaging system would be camera calibration.

3.2 Endoscopic Camera Calibration

Defining the pixel-to-world mapping is known as camera calibration. Once the camera is being calibrated, we can transform the image pixel coordinates to their real-world equivalents. Camera calibration is a critical first step in many applications such as dimensional measurement of mechanical parts, tracking, camera-on-robot configuration and robot vehicle guidance. Camera calibration has been investigated extensively by
many researchers, e.g. [21], [22]. But methods for camera calibration using rigid endoscope have not been addressed before.

In general, endoscopic images have particular characteristics. The endoscope view angle, typically between 65° - 75°, as shown in Chapter 2 Figure 2.2, is narrow compared to the wide field of view of human eyes. Thus, vision is limited by the narrow endoscope filed of view. To enlarge the filed of view, generally wide-angle lens are used in endoscope. However wide-angle lens intrinsically has barrel distortion (transformation of straight line into curves). Barrel distortion cause imaged position to be distorted along radial lines from the image center, this type of distortion is non-linear: image areas farther away from the center appear significantly smaller. In this paper we proposed a calibration method that assumes the image located within small center area are distortion free.

Typical parameters used for calibration can be classified into two classes: extrinsic and intrinsic parameters. This section presents methods for determining intrinsic parameters, namely, the effective focal length, the real image center and the scale factor. The extrinsic parameters: rigid body transformation from the world coordinate system to the camera coordinate system will be discussed later.
Figure 3.1 illustrates the basic geometry of the camera model. \((x, y, z)\) is the world coordinate system. In robotic system, this coordinate frame can be placed at the robot base. \((x_c, y_c, z_c)\) is the camera coordinate system, \((x, y, z)\) is the 3D coordinates of an object \(P\) in the world coordinate system, \((x_c, y_c, z_c)\) is the coordinate of point \(P\) in the camera coordinate system with the \(z_c\) axis the same as the optical axis. Here the camera coordinate frame is defined at the tip of the endoscope. \((x_c, y_c)\) (not shown in Figure 3.1) is the image coordinates of \(P(x_c, y_c, z_c)\) projected into the image plane if a perfect pinhole model assumption is used [23]. \((P', V')\) is the coordinates used in the image buffer. Additional scale factors need to be specified that relates the image coordinate in the front image plane to the image buffer coordinate system.

The focal length \(f\) is the distance between front image plane center \(o\), and the optical center \(o\). Here we define the image center as the optical axis passing through the image plane. In theory, the image center should be the center of the image plane. But in
practice, especially in the endoscope image, the real image center is distorted due to the imperfect grind of endoscope lens.

In the following, first we propose a simple and fast method to obtain the real image center, then a method for calculating the calibration parameters such as the effective focal length and the scale factor. Figure 3.2 shows the calibration flow chart.

The first problem is how to get the real image center. Since abdominal cavity is dark, as described in chapter 2, an external light source is used (See Chapter 2, Figure 2.2). A fiber optic cable is used to transmit light to the side of the endoscope. The light source is evenly distributed through the rod lens. If we point the endoscope perpendicular to a white background, light source forms a white circle in the image as shown in Fig 3.3. (a). Inspired by this feature, we find a simple method of getting the real image center is
letting the endoscope point to a white background and record the image. The center of the white circle in the image is defined as the real image center. The result is shown in Fig 3.3. (b). Other work, such as [24] consider the distortion in endoscope image and proposed to use straight lines patterns and choose the intersection of lines which remain straight in the endoscope image as the real image center which needs complex calculations. The feasibility of the proposed method for determining the real image center will be evaluated later in the experiment result section.

Figure 3.3. Endoscopic image (a) original gray scale image, (b) binary image (the image center has been marked)

The second parameter to be calibrated is the real focal length. As shown in Figure 3.1, transforming from 3D camera coordinate \((x, y, z)\) to image coordinate \((v, f, v)\) based on the perspective projection with pinhole camera geometry, we can get the following equation:

\[
\begin{align*}
\frac{oo}{z_c} &= \frac{f}{z_c} = \frac{(v - v_m)s_x}{x_c} = \frac{(v - v_m)s_x}{y_c} \\
\frac{x_c}{z_c} &= \frac{z(v - v_m)s_x}{f}, \quad \frac{y_c}{z_c} = \frac{z(v - v_m)s_y}{f}
\end{align*}
\]
where $oo_j$ is the distance between the center of the image buffer to the center of the camera coordinate frame, $(V_{oo},V_{oo})$ are rows and column numbers of the image center, $S_x,S_y$ are the scale factors that map the real image coordinate $(x,y)$ to image buffer coordinate $(V,V')$. When $V',V',x,y$ are known parameters the effective focal length can be calculated from equation (3.2). Also, when $f,V',V'$ and $x,y$ are known parameters, we can get the depth information $Z$ of an object in the field of view.

Scale factors $S_x,S_y$ can be obtained using the following formula.

\begin{align}
    x_d &= (V'-V_{oo})S_x \\
    y_d &= (V'-V_{oo})S_y
\end{align}

(3.3)

In general, manufacturers of CCD cameras supply the information of center to center distance between adjacent sensor elements in the $y$ direction (i.e. scale factor $S_y$) as a fixed value: $S_y = N_{y} / d_{y}$, where $N_{y}$ is the number of pixels in the $y$ direction; $d_{y}$ is the dimension of CCD in the $y$ direction. Scale factor $S_x$ is an uncertain value due to various reasons, such as slight hardware timing mismatch between image acquisition hardware and camera scanning hardware, or the imprecision of the timing of TV scanning itself. Here we propose a simple and fast method to obtain the relationship between $S_x$ and $S_y$.

We use the endoscope to capture an image of known dimension (e.g. square grids), then compute the center to center distances of the squares $D_x,D_y$, as shown in Figure 3.4 of pixel unit in the $x$ and $y$ direction. Using the equation: $S_x/S_y = D_x/D_y$. 

220
\[ S_r = S_x \times D_y / D_x \] (3.4)

In actual implementation, we can measure several squares’ center to center distances and use their mean values to reduce the possible error that may bring by the image processing method.

![Figure 3.4 Pattern image for Calibration](image)

### 3.2 Lens distortion and correction methods

No lens can produce perfect images. Common imperfections are aberrations that degrade the quality or sharpness of the image and lens distortions that deteriorate the geometric quality (or positional accuracy) of the image. Endoscopic image has a fundamental lens distortion due to the wide-angle design of the objective lens of endoscopes. Wide-angle lenses are used in endoscope because they provide larger viewing fields. However, lens distortion will result in erroneous measurement for image positions in the resulting images. To eliminate the distortion effect, corrections should be applied to measurements of the resulting endoscope images. Subsequent analysis of laparoscopic images, such as estimation of quantitative parameters (area and perimeter),
is of considerable importance while performing clinical endoscopy (such as tumor measurement). Unless the distortion is corrected estimation errors could be very large[25][26].

![Figure 3.5. Barrel type distortion. The left image is an undistorted grid pattern, and the right image is the same pattern viewed after radial distortion.]

Lens distortion are classified as either radial or tangential. Radial distortion, as its name implies, causes image position to be distorted along radial line from the optical axis. Radial distortion includes barrel distortion, pincushion distortion or the combination of these two types. Tangential distortion is due to imperfect centering of the lens components and other manufacturing defects in a compound lens, it is also called decentering, that is the optical center and the lenses are not strictly collinear. The pixel shift is away from the optical center and the new position lies at a new angle location as measured from the optical center. Tangential lens distortions are generally very small and are seldom corrected for [27].

To capture a larger field of view in a single laparoscopic image, wide viewing angle lens are often used in laparoscope. Thus barrel type radial distortion is common in laparoscopic images, due to wide-angle configuration of the camera lens, see Figure 3.5.
Areas further from the center of the field of view appear smaller than they really are. As the reason described above, we only consider the barrel type distortion in this paper.

Three different methods of correcting for radial distortion are.

1. Reading required corrections from a radial-lens distortion curve.
2. Interpolating corrections from a table.
3. Numerical methods in which the radial-lens distortion curve is approximated by a polynomial.

Though the last method is difficult, it is also the most convenient with a computer. Here we applied the last distortion correction method and proposed to approximate the distortion curve by a polynomial. Using the least square method to find the coefficients for this polynomial. With the experiment determined polynomial, the endoscopic image can then be undistorted accordingly.

3.2.1 Radial Lens Distortion Model

Radial lens distortion model consists of two image planes. The distorted image plane is represented by \((Y', X')\), while the distortion correction image plane is represented by \((Y, X)\). Their corresponding centers are represented by \((V_Y', V_X')\) and \((V_Y, V_X)\). The center of distortion correction image can be chosen arbitrarily. Unlike the method proposed by [28] that chose these two centers at different locations, in this paper, we select both centers the same as the real image center. That is, the straight lines that pass through this point remains straight in the distorted image plane.
Using polar coordinate with the origin at the real image center. A point $P'$ in the distorted image plane can be represented as:

$$r' = \sqrt{(V_{\theta}^r - V_{\alpha}^r)^2 + (V_{\phi}^r - V_{\phi}^r)^2}$$

$$\theta' = \arctan\left(\frac{V_{\phi}^r - V_{\phi}^r}{V_{\theta}^r - V_{\theta}^r}\right)$$

where magnitude $r'$ is the distance of $P'$ to the center $(V_{\alpha}^r, V_{\phi}^r)$. The corresponding point $P'$ in the corrected image plane is $P$. Point $P$ can be represent in polar coordinate as $(r, \theta)$. Because we assume that the distortion is pure radial, the polar angle is unchanged in the distorted and corrected image planes, hence,

$$\theta = \theta'$$

The objective of the distortion model is to find the relationship between $P$ and $P'$.

As shown in Figure 3.6. Relationship between $r$ and $r'$ is:

$$r' = r + \Delta r$$

where $\Delta r$ can be represented as an odd-ordered polynomial series [27]:

$$\Delta r = k_1 r + k_3 r^3 + k_5 r^5 + k_7 r^7 + \ldots$$

where $k_i (i = 1, 2, 3, \ldots)$ are the expansion coefficients. The reason for using odd-order degree interpolating polynomials is because it was shown that even-order degree interpolating polynomial filters are space-variant with phase distortion [29]. This is attribute to the fact that the number of sampling points on each side of the interpolated point always differ by one. As a result, interpolating polynomials of even-degree are not considers.
After obtaining the coefficients the new pixel location in the corrected image plane can be calculated as \( r' = r - \Delta r \). See Figure 3.6, notice that:

![Figure 3.6 Components of Radial Distortion](image_url)

Figure 3.6 Components of Radial Distortion. The image point \((V_i, V_j)\) is radically displaced by \( \Delta r \) to the new position \((V_i', V_j')\).

\[
\frac{r}{\Delta r} = \frac{x}{\Delta x} = \frac{y}{\Delta y} \tag{3.8}
\]

We can use the following equations to calculate the new pixel location in terms of two components \((\Delta x, \Delta y)\). These formulas fit well for computer processing.

\[
x' = x - \Delta x = x(1 - \frac{\Delta r}{r}) = x(1 - k_1 r^2 - k_2 r^4 - ...) \tag{3.9}
\]

\[
y' = y - \Delta y = y(1 - \frac{\Delta r}{r}) = y(1 - k_1 r^2 - k_2 r^4 - ...) \tag{3.9}
\]

To obtain the polynomial series for mapping distorted image to distortion correction image, first, we need to obtain the coefficients for the polynomial.
3.2.2 Coefficients Estimation

The polynomial coefficients define the shape of the curve. They can be calculated by nonlinear regression analysis, such as least square method to obtain a best curve fit to a given data set (obtained from experiments). The calibration pattern (Figure 3.7) was used for obtaining square center positions in the distorted image. Since we assume that the distortion is pure radial, the distortion is circularly symmetric. Without the loss of generality, testing squares that lie in the horizontal or vertical in the image can represent the general distortion pattern for the whole image. For a given data set \( S = [C_1, C_2, ..., C_3, C_4] \), let there be \( N \) columns of testing squares, \( C_i(r, \Delta r) \) representing \( r \): radial distance between the center of the square from the real image center and the distortion \( \Delta r \) at a radial distance \( r \). Consider the polynomial series of degree \( 2M+1 \):

\[
\Delta r_i = k_0 r_i^0 + k_1 r_i^1 + k_2 r_i^2 + ... + k_j r_i^{2j+1} + ... , \quad j = 0, ..., M \tag{3.10}
\]

The deviation of a point from its correct image position is:

\[
F_i = y_i - \sum_{j=0}^{M} k_j r_i^{2j+1}
\]

The least square problem is then to find the values of \( k_0, ..., k_j \) so as to

Minimize \( \sum_{i=1}^{k} [y_i - (\sum_{j=0}^{M} k_j r_i^{2j+1})]^2 \) \tag{3.12}

Hence, \( k_j \) can be calculated from:

\[
\frac{\partial F_i}{\partial k_j} = 0, \quad \text{for } j = 0, ..., M \tag{3.13}
\]
From equation (3.10), $M+1$ simultaneous equations are obtained, which may be represented in matrix form as:

$$AK = Y$$  \hspace{1cm} (3.14)

where

$$A = [r_i^j]_{i=0,..,M, j=0,..,W}$$

$$K = [k_0, k_1, ..., k_M]^T$$

$$Y = [y_0, y_1, ..., y_M]^T$$

Equation (3.13) can be calculated as

$$K = (A^T A)^{-1} A^T Y$$  \hspace{1cm} (3.14)

The matrix $K$ consists of most probable values for unknowns, $k_0, k_1, ..., k_M$. The detailed derivation of $K$ is shown in appendix B.

Once the expansion coefficients are computed, all pixels from the distorted image are mapped onto the corrected image. However, one problem should be paid attention to, that is the direction of mapping. One approach is mapping from the distorted image plane to corrected image plane using the formula of:

$$r = k_0 r' + k_1 r'^2 + k_2 r'^3 + k_3 r'^4 + ...$$  \hspace{1cm} (3.15)

where $r'$ is the radial distance measured in the distorted image plane. $r$ is the radial distance measured in the corrected image plane. But this way of mapping will result in blank pixels in the corrected image. That is due to the non-linear expansion of the image and the original image has to be mapped into a new enlarged image. See Figure 3.7, notice there are many blank pixels in the image especially in the peripheral part. This problem can be avoided by using another way of mapping, inverse-mapping method:
mapping from the corrected image plane to the distorted image plane. The coefficients estimation method described above (see equation (3.9)) used the inverse mapping method. In equation (3.9) \( r \) was measured in the corrected image plane. Thus, for every pixel in the corrected image plane, the corresponding location in the distorted image is obtained using the polynomial (3.9). The information (e.g. gray level) for that pixel location is assigned to the pixel in the corrected image plane. In case the pixel positions calculated using the inverse mapping (3.9) are non-integers, we use the simple way of round it to the closet integer. Because the corrected image enlarged the distorted image in Barrel type radial distortion, it is possible that several pixels in the corrected image may find the same pixel in the original image. In this way, all pixels in the corrected image plane can find their corresponding pixel value, thus generating a complete, undistorted image.
3.3. Marker design

As mentioned in the introduction section, researchers proposed several types of tracking methods such as optical flow (The optic flow field is the 2D distribution of apparent velocities that can be associated with the variation of brightness patterns on the image), feature points correspondence and model-based tracking. In our application, due to the diversity design of instruments, it is difficult to locate the instrument using shape analysis. To solve this problem, a black and strip shaped marker is designed. The marker is attached to the tip of instruments, then identified for the tracking task.

In general, the tracking task includes recognizing markers and calculating its relative position in the camera coordinate. In the real surgery, the instrument may enter the field of view from different positions and different angles, using shape analysis and pattern matching would be infeasible, for there is no fixed shape for tracking. Instead, we use the image contrast information alone for image segmentation to locate the maker position in the image. Two factors should be considered in the marker design. For one thing, in the real-time image tracking, the marker should not be too complex and should be easy to be located. For the other, as the restriction of the size of the instruments, the marker’s size should be chosen accordingly, not too large or too small.

Considering the above constrains, there are several possible marker design types. Figure 3.8 shows one of them, the marker is designed as several circles and attached to
the tip of the instruments. However, experiment shows that this type of maker can only provide two dimensional information.

To obtain three dimensional information using single endoscope, the maker is designed as shown in Figure 3.9. These strips are the designed markers. They are of the same size with the width of \( d \). The diameter of the instrument is \( M \). Points \( P_1 \), \( P_2 \) and \( P_3 \) are centers of each strip projected into the image plane.

Designing the marker in this shape has several advantages. First, the shape of the marker is simple and its contrast with the instruments and background is large, so it is easy for the program to identify the marker. Second, to acquire the depth information, the tool's diameter \( M \) could be taken into consideration. That is because no matter how the instrument rotate along it axis \( \mathbb{H} \), this parameter will not change in the endoscope image. Finally, even if one or two strips being blocked by an organ or other instruments, we can still get the instrument positioning information from the remaining strips. We could chose the strip number to be even four or five. However, the strip size can not be designed too narrow or too small, otherwise it will make it difficult to locate the marker. Considering the constrains of the size of the endoscope field of view and the instruments' size, we chose three strips in the marker design.
When needed, the tracking task is to move camera so as to position an instrument feature, such as markers, at a desired location of the endoscope field of view and also keep the distance between the camera tip and the instrument tip at a given value as the instrument moves.

See Figure 3.9, during the tracking procedure, one of the points among P1, P2, and P3 is being located, which is not being blocked and is the closest to the tip of the instrument. To determine which strip is the closest one to the tip, we can compare their diameter M value in the image (see Figure 3.9). Generally, the smallest M is the one closest to the tip of the scope. Because of the perspective view under endoscope (objects closer to the endoscope will appear bigger than the same size object which located farther away from the endoscope), generally the rectangle strip marker appears as the shape shown in Figure 3.10 a close trapezoid shape ABCD. We choose the length between A and B to represent the tool’s diameter (see Figure 3.10 (b)), and E the blob centre as our tracking point. Based on the change of point E in V, and V, directions in the image plane and the calibration parameters described in section 3.2, we can calculate the

![Diagram](image.png)
displacement in real world unit that the endoscope should be moved. From the change of tool diameter length M in the image plane, the relative distance displacement between the endoscope and the marker on the instrument can also be obtained. These information would then be utilized as feedback for manipulating the endoscope so to keep the tip of the instrument at a desired position in the endoscope filed of view. The detailed calculation procedure will be described in chapter 5 experiment results.
Appendix C – The Dilation and Erosion Operations

To explain the grayscale dilation and erosion operations, their binary versions are described first. According to Gonzalez and Woods [50], with \( A \) and \( B \) as sets in \( \mathbb{Z}^2 \), the binary dilation of \( A \) by \( B \), denoted \( A \oplus B \), is defined as

\[
A \oplus B = \{ z | (\hat{B})_z \cap A \neq \emptyset \}
\]  
(C.1)

It is based on obtaining the reflection of \( B \), the structuring element, about its origin and shifting this reflection by \( z \). The dilation of \( A \) by \( B \) then is the set of all displacements, \( z \), such that \( A \) and \( B \) overlap at least one element. According to this interpretation, Equation (C.1) may be rewritten as

\[
A \oplus B = \{ z | (\hat{B})_z \cap A \subseteq A \}
\]  
(C.2)

Figure C.1a) shows a simple set, and Figure C.1b) shows a structuring element and its reflection (the dark dot denotes the origin of the element). In this case the structuring element and its reflection are equal because \( B \) is symmetric with respect to its origin. The dashed line in Figure C.1c) shows the original set for reference, and the solid line shows the limit beyond which any further displacements of the origin of \( \hat{B} \) by \( z \) would cause the intersection of \( \hat{B} \) and \( A \) to be empty. Therefore, all points inside this boundary constitute the dilation of \( A \) by \( B \).
Figure C.1: Binary dilation operation (Gonzalez and Woods [50]).

For sets $A$ and $B$ in $Z^d$, the binary erosion of $A$ by $B$, denoted $A \odot B$ is defined as

$$A \odot B = \{ z \mid (B)_z \subseteq A \} \tag{C.3}$$

This equation indicates that the erosion of $A$ by $B$ is the set of all points $z$ such that $B$, translated by $z$, is contained in $A$. As in the dilation example, set $A$ is shown as a dashed line for reference in Figure C.2c). The boundary of the shaded region shows the limit beyond which further displacement of the origin of $B$ would cause this set to cease being completely contained in $A$. Thus, the locus of points within this boundary (the shaded region) constitutes the erosion of $A$ by $B$.

Figure C.2: Binary erosion operation (Gonzalez and Woods [50]).
Figure C.3a) shows a binary image composed of squares of sizes 1, 3, 5, 7, 8, and 15 pixels. To remove all the squares except the large ones, we can use the erosion operation with a structuring element of a size smaller than the square we want to keep. Figure C.3b) shows the result of eroding Figure C.3a) with a structuring element of size 13 × 13 pixels. Only portions of the largest squares remain. We can restore these three squares to their original 15 × 15 size by dilating them with the same structuring element used for erosion as shown in Figure C.3c).

Figure C.3: Binary erosion and dilation operations example (Gonzalez and Woods [50]).

The grayscale dilation and erosion operations are defined below. Throughout the following discussions, we deal with digital image functions of the form \( f(x, y) \) and \( b(x, y) \), where \( f(x, y) \) is the input image and \( b(x, y) \) is a structuring element, itself a sub-image function. Let \( Z \) denote the set of real integers and assume that \( (x, y) \) are integers from \( Z \times Z \) and that \( f \) and \( b \) are functions for assigning a gray-level value (a real number from the set of real numbers, \( \mathbb{R} \)) to each distinct pair of coordinates \( (x, y) \). If the gray levels also are integers, \( Z \) replaces \( \mathbb{R} \).

235
Grayscale dilation of \( f \) by \( b \), denoted \( f \oplus b \), is defined as

\[
(f \oplus b)(s,t) = \max_{(x,y) \in D_b} [f(x,y) + b(x,y)]; (s,x), (t,y) \in D_f
\]  

(\text{C.4})

where \( D_f \) and \( D_b \) are the domains of \( f \) and \( b \), respectively. The condition that \( (s-x) \) and \( (t-y) \) have to be in the domain of \( f \), and \( x \) and \( y \) have to be in the domain of \( b \), is analogous to the condition in the definition of binary dilation as in Equation (C.2), where the two sets have to overlap by at least one element.

Figure C.4 shows a greyscale dilation example. Figure C.4a) is a simple function \( f \), and Figure C.4b) is a structuring element \( b \) of height \( h \). Figure C.4c) shows the structuring element \( b \) sliding by \( f \). In Figure C.4d), the result of dilation is shown in solid line, and the dashed line shows the original function \( f \) for reference. There are two general effects of dilating a greyscale image: (1) If all the values of the structuring element are positive, the resulting image tends to be brighter than the original. (2) Dark details either are either reduced or removed, depending on how their values and shapes relate to the structuring element.
Grayscale erosion, denoted $f \Theta h$, is defined as

$$f \Theta h(x, y) = \min \{f(x + s, y + t) - b(s, t) \mid (s, t) \in D_f, (x, y) \in D_b\}$$

(C.5)

where $D_f$ and $D_b$ are the domains of $f$ and $h$, respectively. The condition that $(s + x)$ and $(t + y)$ have to be in the domain of $f$, and $x$ and $y$ have to be in the domain of $h$, is analogous to the condition in the definition of binary erosion as in Equation (C.3), where the structuring element has to be completely contained by the set being eroded. Figure C.5 shows the result of eroding the function $f$ in Figure C.4a) by the structuring element $h$ in Figure C.4b). The result of erosion is shown in solid line, and the dashed line shows the original function $f$ for reference. There are two general effects of eroding a greyscale
image: (1) If all the elements of the structuring are positive, the resulting image tends to be darker than the original. (2) The effect of bright details in the original image smaller in area than the structuring element is reduced, with the degree of reduction being determined by the grey-level values surrounding the bright detail and by the shape and amplitude values of the structuring element itself.

Figure C.5: Greyscale erosion operation of the function $f$ in Figure C.4a) by the structuring element $b$ shown in Figure C.4b) (Gonzalez and Woods [50]).
Appendix D – Introduction to Neural Network

What is Neural Network

According to Stergiou and Siganos [60], an Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

Why Use Neural Network

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.

3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

**Neural Network versus Conventional Computer**

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don’t exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be
functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

How the Human Brain Learns

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron shown in Figure D.1 sends out spikes of electrical activity through a long, thin stand known as an *axon*, which splits into thousands of branches. At the end of each branch, a structure called a *synapse* shown in Figure D.2 converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected
neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

![Figure D.1: Components of a neuron.](image)

![Figure D.2: The synapse.](image)
From Human Neuron to Artificial Neuron

We conduct these neural networks by first trying to deduce the essential features of neurones and their interconnections. We then typically program a computer to simulate these features as shown in Figure D.3. However because our knowledge of neurones is incomplete and our computing power is limited, our models are necessarily gross idealisations of real networks of neurones.

Figure D.3: The neuron model.

A Simple Neuron

An artificial neuron is a device with many inputs and one output as shown in Figure D.4. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.
Firing rules

The firing rule is an important concept in neural networks and accounts for their high flexibility. A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained.

A simple firing rule can be implemented by using Hamming distance technique. The rule goes as follows:

Take a collection of training patterns for a node, some of which cause it to fire (the 1-taught set of patterns) and others which prevent it from doing so (the 0-taught set). Then the patterns not in the collection cause the node to fire if, on comparison, they have more input elements in common with the 'nearest' pattern in the 1-taught set than with the 'nearest' pattern in the 0-taught set. If there is a tie, then the pattern remains in the undefined state.
For example, a 3-input neuron is taught to output 1 when the input \((X_1, X_2 \text{ and } X_3)\) is 111 or 101 and to output 0 when the input is 000 or 001. Then, before applying the firing rule, the truth table is shown in Table D.1.

<table>
<thead>
<tr>
<th>X1:</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X3:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OUT:</td>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
<td>1</td>
<td>0/1</td>
</tr>
</tbody>
</table>

As an example of the way the firing rule is applied, take the pattern 010. It differs from 000 in 1 element, from 001 in 2 elements, from 101 in 3 elements and from 111 in 2 elements. Therefore, the 'nearest' pattern is 000, which belongs in the 0-taught set. Thus, the firing rule requires that the neuron should not fire when the input is 001. On the other hand, 011 is equally distant from two taught patterns that have different outputs and thus the output stays undefined (0/1).

By applying the firing in every column, Table D.2 is obtained.
Table D.2: Truth table for a 3-input neuron example after applying the firing rule.

<table>
<thead>
<tr>
<th>X1:</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X3:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>OUT:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>0/1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The difference between the Table D.1 and Table D.2 is called the **generalisation of the neuron**. Therefore, the firing rule gives the neuron a sense of similarity and enables it to respond 'sensibly' to patterns not seen during training.

**A more complicated neuron**

The previous neuron does not do anything that conventional computers do not do already. A more sophisticated neuron shown in Figure D.5 is the McCulloch and Pitts model (MCP). The difference from the previous model is that the inputs are 'weighted', the effect that each input has at decision making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, $T$, the neuron fires. In any other case the neuron does not fire.
In mathematical terms, the neuron fires if and only if

\[ X_1 W_1 + X_2 W_2 + X_3 W_3 + \ldots > T \]

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by changing its weights and/or threshold. Various algorithms exist that cause the neuron to ‘adapt’; the most used ones are the Delta rule and the back error propagation. The former is used in feed-forward networks and the latter in feedback networks.

Feed-forward networks

Feed-forward ANNs shown in Figure D.6 allow signals to travel one way only, from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straightforward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.
Feedback networks

Feedback networks shown in Figure D.7 can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their ‘state’ is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.
Network Layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units as shown in Figure D.6.

- The activity of the input units represents the raw information that is fed into the network.
- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

The Learning Process

The memorization of patterns and the subsequent response of the network can be categorized into two general paradigms:

1. Associative mapping in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms:
   - Auto-association: an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern completion, i.e. to produce a pattern whenever a portion of it or a
distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns.

- **Hetero-association** is related to two recall mechanisms:
  - Nearest-neighbour recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented, and
  - Interpolative recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping is classification, i.e. when there is a fixed set of categories into which the input patterns are to be classified.

2. Regularity detection in which units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.

Every neural network possesses knowledge that is contained in the values of the connections weights as shown in Figure D.8. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights.
Information is stored in the weight matrix $W$ of a neural network. Learning is the determination of the weights. Following the way learning is performed, we can distinguish two major categories of neural networks:

- Fixed networks in which the weights cannot be changed, i.e. $\frac{dW}{dt} = 0$. In such networks, the weights are fixed a priori according to the problem to solve.

- Adaptive networks which are able to change their weights, i.e. $\frac{dW}{dt} \neq 0$.

All learning methods used for adaptive neural networks can be classified into two major categories:

- Supervised learning which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning. An important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error.
between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms is the least mean square (LMS) convergence.

- Unsupervised learning uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning. The aspect of learning concerns the distinction or not of a separate phase, during which the network is trained, and a subsequent operation phase. We say that a neural network learns off-line if the learning phase and the operation phase are distinct. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed on-line.

**Transfer Function**

The behaviour of an ANN depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

- Linear (or ramp)
- Threshold
- Sigmoid
For linear units, the output activity is proportional to the total weighted output. For threshold units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value. For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another in Figure D.6, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

We can teach a three-layer network to perform a particular task by using the following procedure:

1. Present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
2. Determine how closely the actual output of the network matches the desired output.
3. Change the weight of each connection so that the network produces a better approximation of the desired output.
An Example to illustrate the above teaching procedure:

Assume that we want a network to recognize hand-written digits. We might use an array of, say, 256 sensors, each recording the presence or absence of ink in a small area of a single digit. The network would therefore need 256 input units (one for each sensor), 10 output units (one for each kind of digit) and a number of hidden units. For each kind of digit recorded by the sensors, the network should produce high activity in the appropriate output unit and low activity in the other output units.

To train the network, we present an image of a digit and compare the actual activity of the 10 output units with the desired activity. We then calculate the error, which is defined as the square of the difference between the actual and the desired activities. Next, we change the weight of each connection to reduce the error. We repeat this training process for many different images of each different images of each kind of digit until the network classifies every image correctly.

To implement this procedure we need to calculate the error derivative for the weight (EW) in order to change the weight by an amount that is proportional to the rate at which the error changes as the weight is changed. One way to calculate the EW is to perturb a weight slightly and observe how the error changes. However, that method is inefficient because it requires a separate perturbation for each of the many weights.

Another way to calculate the EW is to use the Back-propagation algorithm which is described below, and has become nowadays one of the most important tools for training neural networks. It was developed independently by two teams, one (Fogelman-Soulie, Gallinari and Le Cun) in France, the other (Rumelhart, Hinton and Williams) in U.S.
The Back-Propagation Algorithm

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the EW.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straightforward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection.
Note that for non-linear units, the back-propagation algorithm includes an extra step. Before back-propagating, the EA must be converted into the EI, the rate at which the error changes as the total input received by a unit is changed.
Appendix E – Motion Recognition Algorithms Matlab Code

Optical Flow
Created by Ghassan Hamarneh

clear all;

[filename1,pathway]=uigetfile('*.bmp;*.tif','Choose your first file');

[x,pathway]=uigetfile('*.bmp;*.tif','Choose your last file');

map=[(0:1/255:1)' (0:1/255:1)' (0:1/255:1)'];

x=find(filename1=='.');

ext=filename1(x+1:end);

end=

% Make sure the files are grayscale (Intensity) images
I1=double(imread([pathway,filename1],ext));
I2=double(imread([pathway,filename2],ext));

% Make sure the files are Grayscale (Intensity) images
I1=double(imread([pathway,filename1],ext));
I2=double(imread([pathway, filename2],ext));

imagesc(I1);

imagesc(I2);

% Specify the crop rectangle with mouse
imagesc(I1);
imagesc(I2);

figure(1)

subplot(2,1,1)

colormap(map)

imagesc(I1)

subplot(2,1,2)

imagesc(I2)

pause

% Laplace
a=1/(12);
b=1/6;
L=[a b a; b 0 b; a b a];

% Gradient
[Ex,Ey]=gradient(I1);

Et=I2-I1;

% Initialize U and V
U=zeros(size(Ex));
V=zeros(size(Ey));

lambda=10;

% Iteration to get U,V

NOIT=3;

for i=1:2:10;

end
Uav=conv2(U,L,'same');
Vav=conv2(V,L,'same');

U=Uav-Ex.*(Ex.*Uav.*Ey.*Vav+Ey.*Ex.*Vav)/(lambda^2+Ex.^2+Ey.^2);
V=Vav-Ey.*(Ex.*Uav.*Ey.*Vav+Ex.*Ey.*Vav)/(lambda^2+Ex.^2+Ey.^2);
end

[X,Y]=meshgrid(1:size(I,1)),(1:size(I,1)));
figure(1)
clear

imagesc(I);
colormap('gray')

imagesc(I(1:5:end,1:5:end));
U=conv2(U,1/25*ones(5),'same');
V=conv2(V,1/25*ones(5),'same');
hold on
quiver(X,Y,U,V,0,'w');
function [padi,nadi,aadi] = makeADIs( refFrame, framename, ext, first, last, digits, thresh )
% MAKEADIS creates accumulative difference images for a given image sequence
% A positive, a negative, and an absolute ADI are returned, each of the same
% dimensions as the passed in images.
% 'refFrame' is the name of the reference frame (no objects in motion) without
% the extension.
% 'framename' is the name of each file in the sequence without the trailing
% digits or the extension.
% 'ext' is the extension of the files.
% 'first' is the number at the end of the filename of first file in the
% sequence.
% 'last' is the same as first, but for the last image in the sequence.
% 'digits' describes the number of digits in the filenames.
refImage = imread(refFrame,ext);
if (ndims(refImage) == 3)
refImage = rgb2gray(refImage);
end;
refImage = int16(refImage);
padi = int16(zeros(size(refImage)));
nadi = int16(zeros(size(refImage)));
aadi = int16(zeros(size(refImage)));
h = waitbar(0,'Progress...');
for j = first+1:last
nextFrame = imread(sprintf('s%0%d', framename, digits,j),ext);
if (ndims(nextFrame) == 3)
nextFrame = rgb2gray(nextFrame);
end;
nextFrame = int16(nextFrame);
pDIF = refImage - nextFrame > thresh;
nDIF = refImage - nextFrame < -thresh;
aDIF = abs(refImage - nextFrame) > thresh;
padi = padi + int16(pDIF);
nadi = nadi + int16(nDIF);
aadi = aadi + int16(aDIF);
waitbar(j/last);
end;
close(h);

[pos,neg,abt] = makeADIs('background', 'scene', 'bmp', 1, 4, 1, 80);
[pos,neg,abt] = makeADIs('background', 'scene', 'bmp', 11, 20, 1, 80);
[pos,neg,abt] = makeADIs('background', 'scene', 'bmp', 21, 27, 1, 60);
[pos,neg,abt] = makeADIs('scene3', 'scene', 'bmp', 31, 38, 1, 60);
[pos,neg,abt] = makeADIs('scene3', 'scene', 'bmp', 41, 54, 1, 60);
[pos,neg,abt] = makeADIs('scene4', 'scene', 'bmp', 41, 54, 1, 60);
image(pos); colormap('default');
image(neg); colormap('default');
image(abt); colormap('default');
REFERENCE LIST


