Hand Tracking and its Pattern Recognition
in a Network of Calibrated Cameras

by

Jingya Wang
B.Eng., Nanjing Normal University, 2010

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Sciences

© Jingya Wang 2015
SIMON FRASER UNIVERSITY
Spring 2015

All rights reserved.
However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for "Fair Dealing". Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.
APPROVAL

Name: Jingya Wang

Degree: Master of Applied Science

Title: Hand Tracking and its Pattern Recognition in a Network of Calibrated Cameras

Examining Committee: Chair: Dr. Carlo Menon
Associate Professor

________________________________________
Dr. Shahram Payandeh
Senior Supervisor
Professor

________________________________________
Dr. Jie Liang
Supervisor
Associate Professor

________________________________________
Dr. Ghassan Hamarneh
Internal Examiner
Professor

Date Approved: April 1th, 2015
Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the non-exclusive, royalty-free right to include a digital copy of this thesis, project or extended essay[s] and associated supplemental files ("Work") (title[s] below) in Summit, the Institutional Research Repository at SFU. SFU may also make copies of the Work for purposes of a scholarly or research nature; for users of the SFU Library; or in response to a request from another library, or educational institution, on SFU’s own behalf or for one of its users. Distribution may be in any form.

The author has further agreed that SFU may keep more than one copy of the Work for purposes of back-up and security; and that SFU may, without changing the content, translate, if technically possible, the Work to any medium or format for the purpose of preserving the Work and facilitating the exercise of SFU’s rights under this licence.

It is understood that copying, publication, or public performance of the Work for commercial purposes shall not be allowed without the author’s written permission.

While granting the above uses to SFU, the author retains copyright ownership and moral rights in the Work, and may deal with the copyright in the Work in any way consistent with the terms of this licence, including the right to change the Work for subsequent purposes, including editing and publishing the Work in whole or in part, and licensing the content to other parties as the author may desire.

The author represents and warrants that he/she has the right to grant the rights contained in this licence and that the Work does not, to the best of the author’s knowledge, infringe upon anyone’s copyright. The author has obtained written copyright permission, where required, for the use of any third-party copyrighted material contained in the Work. The author represents and warrants that the Work is his/her own original work and that he/she has not previously assigned or relinquished the rights conferred in this licence.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2013
Abstract

This thesis presents a vision-based approach for hand gesture recognition which combines both trajectory and hand posture recognition. The hand area is segmented by fixed-range CbCr from cluttered and moving backgrounds, and tracked by Kalman Filter. With the tracking results from two calibrated cameras, the 3D hand motion trajectory can be reconstructed. It is then modeled by dynamic movement primitives (DMP) and a support vector machine (SVM) is trained for trajectory recognition. Scale-invariant feature transform (SIFT) is employed to extract features on segmented hand postures, and a novel strategy for hand posture recognition is proposed. A gesture vector is introduced to recognize hand gesture as a whole which combines the recognition results of motion trajectory and hand postures, where an SVM is trained for gesture recognition based on gesture vectors.

Keywords: Hand gesture recognition; DMP; Kalman Filter; SIFT; SVM
To my family, Mom, Dad, and Gavin, I love you!
"Every present moment is the realization of a dreamt future."

--- To myself
Acknowledgements

First and foremost, I would like to dedicate faithful thanks to my senior supervisor, Dr. Shahram Payandeh, for offering me the opportunity to work on this project, supporting and encouraging me through the research with his patience and enthusiasm. He is an excellent researcher, supervisor and teacher, who is always generous to share his knowledge with me, give me helpful advice and explore my potential at every stage of my research.

I would also like to thank my committee members: my supervisor Dr. Jie Liang and examiner Dr. Ghassan Hamarneh for their time, patience and efforts in reviewing my thesis, also their thoughtful comments and advice on the thesis; and Dr. Carlo Menon for chairing the presentation of my thesis.

My gratitude also goes to my colleagues and friends in the Experimental Robotics Laboratory. Many thanks to Jiawei Zhou, Xiaochen Dai, Carlos Diaz, Xuhong Liu, Xiaochuan Sun and Han Xiao for their inspiration and help to both of my research and life.

In addition, to my friends in and out Engineering Science who made the journey more enjoyable during all these years. Special thanks go to Yasong Li, Jing Xu, Michelle Cua, Sieun Lee, Maxwell Miao, Lukas-Karim Merhi, Yifan Jian, Kevin Wong, and Muyar Htun for giving me a help in my study and life. Many thanks to Yasong Li, Lisa Poole and Renee Anne McCallum for spending their time to proofread this thesis.

Finally, I would like to thank my family, my parents Yanmian Liu and Jianzhong Wang, as well as my parents-in-law Yanhua Xie and Shuli Ding, also my husband Weiguang Gavin Ding for their unconditional love, supports, encouragements and understanding.
List of Acronyms

CLI    Command-line Interface
HCI    Human-Computer-Interfaces
VRS    Video Relay Service
HMMs   Hidden Markov Models
DMP    Dynamic Movement Primitives
2D     Two-Dimensional
3D     Three-Dimensional
KF     Kalman Filter
FOV    Field-of-View
SIFT   Scale Invariant Feature Transform
RGB    Red-Green-Blue
HSV    Hue-Saturation-Value
LOESS  Locally Weighted Scatter-plot Smoothing
RLOESS Robust Version of LOESS
SVM    Support vector machine
DoG    Difference of Gaussian
Contents

Approval ii
Partial Copyright License iii
Abstract iv
Dedication v
Quotation vi
Acknowledgements vii
List of Acronyms viii
Contents ix
List of Tables xii
List of Figures xiii

1 Introduction 1
1.1 Motivation ................................................. 1
1.2 Related Work ............................................. 2
  1.2.1 Overview of Hand Gesture Recognition ................. 3
  1.2.2 Hand Segmentation and Posture Recognition ........... 4
  1.2.3 Hand Motion Trajectory Recognition .................... 5
1.3 Framework ................................................. 6
  1.3.1 System Overview ..................................... 6
  1.3.2 Software Environment and Hardware Setups .......... 7
1.4 Contributions and Layouts ................................. 8
2 Preliminaries

2.1 Hand Segmentation

2.1.1 Skin Segmentation

2.1.2 Hand Area Extraction

2.2 Two Camera Setup and Calibration

2.3 Result for 3D Coordinate Reconstruction

2.4 Discussion

3 Hand Tracking and Trajectory Reconstruction

3.1 Hand Tracking Initialization

3.2 Kalman Filter for Hand Tracking

3.3 Area Constraint for Hand Tracking

3.4 Experiment Result for Hand Tracking

3.4.1 Single Hand Tracking

3.4.2 Two Hands Tracking

3.5 Trajectory Reconstruction and Smoothing

3.5.1 Trajectory Reconstruction

3.5.2 Trajectory Smoothing

3.6 Discussions

4 Gesture Recognition

4.1 Trajectory Recognition

4.1.1 An Overview of Dynamic Movements Primitives

4.1.2 Extracting Weight Vector from 3D Trajectory

4.1.3 Training Stage for Trajectory Classifier Using SVM

4.1.4 Testing Stage for Trajectory Recognition

4.2 Posture Recognition

4.2.1 Scale-Invariant Feature Transform (SIFT)

4.2.2 Bag of Visual Words

4.2.3 Training Stage for Posture Classifier Using SVM

4.2.4 Testing Stage for Posture Recognition

4.3 Gesture Recognition

4.3.1 Gesture Vector
4.3.2 Defining Gesture Classes and Gesture Vectors ................. 57
4.3.3 Training Stage for Gesture Classifier Using SVM ............. 57
4.3.4 Testing Stage for Gesture Recognition ...................... 59
4.4 Discussions ................................................. 60

5 Conclusions and Future Work .................................. 61
  5.1 Conclusions ............................................... 61
  5.2 Future Work .............................................. 62

Bibliography .......................................................... 64

Appendix A Conversions of Color Space .......................... 71
  A.1 RGB to HSV ............................................... 71
  A.2 RGB to YCbCr ............................................ 72

Appendix B Multi-Camera Geometry ............................... 73
  B.1 Pinhole Camera Model ................................. 73
  B.2 Single-Camera Calibration ............................. 74
  B.3 Multi-Camera Calibration .............................. 75
  B.4 3D Coordinate Reconstruction ......................... 76

Appendix C Appendices: K-means Clustering ...................... 79

Appendix D Appendices: Support Vector Machine .................. 81

Appendix E Appendices: Locally Weighted Scatterplot Smoothing (LOESS) 83

Appendix F Appendices: Dynamic Movement Primitives ........... 84

Appendix G Appendices: Hidden Markov Models .................... 86
# List of Tables

1.1 Software and hardware for the experimental studies .......................... 7

2.1 The skin segmentation results of fixed-range CbCr classifier. ............. 14

2.2 The 3D coordinates of eight corner points on the plastic box in the world coordinate system recovered by their projected points on the image plane of two cameras. ......................................................... 20

2.3 The relative location relationship between eight corner points. ............ 20

3.1 The logic table of initialization. ..................................................... 23

4.1 Recognition accuracy with different number of $D_w$ ......................... 47

4.2 Recognition result of testing trajectories. ................................. 47

4.3 The number of matching points between different posture classes ...... 52

4.4 Training accuracy with different number of training data ................. 54

4.5 Result matrix for posture recognition on the testing set ................... 55

4.6 Recognition result and abandon rate for the testing set .................. 56

4.7 The names of gesture classes and their compositions ....................... 57

4.8 The cross-validation accuracy on training gestures ......................... 59

4.9 Gesture recognition results of the testing set .............................. 60
List of Figures

1.1 Hand gesture controlled HIC. (a) A Kinect game [68]. (b) Processing surgery image in an operating room [17]. (c) Controlling a humanoid robot [28] .... 2

1.2 The setup of the cameras. .......................................................... 7

1.3 The framework of hand gesture recognition system. ......................... 8

1.4 (a) Sony EVI-D100 PTZ color video camera [25]. (b) Euresys Picolo PCI frame grabber [59] ................................................................. 9

2.1 The complicated appearance of difference hand postures and orientations. 12

2.2 Hand segmentation by different skin classifier. ............................. 13

2.3 Skin segmentation results by fixed-range CbCr classifier when the background contains big skin-color-liked area other than hand. (a) The original image. (b) The segmentation result. ................................. 15

2.4 Segmentation results by adding a size constraint on the blobs. .......... 15

2.5 The set-up of the calibrated cameras. ............................................. 17

2.6 Schematics of spatial relationship of camera $C_1$ and camera $C_2$ with respect to world coordinates with origin at $O$. ................................. 17

2.7 Dimensions of the transparent box. .............................................. 19

2.8 The transparent box in the view of two cameras. The eight corners are marked with red crosses. ................................................. 19
3.1 (a) and (c) are sample frames from camera $C_1$ of a two-hand gesture; (b) and (d) are the corresponding sample frames taken from camera $C_2$. The red bounding boxes in (a) and (b) represent the initial area for left hand; the green bounding boxes are for the right hand. The crosses in (a) and (b) shows the centroids of the detected skin-color blobs. Only the centroids of the blobs which enter into the initial areas are taken as hand blobs and labeled. The green bounding boxes label the right hand in (c) and (d); the red bounding boxes label the left hand.

3.2 (a) and (c) are sample frames from camera $C_1$ of a one-hand gesture; (b) and (d) are the corresponding sample frames taken from camera $C_2$. The red bounding boxes in (a) and (b) represent the initial area for left hand; the green bounding boxes are for the right hand. The crosses in (a) and (b) show the centroids of the detected skin-color blobs. The blobs which centroids enter the left initial area are labeled as left. Since there is no blob detected in the right hand initial areas, only the left hand is being labeled and tracked in (c) and (d).

3.3 The hand location is represented by the pixel value of the centroid of the segmented hand blob.

3.4 The iteration circle of KF.

3.5 The bounding box expands when the hand blob is overlapped with background noises. The red crosses show the location of hand palm and the green one shows the center of the enlarged bounding box which is mis-marked as the hand location.

3.6 When hands contact and occlusion happen, both hand is included in one bounding box.

3.7 For hand motion without contaminated by background noise, the increase and decrease ratio is in the range of 0.8 and 1.2 between two adjacent frames.

3.8 (a)Hand tracking result before adding the area constraint; (b)hand tracking result after adding the area constraint.

3.9 The system setup of two cameras. The example frames taken by camera $C_1$ and $C_2$ are listed on the left of this image.
3.10 Sample frames from gesture video with a hand moving in slow speed and its KF tracking result. The first column ((a),(e),(i),(m)) and the second column ((b),(f),(j),(n)) display the original frame clipped from camera $C_1$ and camera $C_2$ respectively. The tracking result for both cameras views are showed in the third column ((c),(g),(k),(o)) and fourth column ((d),(h),(l),(p)).

3.11 Sample frames clipped from gesture video with hand moving with fast speed and direction change suddenly. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_1$. The second row ((e),(f),(g),(h)) shows the tracking result.

3.12 Sample frames clipped from gesture video with hand moving and overlapping with background noise. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_2$. The second row ((d),(e),(f),(g)) shows the tracking result.

3.13 Sample frames clipped from gesture video with a hand blob overlapping with a background blob. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_1$. The second row ((e),(f),(g),(h)) shows the tracking result.

3.14 Sample frames clipped from gesture video of two hand moving with no contact. The original frame captured from camera $C_1$ and camera $C_2$ are shown in the first row ((a),(b),(c),(d)) and second row ((e),(f),(g),(h)). The third ((g),(h),(i)) and fourth row ((j),(k),(l)) display the tracking result.

3.15 Sample frames from a gesture video with two hands contact with each other. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_1$. The second row ((e),(f),(g),(h)) shows the tracking result.

3.16 Sample frames from a gesture video which right hand is fully occluded. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_1$. The second row ((e),(f),(g),(h)) shows the tracking result.

3.17 Examples of reconstructed 3D moving trajectory. (a) single hand moving trajectory; (b) the trajectory of two hand moving without contact; (c) the trajectory of two hand moving with contact; (b) the trajectory of two hand moving with occlusion happens. Red crosses show the trajectory of the right hand and blue crosses for the left hand.
3.18 (a) Reconstructed 3D trajectory when hand moving in a circle. (b) The hand moving trajectory is represented in $x$, $y$ and $z$ directions.

3.19 Comparison of original data points (red dots) and smoothed data points (blue curve) using LOESS (Figure (a) and (b)) and RLOESS (Figure (c) and (d)) with different span. (a) span = 0.1; (b) span = 0.5; (c) span = 0.1; (d) span = 0.5.

3.20 Smoothed trajectory by RLOESS.

4.1 (a), (c), (e) and (g) are the comparisons between DMP learned trajectories (in blue) and the original trajectories (in green) along with different weight vectors dimension $D_w$. (b), (d), (f) and (h) show the learned weight vectors.

4.2 Trajectory samples of five classes. The starting point is marked in green, and the ending point is in red.

4.3 Hand posture is changing along the trajectory, and the hand posture is captured by both cameras.

4.4 For each level of the octave, the original image is convolved with Gaussian in different variance. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images. After each octave, the Gaussian images is down-samples by a factor of 2, and the process repeated [47].

4.5 Extrema points are detected by comparing a pixel (marked with X) to its 26 neighbours (marked with circles) in $3 \times 3$ regions at the current and adjacent DoGs. [47]

4.6 The matching feature points detected by SIFT on two palms.

4.7 Six classes of posture, from left to right the postures are: "Palm", "V", "Point", "Six", "Fist" and "Eight".

4.8 Feature points detected on six postures by SIFT. From left to right the postures are: "Palm", "V", "Point", "Six", "Fist" and "Eight".

4.9 The pipeline of generating histogram of visual word for each image based on the detected feature points.

4.10 The front view and side view of 6 posture classes.

4.11 Examples of misclassified postures.
4.12 The example of eight gesture classes. The beginning and ending frames are shown, and the moving directions are shown in blue arrows. 58

4.13 The testing pipeline of gesture recognition. 59
Chapter 1

Introduction

It is shown that nearly 90% of daily communication is non-verbal [50]. Hand gestures have always been a powerful non-verbal communication tool in people’s daily life. As technology is developing, hand gesture recognition, as a key component of natural Human-Computer-Interfaces, has become an important topic. Hand gestures in this thesis are referred to the combination of hand moving trajectory, and hand posture, which means the hand shape and appearance. This chapter first addresses the motivation for hand gesture recognition using multiple cameras in Section 1.1. Following that, related work is reviewed in Section 1.2. An integrated framework for hand gesture recognition using multi-cameras is displayed in Section 1.3. The main contributions we have made are shown in Section 1.4. The layout of this thesis is also included.

1.1 Motivation

Decades ago, the interaction between human and computers was command-line interface (CLI) through a keyboard. With new the technology develops, other input devices were needed to make Human-Computer-Interfaces (HCI) more convenient and usable. The mouse, as a widespread input device today, was invented. The development did not stop there. As years past, people keep making efforts to seeking other HCI to meet the growing needs. Now, computers are not just computational machines. They are made for many general purposes such as office suites, design, communication, robot controlling and entertainment. New HClS need to be introduced.
It is concluded that 65% of human communication involves non-verbal gestures [35]. For hearing impaired people, this percentage rises to nearly 100%. Therefore, hand gestures, as one of the most dominant communication tools in our daily life, express abundant messages. Developing a hand gesture interface to enrich the HCIs is not only convenient for hearing impaired people but also useful to meet people’s needs for different tasks.

A Video Relay Service (VRS) is being developed for hearing impaired people. VRS is designed as a manual interpreter that translates sign language into corresponding text or voice and vice versa [60]. Hand gesture recognition is the essential component of VRS.

In the entertainment area, game consoles, such as Microsoft Kinect (Figure 1.1a), and Wii have also updated our knowledge of HCI [15]. Games that involved gesture control give players a more realistic game experiences. In the future, computers will be integrated with many daily devices, such as televisions, cars and household appliances.

Hand gesture recognition also plays an important role to improving non-contact HCI. For example, in the operating room (Figure 1.1b), contact HCI will bring many inconveniences and troubles such as infection or misunderstanding if an assistant is operating the HCI based on the surgeon’s instruction. By developing hand gesture recognition, an accurate non-contact HCI will be achieved to reduce unnecessary communication and optimise surgery processes. Also, hand gestures could be used to control the movement of a robot (Figure 1.1c).

![Figure 1.1: Hand gesture controlled HCI. (a) A Kinect game [68]. (b) Processing surgery image in an operating room [17]. (c) Controlling a humanoid robot [28].](image)

1.2 Related Work

In this thesis, hand gestures are referred to the hand moving trajectory and also the hand postures made along the trajectory. In this section, a review of previous research and
developments on hand gesture recognition are presented, both the methods of trajectory and posture recognition are included.

1.2.1 Overview of Hand Gesture Recognition

Based on input devices, hand gesture recognition can be mainly divided into two categories: device-based and vision-based. Device-based hand gesture recognition requires the users to wear devices such as gloves, marks or other sensors to acquire hand or arm joint angles and their spatial positions. A data glove modelling a 3D hand is designed. This glove measures the angles of finger bending using analog flex sensors.

Hand position and orientation are measured by ultrasonics or magnetic flux sensors. A multi-coloured glove is adopted to reconstruct the pose of the hand based on its color pattern. Due to the development of sensors, device-based hand gesture recognition collects relatively accurate information when the hand is making gestures and also such recognition is robust to illuminate changes which is the main drawback of vision-based hand gesture recognition.

For vision-based hand gesture recognition, it acquires gesture information mainly by the hand color or texture. Users do not need to wear any cumbersome devices. Cameras can capture hand movements for further analysis. In [46], the hand gesture controls the movement of a humanoid robot. They use a camera which is mounted on the robot to acquire 2D information of hand motion trajectories. Based on the recognition results, they make the robot do corresponding actions. To acquire 3D hand positions and solve the occlusion problem during a hand motion, In [77], they built a system with five cameras. This system successfully realizes hand tracking and gesture recognition. But it requires a clean background to extract hand features, which restricts its application in many practical cases. In [71], a wearable real-time hand gesture recognition system was developed. They mount a camera on a hat and point downward toward the hands. The hand gesture recognition is done base on such input. Vision-based hand gesture recognition is widely used and explored for its conveniences. But the difficulties of vision-based approaches are the same with what they are facing in the image processing and computer vision areas: not robust enough to illumination variation, background noise, and occlusion.
There are other methods for hand gesture recognition. To avoid the weak points of vision-based methods, some researchers use depth cameras to capture the hand motions [45, 61, 80]. Depth cameras generate gray-scale images based on the distances between the objects and the camera. For hand gesture recognition, the hand is usually the nearest moving object to the camera. Based on setting thresholds on gray-scale images, background noise can be eliminated.

1.2.2 Hand Segmentation and Posture Recognition

In this thesis, the system is working under a relative steady lighting condition. To avoid cumbersome devices, a vision-based gesture recognition system with multi-cameras is built. This section reviews the methods of hand segmentation and posture recognition for vision-based gesture recognition.

Hands, as one of the most dexterous part of a human body, have 27 degrees of freedom [23], which occupy various shapes and appearances. To extract hand regions from the rest of the image, colour cues and motion cues are most often used to segment a hand from the background. The skin colour is usually more distinctive and less sensitive to illumination changes in the hue-saturation space than in RGB colour space [57]. Most of the color segmentation approaches rely on histogram matching [1, 48]. Colour cue is not robust to illumination variation and frequently results in undetected skin regions or falsely detected non-skin area. To solve such a problem, some assumptions, such as area size (scale filter) or certain spatial positions (position filter), are adopted. Another solution to such a problem is to allow the users to wear gloves having distinctive colors [79] or special markers (LED light [56, 75], fluorescent material [42]) or clean the background so there is not much noise from it [77]. These methods are robust to illumination variation but lose the intention of liberating hands from gloves. Motion cue requires the main component in the image frames be the moving hand or arm, it also requires the hand gesture be made before a stationary background [30, 8, 46].

Feature extraction is very important to posture recognition. The simplest and most frequently used features are hand silhouettes which can be easily extracted. Contours are a group of commonly used features. Several different edge detection schemes can be used to produce contours [57]. Contours are often employed with 3D hand models that are build based on hand shape and structure. Hand posture can be recognized by comparing the
similarities between detected contours and generated contours based on the hand model [72, 31]. In [72], they build a 3D hand model with 27 degrees of freedom to model the articulates of the hand. The hand gesture recognition is done by comparing the generated contours based on the hand model and the inputed hand images. Another frequently used feature in posture recognition is the finger tip. Postures can be recognized base on the positions of finger tips, either extracted by markers (LED light [56, 75] or spatial colour ) or convex hull on silhouette [44]. There are also other feature detectors that can be applied to achieve posture recognition, such as scale-invariant feature transform (SIFT) which is insensitive to illumination variations, scale and orientation changes [86, 66, 78, 55]; Haar-like feature which transforms hand postures into a coefficients vector in the Haar wavelet transform [14] ; and orientation histogram [29].

1.2.3 Hand Motion Trajectory Recognition

Hand motion is a movement that involves both time and spatial information. The same movement made by different people usually have different a scale, speed and shape due to the difference in people's size, habit and states. Template matching that works well for posture recognition would fail on dynamic trajectory recognition [53]. Solving such a problem involves approaches such as hidden Markov models (HMM), finite-state machines (FSM), and dynamic movement primitives (DMP).

As a statistical model, HMM has been found to be efficient on modeling Spatio-temporal time series where the same gesture can differ in shape and duration. In [24], the researchers developed a hand motion trajectory recognition system by segmenting extracted 2D hand motion trajectories into straight-line pieces, and the orientation of each piece is grouped into a vector to represent the whole trajectory. Each straight-line segment is a single HMM state. The feature vector is taken as the input. HMM is used for matching the input with the trajectory templates to find the corresponding maximal gesture model. Other features extracted by the Gaussian mixture model and principal component analysis [3, 4] can be used for HMM to classify and recognize trajectories.

In the FSM approach, a gesture can be modeled as an ordered sequence of states in a spatio-temporal configuration space. The number of states in the FSM may vary between applications. The gesture is recognized as a prototype trajectory from an unsegmented,
continuous stream of sensor data constituting an ensemble of trajectories. The trajectories of the gestures are represented as a set of points in 2D space [53]. In [37], an FSM recognizer is built based on the spatial and temporal information extracted from given gestures. Each gesture has a corresponding FSM. When a new gesture comes, the recognizer only takes the current data sample into account since the past is modelled by the current state. In this way, the gesture recognition can be done in real-time which is different from other approaches that require the complete gesture data for recognition. Davis and Shah [21] proposed a method to use FSM to model four distinct phases (static starting position, smooth motion of a hand, static end position and smooth motion of the hand back to the starting position) of a gesture. Gestures are represented as a list of vectors and then matched with the stored templates.

In 2006, a method called DMP was proposed by [65] and was extended in [38] for human limb control. This method makes robots imitate human limb actions based on the starting and ending points, as well as the position, speed and acceleration information then learn from actual human moves. Based on its property, this method also obtains good performance in handwriting and hand gesture recognition [74, 46].

1.3 Framework

1.3.1 System Overview

This thesis proposes an integrated framework for hand gesture recognition using multiple cameras. Two calibrated cameras are fixed on a table monitoring at the same area. Figure 1.2 shows the setup of the cameras. Figure 1.3 uses a flowchart illustrate how the system works.

The frames captured from both cameras are filtered by skin color classifier to detect skin-colored blobs. If skin-colored objects are inside the predefined initial areas, the system will take the blobs as a hand and start tracking. At the same time, the system also samples hand images and uses SIFT to extract features on hand images, and compares the sampled image with saved templates for posture recognition. For trajectory recognition, the system records the hand moving trajectory between two points where the hand is static. The trajectory between starting and ending points is recognized by DMP. Combined with the posture recognition result, a gesture vector is obtained for gesture recognition.
1.3.2 Software Environment and Hardware Setups

For experiment setup, two Sony EVI-D100 PTZ color video cameras [25] are employed. Figure 1.4a shows the model of the EVI-D100. It is a high quality Charge-Coupled Device (CCD) camera that combines a high-speed, quiet pan-tilt with a wide-angle view and 40× zoom (10× optical + 4× digital). Live NTSC format images with a resolution of 640 × 480 can be achieved at a full frame rate of 25 frames per second (FPS) by Euresys Picolo PCI frame grabbers (Figure 1.4b). The cameras are working under the consistent lab lighting condition and an Intertec Desk Lamp (20W, 12V) is adopted when we need a strong light.

The desktop workstation runs Microsoft Windows 7, and the algorithms are programmed in both Visual Studio 2010 with Open Source Computer Vision Library (OpenCV) 2.9 and Matlab R2013b. All the training data and templates used in this thesis are collected by our system. A brief description of the software and hardware for the experimental studies is given in Table 1.1.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop Computer</td>
<td>Windows 7 Professional</td>
</tr>
<tr>
<td>Sony EVI-D100 PTZ Cameras</td>
<td>Visual Studio 2010</td>
</tr>
<tr>
<td>Euresys Picolo PCI Frame Grabbers</td>
<td>OpenCV 2.9</td>
</tr>
<tr>
<td>Intertek Desk Lamp (20W, 12V)</td>
<td>Matlab R2013b</td>
</tr>
</tbody>
</table>
1.4 Contributions and Layouts

Hand gesture recognition is a complex problem. It is challenging for various disciplines including hand segmentation, hand tracking, posture recognition and tracking trajectory recognition. The main contributions of this thesis are:

- Hand segmentation with clutter and moving background.

Making a gesture in front of the chest is the most natural way for users. To achieve such a natural HCI, we need to deal with hand segmentation before a cluttered and potentially moving background. Based on some reasonable assumptions, we introduced an approach based on fixed-range CbCr for skin color segmentation and hand detection to deal with this problem (Section 2.1).
• Proposed a method to solve the hand occlusion problem based on reasonable assumptions.

Some gestures involve two hands, and thus hand occlusion occurs. To continue to track hands and build the trajectories, we developed an approach based on the Kalman Filter to track the hand even if the hand is occluded (Chapter 3).

• Posture recognition based on multiple cameras.

Unlike other systems that recognize hand posture based on a single view, a multi-view posture recognition method is proposed in this thesis. The posture is recognized by its front and side views Section 4.2.

• Proposed a concept to recognize hand gestures as one entirety.

Previous works recognize hand posture and trajectory separately. We proposed a concept called gesture vector which includes posture and trajectory information in a vector. Each gesture is represented by the vector for gesture reignition Section 4.3.

This thesis is organized as follows. Chapter 2 introduces the preparation work for hand gesture recognition, including camera calibration and the skin color classifier for hand segmentation. Multiple hand tracking and trajectory reconstruction are addressed in Chapter 3. Chapter 4 proposes the methods for hand gesture recognition. Also, the hand posture
recognition using SIFT and trajectory recognition using DMP are introduced. Chapter 5 presents the discussions, concluding remarks and future work.
Chapter 2

Preliminaries

This chapter presents preliminary results and analysis which are important to the follow-up chapters. Two preliminaries, hand segmentation and 3D coordinate reconstruction, are introduced. Section 2.1 introduces and compares several approaches to skin color segmentation and the scheme of extracting the hand area from the background. Experimental results are also included. The method and results are reviewed for 3D coordinate reconstruction of a point in world space using two calibrated cameras in Sections 2.2 and 2.3.

2.1 Hand Segmentation

For vision-based gesture recognition, efficient hand segmentation is the key to successful hand tracking and posture recognition [32]. There are a large variety of hand appearances in different postures, angles and orientations (Figure 2.1). Color cue is an efficient tool to distinguish the hands from the background. However, segmenting the hand from a cluttered background is very challenging [26] because different skin colors exist among different people where the color of skin can also change under different illumination. In this section, a proper color space to represent skin colors is explored; a position and size constraint is added to locate the hand area.

2.1.1 Skin Segmentation

If the background is uni-colour or distinct colors from the hand area, the hand area can be segmented by thresholding the background color. For cluttered backgrounds, there are multiple colors included inside the camera view. Human skin has relatively consistent
colors which are distinct from the colors of many objects [58]. Therefore, skin color can be an essential cue to separate the hand area from the background.

A suitable color space and a classification algorithm are essential for successful and efficient skin segmentations. Skin segmentations have adapted many color spaces in the previous research [64, 12, 12, 39, 58]. Red-Green-Blue (RGB) color space is sensitive to illumination variations, which is less efficient for hand segmentation. Hue-Saturation-Value (HSV) [64] and YCbCr [12] color spaces are more robust compared to RGB, thus widely used in skin segmentations with various lighting conditions.

Also, there are different classification algorithms, such as piecewise linear classifiers [12, 70], Bayesian classifier with the histogram technique [39] and Gaussian classifiers [39, 51], for skin segmentation. An analysis and comparison of skin segmentation using color pixel classification is done in [58]. The Bayesian RGB, 3D Gaussian mixture (RGB) and fixed-range CbCr classifiers all produced good performances. We reproduced the results of these three skin color classifier performance using our system setup. The Bayesian RGB classifier and 3D RGB Gaussian mixture classifier are trained using the methods and data reported in [39]. The fixed-range CbCr classifier \((77 \leq Cb \leq 127 \text{ and } 133 \leq Cr \leq 173)\) is obtained using the method reported in [12]. Figure 2.2a shows a typical frame in a gesture video. The skin segmentation results using Bayesian RGB, 3D RGB Gaussian mixture and fixed-range (CrCb) classifiers are shown in Figures 2.2b, 2.2c and 2.2d respectively. The fixed-range CbCr classifier gave the best result among the three classifiers, from the accurate skin segmentations represented in the Figure 2.2.

We tested the fixed-range CbCr classifier on several other sample gesture frames taken with different skin colors, lighting conditions and backgrounds. Table 2.1 shows the segmentation results. Room light stands for the fluorescent lamp equipped in the lab. The
strong light equals the room light plus an LED lamp (specifcics can be found in Table 1.1). As you can see, all the hand areas are well-segmented in Table 2.1.

Based on its reliable performance, the fixed-range CbCr skin classifier is adopted for skin segmentation in our system which is designed to deal with cluttered and moving backgrounds. However, this skin segmentation approach relies heavily on color cue, which requires the background to have distinct colors from the skin color of a human hand, or it will be unsuccessfally. Figure 2.3 shows an example when the background contains a large skin-color-like area. The skin-color-like background is segmented as the hand area falsely. The hand tracking and posture recognition of our system are based on the segmented hand area. Thus the system requires less skin-color-like objects in the background. Also, too much exposure of the skin surface, such as arm and neck skin surfaces, will affect the system’s performance. Our system requires the hand gesture to be performed in front of a background with less skin-color-like object and skin surfaces other than hand. Under such condition, the fixed-range CbCr were tested on 50 gesture video frame taken under difference illumination and background, all the hand areas can be successfally segmented.

2.1.2 Hand Area Extraction

After applying the fixed-range CbCr, some background noises are left in the segmentation results (small blobs showed in the segmentation results in Table 2.1). By adding an upper size constraint on the size of the segmented blobs, all the noise blobs smaller than the constraint will be eliminated. For the system setup, the threshold is set at 3000 pixels. All the color blobs with a size smaller than this value will be filtered out. Several examples are shown in Figure 2.4. Figures 2.4a, 2.4b and 2.4c are the images after skin
Table 2.1: The skin segmentation results of fixed-range CbCr classifier.

<table>
<thead>
<tr>
<th>Image acquire condition</th>
<th>Original image</th>
<th>Segmentation results</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Room light</td>
<td><img src="image1.png" alt="Original Image" /></td>
<td><img src="image2.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Lighter skin color</td>
<td><img src="image3.png" alt="Original Image" /></td>
<td><img src="image4.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Strong light</td>
<td><img src="image5.png" alt="Original Image" /></td>
<td><img src="image6.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Lighter skin color</td>
<td><img src="image7.png" alt="Original Image" /></td>
<td><img src="image8.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Room light</td>
<td><img src="image9.png" alt="Original Image" /></td>
<td><img src="image10.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Lighter skin color</td>
<td><img src="image11.png" alt="Original Image" /></td>
<td><img src="image12.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Strong light</td>
<td><img src="image13.png" alt="Original Image" /></td>
<td><img src="image14.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Darker skin color</td>
<td><img src="image15.png" alt="Original Image" /></td>
<td><img src="image16.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Room light</td>
<td><img src="image17.png" alt="Original Image" /></td>
<td><img src="image18.png" alt="Segmentation Results" /></td>
</tr>
<tr>
<td>• Darker skin color</td>
<td><img src="image19.png" alt="Original Image" /></td>
<td><img src="image20.png" alt="Segmentation Results" /></td>
</tr>
</tbody>
</table>
Figure 2.3: Skin segmentation results by fixed-range CbCr classifier when the background contains big skin-color-liked area other than hand. (a) The original image. (b) The segmentation result.

Segmentation and Figures 2.4d, 2.4e and 2.4f are the corresponding results after adding a size constraint to the segmentation areas. It can be seen that only skin areas are left in the image. The next step would be to separate the hand areas from other skin areas.

Separating a hand blob and other skin areas is another challenging task due to their similarity in color, size and shape. In general, there are no distinct differences in RGB or YCbCr values for such two areas. In order to address this problem, we introduced a position constraint to locate the hand areas. During the initialization step, a position constraint
is applied to locate the hand area. After the initialization and segmentation for the hand detection, the hand tracking will be activated, without the position constraint running in the later tracking procedures. After the initialization, the hand tracking algorithm can obtain the spatial position information even with the hand overlapping the neck and face areas. The details, discussions and results of the hand tracking controller will be presented in Chapter 3.

2.2 Two Camera Setup and Calibration

The majority of the vision-based gesture recognition utilized a single camera to capture the hand motion and appearance. The hand moving trajectory in the 3D space is projected onto a 2D image plane. The depth information is lost after the projection. However, in most cases, the 3D trajectory is needed as a part of the gestures. For example, when the hand is moving towards the camera, the projected trajectory on an image plane produces little information about the actual motion of the hand. To overcome this deficiency, we utilize a multi-camera setup to capture the motion of the hands and further reconstruct their 3D coordinates for trajectory recognition.

Two calibrated cameras are employed, to capture hand motion made in the overlapping field of view, to reconstruct 3D hand motion trajectory. The setup of the two cameras is shown in Figure 2.5. The relationship of coordinates between the cameras and world are marked in Figure 2.6. Based on the pinhole camera model (Appendix B.1), the relationship between a point \( Q \) in the world coordinates and its projection \( q_1 \) on the image plane of camera \( C_1 \) is shown in Equation 2.1. The relationship between projection \( q_2 \) in the camera \( C_2 \) and the same point \( Q \) is formulated in Equation 2.2.

\[
q_1 = K_1 [R_1 | t_1] Q. 
\]  
\[
q_2 = K_2 [R_2 | t_2] Q. 
\]  

The intrinsic matrices, \( K_1 \) and \( K_2 \), are separately calculated by Camera Calibration Toolbox for Matlab [9] on both cameras. The extrinsic matrices, \( [R_1 | t_1] \) and \( [R_2 | t_2] \), represent the rotation \( R \) and translation \( t \) between the world coordinate system and the camera coordinate system, \( C_1 \) and \( C_2 \), respectively.
Figure 2.5: The set-up of the calibrated cameras.

Figure 2.6: Schematics of spatial relationship of camera $C_1$ and camera $C_2$ with respect to world coordinates with origin at $O$. 

Table plane
The objective of this section is to reconstruct a point location in world space using its projected points on image planes taken by the two calibrated cameras. In other words, the objective is to reconstruct the coordinates of \( Q \) based on \( q_1 \) and \( q_2 \) in Equations 2.1 and 2.2. This would make Equations 2.1 and 2.2 an over-complete (four equations with three unknowns) equation system (Appendix B.4). The coordinates of \( Q \) that have three unknowns can be calculated. Based on the geometry of our experimental setup, the specific values of \( K_1, K_2, R_1, R_2, t_1 \) and \( t_2 \) are computed as:

\[
K_1 = \begin{bmatrix} 561 & 0 & 338 \\ 0 & 561 & 230 \\ 0 & 0 & 1 \end{bmatrix}, \quad K_2 = \begin{bmatrix} 560 & 0 & 339 \\ 0 & 560 & 220 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)
\]

\[
R_1 = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}, \quad t_1 = \begin{bmatrix} 0 \\ 9 \\ 44 \end{bmatrix} \quad (cm) \quad (2.4)
\]

\[
R_2 = \begin{bmatrix} 0.707 & -0.707 & 0 \\ 0 & 0 & -1 \\ 0.707 & 0.707 & 0 \end{bmatrix}, \quad t_2 = \begin{bmatrix} 0 \\ 9 \\ 40 \end{bmatrix} \quad (cm) \quad (2.5)
\]

The 3D coordinate reconstruction based on the projected location on two image planes can be achieved based on these matrices; results are presented in the following section.

2.3 Result for 3D Coordinate Reconstruction

A box made by transparent plastics is placed in the overlapping view of both cameras. The dimensions of the box are marked in Figure 2.7. Figure 2.8 shows the two different views captured by the cameras. Table 2.2 lists the coordinates of eight vertices of the box in both image planes, and the reconstructed 3D coordinate in the physical world. The ground truths of the eight box corners are also listed in this Table 2.2.

To eliminate the displacement error between coordinate systems, the relative locations between the eight corner points on the box are calculated and compared with their ground truths. The results are listed in Table 2.3. This table shows that the error is quite small after eliminating the measurement error between coordinate systems.
The key information for trajectory recognition is the relative location information between captured points during hand motions, not the actual location in the world coordinate system. Therefore, based on the experiment results, we can conclude that the reconstructed 3D hand trajectory based on our approach is accurate enough for trajectory recognition.

### 2.4 Discussion

This chapter introduced two preparation steps for hand gesture recognition: hand segmentation by the fixed-range CbCr classifier and 3D coordinate reconstruction based on two calibrated cameras. The performances of hand segmentation and 3D coordinate reconstruction were also evaluated in this chapter. Hand segmentation with size constraint and position initialization can efficiently detect hand area from cluttered and moving backgrounds. Such a skin segmentation approach works for people with different skin tone.
Table 2.2: The 3D coordinates of eight corner points on the plastic box in the world coordinate system recovered by their projected points on the image plane of two cameras.

<table>
<thead>
<tr>
<th>$C_1$ (pixel)</th>
<th>$C_2$ (pixel)</th>
<th>Recovered 3D Coordinate (cm)</th>
<th>Ground truth (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1 = [334, 334]^T$</td>
<td>$b_1 = [293, 375]^T$</td>
<td>$[-3.9, 0.3, -0.5]^T$</td>
<td>$[-3, 0, 0]^T$</td>
</tr>
<tr>
<td>$b_2 = [334, 160]^T$</td>
<td>$b_2 = [291, 173]^T$</td>
<td>$[-4.0, 0.4, 12.8]^T$</td>
<td>$[-3, 0, 13.1]^T$</td>
</tr>
<tr>
<td>$b_5 = [335, 335]^T$</td>
<td>$b_5 = [323, 368]^T$</td>
<td>$[-1.2, 0.3, -0.4]^T$</td>
<td>$[0, 0, 0]^T$</td>
</tr>
<tr>
<td>$b_6 = [333, 165]^T$</td>
<td>$b_6 = [322, 182]^T$</td>
<td>$[-1.2, 0.5, 12.6]^T$</td>
<td>$[0, 0, 13.1]^T$</td>
</tr>
<tr>
<td>$b_8 = [463, 333]^T$</td>
<td>$b_8 = [438, 396]^T$</td>
<td>$[-1.4, -9.5, -0.3]^T$</td>
<td>$[0, -9.8, 0]^T$</td>
</tr>
</tbody>
</table>

Table 2.3: The relative location relationship between eight corner points.

<table>
<thead>
<tr>
<th>Points Relationship</th>
<th>Recovered Distance (cm)</th>
<th>Ground truth (cm)</th>
<th>Error (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1 - b_2$</td>
<td>$[0.1, 0.1, 13.3]^T$</td>
<td>$[0, 0, 13.1]^T$</td>
<td>$[0.1, 0.1, 0.2]^T$</td>
</tr>
<tr>
<td>$b_1 - b_4$</td>
<td>$[0.5, 9.8, 0.2]^T$</td>
<td>$[0, 9.8, 0]^T$</td>
<td>$[0.5, 0, 0.2]^T$</td>
</tr>
<tr>
<td>$b_1 - b_5$</td>
<td>$[2.7, 0, 0.1]^T$</td>
<td>$[3, 0, 0]^T$</td>
<td>$[0.3, 0, 0.1]^T$</td>
</tr>
</tbody>
</table>

After the achievement of hand detection using the fixed-range CbCr classifier, a two-camera system was introduced in our setup to provide 3D trajectory information. For 3D coordinate reconstruction, based on the calibrated information of two cameras, the coordinates of a point in the overlapping view of two cameras can be reconstructed based on its pixel values on the projected image planes. The experiment proves such an approach can successfully recover the relative location information between points on the gesture trajectory in world space.
Chapter 3

Hand Tracking and Trajectory Reconstruction

The objective of this chapter is to introduce the methods of locating and tracking hands in image frames. In the previous chapter, hand blobs were segmented, where in some non-ideal cases, other skin areas such as face and neck were also included as a part of the segmentation process. Due to the similarities of the hand area and other skin area shared by various attributes, such as shape and color, a scheme involving position constraint is adopted to separate hand blobs from other skin areas. Kalman Filter (KF) is employed to track the hand and reduce the influence of other skin areas on posture and trajectory recognition. In addition, with KF and reasonable assumptions, the occlusion problem while tracking with two hands is solved. Then, based on the tracking results, hand movement trajectories can be reconstructed.

The rest of this chapter is organized as follows: a scheme for initialization and tracking are introduced in Section 3.1. A design of KF for hand tracking is described in Section 3.2. Section 3.3 presents a scheme to improve tracking accuracy. Section 3.4 presents experiment results of hand tracking, both one-hand and two-hand cases are included. The results for 3D trajectory reconstruction and smoothing is presented in Section 3.5.

3.1 Hand Tracking Initialization

Recalling from Chapter 2, after applying the fixed-range CbCr skin classifier on the video frames, background such as face and other skin-color-liked areas were segmented
Figure 3.1: (a) and (c) are sample frames from camera $C_1$ of a two-hand gesture; (b) and (d) are the corresponding sample frames taken from camera $C_2$. The red bounding boxes in (a) and (b) represent the initial area for left hand; the green bounding boxes are for the right hand. The crosses in (a) and (b) show the centroids of the detected skin-color blobs. Only the centroids of the blobs which enter into the initial areas are taken as hand blobs and labeled. The green bounding boxes label the right hand in (c) and (d); the red bounding boxes label the left hand.

to the foreground (Figure 2.4) in some cases. There is no distinct difference between the hand blobs and other skin blobs in color, shape and size. Considering such noises usually appear in a higher position than the hand, a position constraint is added in the initial step. For each camera view, two initial areas are predefined in Figures 3.1a and 3.1b. The green bounding boxes are the initial areas for the right hand detection and labeling, and the red bounding boxes are for the left hand.

For hands detection and labeling, the system requires the hands enter into the corresponding initial areas. Figures 3.1a and 3.1b show an example of hand positions before making a two-hand gesture. When the right hand (represented by red crosses) is inside the green bounding boxes in both camera views, it will be taken as detected and labeled.
Table 3.1: The logic table of initialization.

<table>
<thead>
<tr>
<th>Skin color blob detected in green bounding boxes in both camera views</th>
<th>Skin color blob detected in green bounding boxes in both camera views</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Two-hand gesture</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Right-hand gesture</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Left-hand gesture</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Get next video frame</td>
</tr>
</tbody>
</table>

as the right hand. Similarly to the left hand, it will be detected and labeled as the left hand if the centroids (the yellow crosses) of the hand are inside the red bounding boxes in the camera views. For the skin-color blobs which are outside the initial areas, they will not be taken as hand blobs (the blue crosses) and will not be tracked. After the hands (either one hand or both hands) are detected and labeled, the system will start tracking and remove the position constraint (Figure 3.1c and 3.1d). So, the hands can still be tracked when they move outside of the initial areas.

Also, in this initialization step, the system will decide if the video is a one-hand gesture or two-hand gesture. If hands are detected in both initial areas (Figure 3.1), it will be taken as a two-hand gesture. If only one hand is detected in one initial area (either the left one or right one), the system will take it as a one-hand gesture. Figure 3.2 shows an example of a one hand gesture. Similar to the two-hand case, the centroids of the detected color blob are inside the left hand initial areas in both camera views (the red bounding boxes in Figure 3.2a and 3.2b). Because there is no hand blobs detected in the right hand areas, the system will take the gesture as a one-hand gesture and only track the left hand in later steps (Figure 3.2c and 3.2d).

Table 3.1 shows the logic table of how the system decides what gesture is being made in the video based on the blobs detected in the initial step. After initialization, the hand blobs are located and labeled. Also, how many hands are involved in the gesture can be decided, which can help with gesture recognition in later steps.
Figure 3.2: (a) and (c) are sample frames from camera $C_1$ of a one-hand gesture; (b) and (d) are the corresponding sample frames taken from camera $C_2$. The red bounding boxes in (a) and (b) represent the initial area for left hand; the green bounding boxes are for the right hand. The crosses in (a) and (b) show the centroids of the detected skin-color blobs. The blobs which centroids enter the left initial area are labeled as left. Since there is no blob detected in the right hand initial areas, only the left hand is being labeled and tracked in (c) and (d).
3.2 Kalman Filter for Hand Tracking

KF is an estimator developed by Kalman in 1960 [41] that uses measurement states observed over time to predict or estimate the next state in linear system. It has been extensively used in the computer vision community for object tracking [85, 22, 87]. For nonlinear system, other forms of KF have been developed, such as extended Kalman filter [52] and unscented Kalman filter [40]. Here, we assume that hand motion model can be linear at a given instance where we have utilized KF for estimating its position based on the current observation in presence of noise For each segmented frame, the hands are taken as the objects of interest, and other segmented skin-color areas are taken as noise in the background. The hand location is represented by the center of the hand blob. The objective is to track the hand in every skin-color-segmented video frame as accurately as possible.

In the tracking system, the hand state is simplified as the position and velocity of the centroid of the hand blob. The hand state $x$ at time $t$ in a video frame is modeled as:

$$x_t = [u_t, v_t, u'_t, v'_t]^T,$$

where $(u_t, v_t)$ represents the pixel values of the centroid of the hand blob, which is calculated by the yellow bounding box in Figure 3.3. The bounding box is generated by the utmost points in four directions, including: up, down, left, and right, of the hand blob. The point is represented by a vector $[u', v']^T$ where $u'$ and $v'$ are the pixel values of the bounding box center which is represented by a red cross. $u'_t$ and $v'_t$ indicate the velocity of the hand blob in the direction of $u$ and $v$ respectively at frame $t$. 

![Figure 3.3: The hand location is represented by the pixel value of the centroid of the segmented hand blob.](image-url)
The KF works in two steps: the prediction step (updates) and the correction step (measurements). For each video frame, the hand location is predicted from previous frames. The KF model is modified according to the measurements. Figure 3.4 displays the conceptual diagram of iterations in KF. Since KF is a recursive estimator, in the prediction step, the computation of the estimated state for the next time step only requires the estimated state and the measurement of the current time step. No history of measurements and estimations is required.

$A$ is a $4 \times 4$ the state transition matrix which models the transition relationship between the current state $x_{t-1}$ and the next state $x_t$. The measurement matrix $H$ is a $2 \times 4$ matrix which maps the state $x_t$ into the observation vector $z_t$. The KF motion model in this system is simplified as a constant velocity model. The transition model $A$ and measurement model $H$ remain the same during the whole tracking process. In the prediction step, the prior estimated state $x_t^-$ and the prior estimated error covariance $P_{t-}$ can be calculated by the following equations, where $W$ is the process noise in Gaussian distribution with the noise covariance $Q$.

$$x_t^- = Ax_{t-1}^- + W_{t-1}$$  \hspace{1cm} (3.2) \\

$$P_{t-} = AP_{t-1}A^T + Q$$  \hspace{1cm} (3.3) \\

$$p(W) \sim N(0, Q)$$  \hspace{1cm} (3.4)
The actual hand location needs to be measured to correct the prior state $x_t^-$ and error covariance $P_t^-$. In order to generate and improve the posterior state $x_t$ with better accuracy, the measurement $z_t$ can be calculated based on our constant velocity assumption as follows:

$$z_t = Hx_t + V_t \Rightarrow \begin{bmatrix} u_{\text{meas},t} \\ v_{\text{meas},t} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_t \\ v_t \\ u'_t \\ v'_t \end{bmatrix} + V_t$$

(3.5)

$$p(V_t) \sim N(0, R)$$

(3.6)

where $V_t$ is the measurement noise in Gaussian distribution with noise covariance $R$. The posterior state $x_t$ and error covariance $P_t$ can be obtained from the correction equations.

$$K_t = P_t H^T (HP_t H^T + R_t)^{-1}$$

(3.7)

$$x_t = x_t^- + K_t (z_t - H_t x_t^-)$$

(3.8)

$$P_t = (I - K_t H) P_t^-$$

(3.9)

$K$ is the gain matrix.

### 3.3 Area Constraint for Hands Tracking

During the hand tracking, the size of the bounding is updated based on the segmented hand blob area. In some cases, the hand area overlaps with the segmented skin-color areas such as face, neck or other background areas with a similar color. The hand and the skin-color areas will be segmented as a color blob. The size of the bounding box will be expand in that frame. In this case, the center of the bounding box cannot reflect the actual location of the hand area.

Figure 3.5 shows an example of this case. Two adjacent frames are clipped from a hand tracking video. The hand in Figure 3.5a is well segmented and tracked. The yellow bounding box indicates the hand tracking result. Its centroid, which is marked by a red cross, represents the hand location. When the hand overlaps with the background noise, the hand area and the background noise will be segmented as one color blob. The bounding
Figure 3.5: The bounding box expands when the hand blob is overlapped with background noises. The red crosses show the location of hand palm and the green one shows the center of the enlarged bounding box which is mis-marked as the hand location.

Figure 3.6: When hands contact and occlusion happen, both hand is included in one bounding box.

box will include the enlarged blob as a hand (Figure 3.5b). The centroid of the bounding box (green cross) cannot reflect the real location (red cross) of the hand. Such error will affect the results of trajectory reconstruction and recognition.

Also for two-hand tracking, when the hands make contact with each other or occlude one behind the other (Figure 3.6), a similar problem will arise. The KF will include both hands inside one bounding box and take the other one as lost.

To solve this problem, an area constraint is added to the bounding box. Because for regular hand motion, the size of the hand blob will not change suddenly, usually the increase and decrease ratio of the blob size is in the range between 0.8 to 1.2 of two adjacent frames (Figure 3.7). The area of a bounding box at time $t$ is represented by $S^t_b$. A constraint on the
Figure 3.7: For hand motion without contaminated by background noise, the increase and decrease ratio is in the range of 0.8 and 1.2 between two adjacent frames.

Figure 3.8: (a) Hand tracking result before adding the area constraint; (b) hand tracking result after adding the area constraint.
size ratio is added to the system (Equation 3.10).

\[
0.8 < \frac{S_{b}^{t+1}}{S_{b}^{t}} < 1.2
\]  

(3.10)

This means \( S_{b}^{t} \) will not increase or decrease largely between two adjacent frames. When the ratio is bigger or smaller than the thresholds, it means the hand is either overlapped with background noise area or the other hand area. Based on observation, the KF filter will stop updating the bounding box and take predicted tracking result as the hand location (Figure 3.8b). Therefore, the background noise area is excluded outside the bounding box, and the center of the bounding box can represent the hand location again. When the hand area is not overlapping with the background noise, base on observation, the KF will keep updating the bounding box.

### 3.4 Experiment Result for Hand Tracking

In this section, the experiment results for the KF hand tracking are achieved. Both one-hand and two-hand tracking taken at different speed and conditions are included. Figure 3.9 shows the setup of the two cameras, and example frames taken from camera \( C_{1} \) and \( C_{2} \) are also included. Frames taken from camera \( C_{1} \) and camera \( C_{2} \) are also referred to the front view and the side view respectively for the rest of this thesis.

![Figure 3.9: The system setup of two cameras. The example frames taken by camera \( C_{1} \) and \( C_{2} \) are listed on the left of this image.](image)
Figure 3.10: Sample frames from gesture video with a hand moving in slow speed and its KF tracking result. The fist column ((a),(e),(i),(m)) and the second column ((b),(f),(j),(n)) display the original frame clipped from camera $C_1$ and camera $C_2$ respectively. The tracking result for both cameras views are showed in the third column ((c),(g),(k),(o)) and fourth column ((d),(h),(l),(p)).

3.4.1 Single Hand Tracking

For single-hand tracking, when there is none or fewer background noises, the hand can be perfectly tracked in both camera views. Figure 3.10 shows the tracking results when the hand is moving at a slow speed. The hand blob is successfully tracked in each video frame. The frame number is marked in the upper-left corner. Four columns from the left to the right list the original frames captured by camera $C_1$, the original frames captured by camera $C_2$, tracking results for camera $C_1$, and tracking results for camera $C_2$.

Figure 3.11 shows the tracking performance of KF when the hand is moving at a faster
Figure 3.11: Sample frames clipped from gesture video with hand moving with fast speed and direction change suddenly. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_1$. The second row ((e),(f),(g),(h)) shows the tracking result.

speed. In this example video, the user is moving their hand at the fastest speed from the left to the right and then moving back without a pause. Even though the fast speed of the hand makes the hand image blurry, the segmented hand area is tracked successfully in each video frame. Figure 3.11g shows the frame clipped after the hand suddenly changes the direction it is moving of, the hand is still well tracked.

For the case that involves background noises, Figure 3.12 shows an example of the tracking result. With initialization, the hand blob is successfully located and tracked when the hand blob is not overlapping with any background noise area (Figure 3.12a). In the next frame (Figure 3.12e), the hand is overlapped with a noise area, instead of expanding the bounding box to include the whole color blob, the predicted KF tracking result is adopted as the hand location. In the next frame, the overlapping is over and the bounding box is updated with the size and location of the hand blob. The KF tracking performed well in such a case, but if the hand area is overlapped with the background noise for a long time, the tracking results of the hand blob will drift away based on the predicted hand speed. Figures 3.13 shows an example. The hand blob is tracked in Figures 3.13a, 3.13b and 3.8b. When the hand is overlapping with the background noise area for a long time, the system looses track of the hand area.
Figure 3.12: Sample frames clipped from gesture video with hand moving and overlapping with background noise. The first row ((a),(b),(c)) shows the original frame captured by camera $C_2$. The second row ((d),(e),(f)) shows the tracking result.

Figure 3.13: Sample frames clipped from gesture video with a hand blob overlapping with a background blob. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_1$. The second row ((e),(f),(g),(h)) shows the tracking result.
3.4.2 Two Hands Tracking

In the two-hand case, we present the results of two-hand tracking with respect to the hand relationships in three categories: no contact, contact without overlapping, and occlusion. In this section, the background noise is not considered and we assume the gesture of the hands remain constant during the occlusion.

a) No contact case:

In the no contact case, it is assumed that the blobs representing the two hands in the camera views are not contacting each other. The tracking result is shown in Figure 3.14. The first row and second row (Figure 3.14(a-f)) are the original frames taken by camera $C_1$ and camera $C_2$. The third and fourth row (Figure 3.14(g-l)) are the tracking results of camera $C_1$ and camera $C_2$ respectively. The number in the upper-left corner shows the frame number. It can be seen in Figure 3.14, the left hand is labeled as 1 and the right hand is labeled as 2. They can constantly be tracked.

b) Contact without occlusion:

Figure 3.15 shows the tracking result when contact happens between the hands. The first row (Figure 3.15(a-d)) is the original frame clipped from the gesture video and the second row (Figure 3.15(e-h)) is the tracking result. Figure 3.15e is the frame before the contact happens, there are two separate bounding boxes for two separate hand blobs, track 1 and track 2. In the next frame (Figure 3.15f), the contact is happening, and the two hand blobs are combined as one. Instead of using a big bounding box to include both hands, the area constraint makes the system take the predicted position of the bounding box 1 and 2 to represent the hand positions. Both track 1 and track 2 are predicted bounding boxes and there exists no measurement information to correct the KF model. Although the hand blobs are static at the instant when contact happens, the bounding boxes which indicate the tracking results are still moving according to the KF prediction model. Figure 3.15g shows when the hands are static, while the bounding box 1 and 2 are moving. Figure 3.15h shows the result when the two hands separate from each other, the detection of the hand blobs updates the position and size of both bounding boxes.

c) Occlusion:

Similar to the contact case, area constraint has been included to handle occlusion cases. Figure 3.16 shows sample frames from a gesture video of both hands. In this
Figure 3.14: Sample frames clipped from gesture video of two hand moving with no contact. The original frame captured from camera \( C_1 \) and camera \( C_2 \) are shown in the first row ((a),(b),(c)) and second row ((d),(e),(f)). The third ((g),(h),(i)) and fourth row ((j),(k),(l)) display the tracking result.
Figure 3.15: Sample frames from a gesture video with two hands contact with each other. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_1$. The second row ((e),(f),(g),(h)) shows the tracking result.

video, the left hand is moving from the right to the left while the right hand is moving from the left to the right. The right-hand becomes occluded by the left hand for several frames. When the occlusion happens, the left hand is detected since the area of the left-hand blob does not increase suddenly. The right hand is occluded by the left and lost track of, so the system will predict the right-hand position based on its former time state and take it as the right-hand position (Figure 3.16(e)). When the occlusion ends, and the detection of the right hand is recovered, the system will resume the tracking of the right hand again.

3.5 Trajectory Reconstruction and Smoothing

3.5.1 Trajectory Reconstruction

From Section 2.2, the 3D coordinates of a point which are inside the overlapping area of the camera views can be reconstructed by the pixel values on projected image planes. For each time state $t$, the hand center $[u^t_1, v^t_1]$ and $[u^t_2, v^t_2]$ can be extracted from both camera views. Substitute $[u^t_1, v^t_1]$ and $[u^t_2, v^t_2]$ with $q_1, q_2$ in Equations 2.1 and 2.2, the hand coordinates at time $t$, $Q^t$ in the world space can be reconstructed.
Figure 3.16: Sample frames from a gesture video which right hand is fully occluded. The first row ((a),(b),(c),(d)) shows the original frame captured by camera $C_1$. The second row ((e),(f),(g),(h)) shows the tracking result.

Figure 3.17a shows a reconstructed single hand moving trajectory. The reconstructed trajectory of two hands in different cases: two hands have no contact (Figure 3.17b), contact without occlusion (Figure 3.17c) and occlusion (Figure 3.17d) are also shown. The trajectory of the right hand and left hand are shown in red and blue respectively.

3.5.2 Trajectory Smoothing

The reconstructed trajectory in Figure 3.17 is not smoothed. An example is shown in Figure 3.18a. The trajectory starts from the green point, follows the blue arrows and ends at the red point. The fluctuations on the trajectory are generated based on the size changing of the tracking box. For instance, if the hand area is occluded with another skin-like area, the center of the segmented color blob cannot reflect the hand center. The reconstructed location will have an error from the real location and make such fluctuations on the trajectory.

To eliminate such a noise effect, we adopt a smoothing method for reconstructed trajectories. Each trajectory is presented by as three spatio-temporal in $x$, $y$ and $z$ directions. Figure 3.18b shows the projected information of circle trajectory over time in $x$, $y$ and $z$ directions.
Figure 3.17: Examples of reconstructed 3D moving trajectory. (a) single hand moving trajectory; (b) the trajectory of two hand moving without contact; (c) the trajectory of two hand moving with contact; (b) the trajectory of two hand moving with occlusion happens. Red crosses show the trajectory of the right hand and blue crosses for the left hand.

Figure 3.18: (a)Reconstructed 3D trajectory when hand moving in a circle. (b) The hand moving trajectory is represented in $x$, $y$ and $z$ directions.
We compare two smooth methods on the trajectory: locally weighted scatterplot smoothing (LOESS) [16] and the robust version of LOESS (RLOESS). LOESS is Locally weighted scatterplot smooth using the least squares quadratic polynomial fitting (Appendix E). RLOESS is a robust version of LOESS smoothing that assigns lower weight to outliers in the regression.

Figure 3.19 compares the results of these two methods. In this figure, span is a percentage of the total number of data points, less or equal to 1. For example, if the span is equal to 0.1, that means 10% of data points are included for each smooth calculation. The red dots in Figure 3.19 represent the original data points along $x$ direction of the 3D reconstruction circle trajectory. The blue curve represents the smoothed data points. Based on observation, if span value are too big (Figure 3.19b and 3.19d), the smoothed curve does not fitting well with the original data points. If the span value is small, the effect of outliers in the original data cannot be completely eliminated (Figure 3.19a). In conclusion, RLOESS is superior to LOESS for this case. In our system, we adopt RLOESS with a span of 0.1 to eliminate small fluctuations on the reconstructed hand trajectories. Figure 3.20 shows the smoothed trajectory by RLOESS, where the fluctuation has been erased.

3.6 Discussions

This chapter introduced the position and area constraints on segmented color blobs for hand locating and tracking. In the initialization step, it is crucial for the hand blobs to appear inside the predefined areas. Otherwise the hand cannot be located and labeled. The KF is employed for hand tracking. Our system added an area constraint on the KF tracking which performed very well for one-hand and two-hand tracking. It only fails when the hands are overlapped with the background for a long time, or are in contact with each other for a long time, which both cases will not happen in the gesture videos we collected. To further explore the performance of the tracking results is to extend the definition of the state vector by including the dimensions of the bounding box and their rate of changes. Although, the KF tracing achieved good tracking performance in our system, when hand gesture movements become more complex in the future, a more accurate and comprehensive tracking method needs to be explored. In addition, this chapter also displayed the reconstructed 3D hand
Figure 3.19: Comparison of original data points (red dots) and smoothed data points (blue curve) using LOESS (Figure (a) and (b)) and RLOESS (Figure (c) and (d)) with different span. (a) span = 0.1; (b) span = 0.5; (c) span = 0.1; (d) span = 0.5;

Figure 3.20: Smoothed trajectory by RLOESS.
moving trajectories based on the tracking information from both camera views. RLOESS is applied on each reconstructed trajectory to eliminate the background noise further.
Chapter 4

Gesture Recognition

Hand gesture is decomposed into two parts in this thesis: hand postures (sampled hand shape) and the trajectory of hand movements. In this chapter, trajectory and posture recognition are first introduced in Sections 4.1 and 4.2 respectively. Then these two parts are integrated into a gesture vector for gesture recognition.

4.1 Trajectory Recognition

In the previous chapter, methods for hand tracking, trajectory reconstruction and smoothing were presented. Recognition of hand trajectory is a challenging task due to various patterns that hands can make in space and time. For instance, the same intended motion trajectory performed by different people usually have the same spatial pattern. Dynamic movements primitives (DMP) was originally proposed by [65] and further extended in [38] as a method of trajectory control. It shows good performances in [74] for handwriting recognition and in [46] for 2D hand trajectory recognition. Here, we adopt it for 3D hand trajectory recognition.

4.1.1 An Overview of Dynamic Movements Primitives

The dynamic movements primitives (DMP) method models movements with the given start and end states into a set of differential equations. For example, it is capable of encoding the spatio-temporal information of trajectories of the hand movements into a weight vector that is robust with respect to the spatio-temporal variations along the same hand trajectory.
The differential equations that characterize the spatio-temporal evolution of a dynamic system with the given start and end states is given in Equations 4.1 and 4.2. A second-order linear damped spring model with a non-linear function $f$ is added in Equation 4.1 as the forcing term. This non-linear force function can capture the complexities of motion patterns that are made by humans.

\[
\tau \ddot{z} = \alpha_z (\beta_z (g - x) - z) + f, \tag{4.1}
\]

\[
\tau \dot{x} = z \tag{4.2}
\]

In Equations 4.1 and 4.2, $x$, $z$ and $\dot{z}$ represent the position, velocity and acceleration of the hand motion dynamics. $\tau$ is a time constant which represents the trajectory duration and $g$ is a known goal representing the final hand position of the trajectory. For a suitable selection of parameters $\alpha_z$ and $\beta_z$, the forcing term $f$ would decay to zero over time, which allows the system converge to the goal position $(x, z) = (g, 0)$.

The non-linear forcing function $f$ is composed by a set of Gaussian-like basis functions as:

\[
f(y) = \frac{\sum_{i=1}^{N} \phi_i(y) w_i}{\sum_{i=1}^{N} \phi_i(y)} y(g - x_0) \tag{4.3}
\]

where $w_i$ are weights of basis functions and $\phi_i(y)$ are $N$ basis Gaussian-like functions. An example of generating a weight vector by DMP is shown in Appendix F. The forcing term $f$ will vanish along time. A phase variable $y$ regulated by 4.4 is introduced in $f$ to ensure it is vanishing.

\[
\tau \dot{y} = -\alpha_y y \tag{4.4}
\]

The weight vector $w = [w_1, ..., w_N]^T$ in $f$ preserves the shape information of the trajectories. For instance, if $w$ is fixed and other parameters such as the goal state $g$ or time constant $\tau$ changes, the DMP will generate topologically similar trajectories. In other words, similar trajectories would have a similar feature vector $w$ which is called the invariance properties of the DMP model [38]. With such a property, trajectories can be classified based on the weight vectors.

### 4.1.2 Extracting Weight Vector from 3D Trajectory

Given a trajectory in one dimension $Q = (q_0, ..., q_{t-1})$, the dynamics at each point $q_s = (x, z, \dot{z})^T$ denote the state vector at time $s$ that can be calculated based on frame rate, where
$t$ is the time duration of the whole path. To learn the weight vector for a given trajectory, the initial and goal states and also the time duration can be extracted from the trajectory. From Equation 4.1, $f$ can be rewritten as

$$f = \alpha \dot{z} (g - x) - \tau z = \sum_{i=1}^{N} \frac{\phi_i(y) w_i}{\sum_{i=1}^{N} \phi_i(y)} y(g - x_0)$$

(4.5)

The weight vector $w$ can be learned using the locally weighted regression (LWR) as stated in [65].

Before conducting the trajectory recognition, a trajectory instance "Circle" (Figure 4.2d) is selected from the collected trajectories in order to visualize the weight vector and learned DMP models. The acquired hand trajectory is in 3D space, where DMP can be utilized along each projected direction of the world coordinate system. Figure 4.1 shows the learned DMP models and weight vectors with different dimensions ($D_w$) of the trajectory projected in $x$ direction. Figures 4.1a, 4.1c, 4.1e and 4.1g display the original trajectories in green and the learned trajectories by DMP in blue. Figures 4.1b, 4.1d, 4.1f and 4.1h show the corresponding learned weight vectors. It can be seen, when the dimension of the weight vector $N$ increases, the learned trajectory approaches the original trajectory.

4.1.3 Training Stage for Trajectory Classifier Using SVM

The invariance properties of DMP preserve the shape information of the trajectory in weight vectors and can be used for trajectory recognition. In [46], they compare two classification methods: k-nearest-neighbor (k-NN) and SVM. Based on their experiment result, SVM obtains much better accuracy than k-NN. Consider the similarity between our and their work, SVM is utilized for trajectory recognition. The method of SVM is introduced in Appendix D. In our implementation, multi-class SVM training and testing are performed using the LIBSVM library [13].

For trajectory recognition, five classes of trajectories are collected. These five classes consist of "Jump", "Left", "Right", "Circle" and "Forward". Figure 4.2 shows the example trajectories for each class. Eight people (two male and six female) are asked to perform the trajectories, 5 for each class. Over 200 trajectories are collected, and 3/4 of the data set is taken to train the SVM while the rest of the data set are held out for testing. The trajectories in the training and testing data set are performed by different people.
Figure 4.1: (a), (c), (e) and (g) are the comparisons between DMP learned trajectories (in blue) and the original trajectories (in green) along with different weight vectors dimension $D_w$. (b), (d), (f) and (h) show the learned weight vectors.
Figure 4.2: Trajectory samples of five classes. The starting point is marked in green, and the ending point is in red.
The SVM with a linear kernel is trained for trajectory recognition. Table 4.1 gives the 5-fold cross-validation recognition accuracy based on different weight vector dimensions. As the dimension increases, for the same number of training data, the accuracy decreases. This is because the bigger the weight vector dimension is, the more parameters in the SVM need to be decided and more training data is needed. The highest recognition rate is obtained at $D_w = 5$. This is because the classes of trajectories we have collected are relatively simple and well distinguished from each other. For complex trajectories, a weight vector with higher dimensions is needed.

Table 4.1: Recognition accuracy with different number of $D_w$

<table>
<thead>
<tr>
<th>Training trajectory number</th>
<th>Weight vector dimension $D_w$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>150</td>
<td>86.67%</td>
</tr>
</tbody>
</table>

4.1.4 Testing Stage for Trajectory Recognition

Applying the trained SVM on the testing data set, we obtain an accuracy of 88.0%. The recognition results are shown in Table 4.2. The misclassification appears due to the user habit. Because the trajectory "Jump" and "Push" are both moving forward and for some cases if the trajectory "Jump" is made with a smaller radian, it will be misclassified as "Push". Also for the trajectory "left", sometimes it is performed with an angle, and the SVM will recognize it as "Push" as well. But for the trajectory "Circle", which is quite distinctive from other classes, it can be classified perfectly.

Table 4.2: Recognition result of testing trajectories.

<table>
<thead>
<tr>
<th>Testing Trajectories</th>
<th>Jump</th>
<th>Left</th>
<th>Right</th>
<th>Circle</th>
<th>Push</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Left</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Right</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Circle</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Push</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

| Accuracy             | 88.0% (44/50) |

47
4.2 Posture Recognition

Hand posture recognition is another key part of gesture recognition. Our system samples hand postures and follows the posture changing along the hand trajectory (Figure 4.3). Usually, there are three steps for posture recognition: hand image acquisition, feature extraction and classification. For our system, hand image acquisition is done along the trajectory tracking process where the hand postures are already segmented from the background.

Template matching is a simple method of posture recognition, and it is easy to add or remove template classes. To extract features on hand posture, a convex hull on silhouette [44], or fingertip detection using a circular mask as a correlation techniques [43] can be employed. However, these recognition approaches based on hand silhouettes or contours usually require a clean background where the hand can be well segmented. In our case, hand posture is made in a clutter and moving background and segmented using the fixed-range CbCr skin classifier. Sometimes the hand can not be well segmented from the background. In such cases, the hand can be taken as partially occluded. Therefore, a feature detector that is insensitive to illumination variation and partially occlusion is needed. SIFT is such a feature detector and descriptor which is also robust to scale and orientation changes. In addition, it is robust to affine distortion in some ranges, which could benefit posture recognition, since the relative position is kept changing between hand and cameras and causes affine distortion between the input hand postures and posture templates. In our work, SIFT is used for the feature detection and posture recognition. Bag of visual words and SVM are combined for classification.

4.2.1 Scale-Invariant Feature Transform (SIFT)

SIFT is a feature detector developed by D.Lowe [47]. SIFT features are shown to provide robust matching in a range of occlusion and affine distortions, with the addition of noise and changes in illumination. To acquire features at different scales, SIFT convolves Gaussian filters with different variances with the original image, and also the down-sampled images. The difference of Gaussian (DoG) is calculated by subtracting the adjacent images convolved with Gaussians in the same octave. This process is shown in Figure 4.4.
In order to detect the local maxima or minima of difference of Gaussian (DoG), each sample point is compared to its 26 neighbours in a $3 \times 3$ region, 8 in its own image and 9 in the scale above and below (Figure 4.5). A point is selected as a feature point only if it is larger or smaller than all of the other 26 neighbours.

For the stability of feature points, once a feature point is detected by the above method, a threshold on minimum contrast is performed on these feature points between its neighbours. Also eliminating edge response is applied on the DoG since it has a strong response to edges. In this way, the feature points with strong contrast and within the image remain for feature point matching.

By assigning a constant orientation to each feature point based on local image properties, the feature point is invariant to rotation. This orientation information is also used to build a feature point descriptor that is a vector containing 128 non-negative elements. The resulting vector is defined as SIFT keys and are used in a nearest-neighbours approach to finding the matching points and detecting the same object between images. Figure 4.6 shows the feature points detected by SIFT in green circles on two hand palms. The size of the circles presents the scale of feature points, and the radius of each circle represents the direction of each feature point. A set of 36 matching points is found between these two posture examples and connected with blue lines.

Our work is designed to recognize six targeted hand postures: "Palm", "V", "Point", "Six", "Fist" and "Eight" (Figure 4.7). To prove the feature detected on these postures have
Figure 4.4: For each level of the octave, the original image is convolved with Gaussian in different variance. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images. After each octave, the Gaussian images is down-samples by a factor of 2, and the process repeated [47].

Figure 4.5: Extrema points are detected by comparing a pixel (marked with X) to its 26 neighbours (marked with circles) in $3 \times 3$ regions at the current and adjacent DoGs. [47]
good discrimination between different classes, we apply SIFT to each class (Figure 4.8) of postures and list the number of matching points in Table 4.3. The numbers in red on the diagonal are the number of the matching points between the same class. As you can see, it is bigger than other numbers in its same row or column, which presents the number of matching points between two different classes. It shows that although there are matching features between different posture classes, the number of the same class is bigger than others.
Table 4.3: The number of matching points between different posture classes

<table>
<thead>
<tr>
<th>Posture</th>
<th>Palm</th>
<th>V</th>
<th>Point</th>
<th>Six</th>
<th>Fist</th>
<th>Eight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palm</td>
<td>45</td>
<td>12</td>
<td>6</td>
<td>16</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>V</td>
<td>7</td>
<td>36</td>
<td>13</td>
<td>8</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>Point</td>
<td>7</td>
<td>22</td>
<td>39</td>
<td>9</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>Six</td>
<td>12</td>
<td>14</td>
<td>7</td>
<td>25</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Fist</td>
<td>4</td>
<td>8</td>
<td>15</td>
<td>6</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>Eight</td>
<td>16</td>
<td>15</td>
<td>8</td>
<td>15</td>
<td>9</td>
<td>27</td>
</tr>
</tbody>
</table>

4.2.2 Bag of Visual Words

Bag of visual words is a popular algorithm for image classification. Each image is represented by a set detected feature points, and bag of visual words is using a vector to represent the occurrence counts of feature points. In other words, it is a histogram over the feature points. In this way, each image is represented by a histogram vector. Figure 4.9 shows a typical processing pipeline for generating a feature point histogram for each image.

![Processing pipeline for generating histogram of visual word for each image](image)

Figure 4.9: The pipeline of generating histogram of visual word for each image based on the detected feature points.

In this work, the feature points are detected by SIFT and the clustering is done by K-means (Appendix C). K-means clustering encode each feature point by the index of which cluster it belongs. Usually, this is done by finding the shortest Euclidean distance between the input feature point and the cluster centers which are trained by a group of feature points
extracted from a set of training images. How to determine the number of clusters is a problem. If the number of clusters is too small, there have not enough discrimination between classes, and the classification accuracy will be decreasing. If the number of clusters is too big, then the features would be over-scattered, and the classification accuracy is still going to be decreasing. In \[49\] a rule is proposed to determine the number of clustering \(k\).

\[ k \approx \sqrt{\frac{n}{2}}, \]  

(4.6)

where \(n\) is the number of detected SIFT feature points on the training images. There are over 20,000 feature points detected on the our training images. Therefore, \(k\) is set at 150. After clustering, each template is represented by a vector of indices that shows which cluster the corresponding feature is belonging to. The final step would be generating this vector into a histogram which have \(k\) bins. In this way, each template would be mapped into a \(1 \times 150\) vector which is called bag-of-words vector.

### 4.2.3 Training Stage for Posture Classifier Using SVM

After mapping the feature points of each template into one bag-of-words vector, those vectors are employed to train a multi-class SVM classifier model. For posture recognition, six classes of postures are collected, both front view and side view for each class are included. Figure 4.10 shows the six classes of postures, the front view is shown in Figure 4.10a while the side view is shown in Figure 4.10b, from left to right, the postures are "Palm", "V", "Point", "Six", "Fist" and "Eight".

For training a posture classifier, well-segmented hand posture templates are collected from six posture classes clipped from gesture videos, both front views and side views are included. By applying the bag of words, each image is represented by a \(1 \times 150\) vector. The multi-class SVM is trained base on such vectors. Table 4.4 shows the 5-fold cross-validation accuracy of different numbers of training data with a linear kernel SVM. By increasing the number of training data, the accuracy would have slightly increased. Here we use 402 posture images as the training data set to build and SVM classifier.

### 4.2.4 Testing Stage for Posture Recognition

The testing set contains 432 postures make by the other four people at 216 time state, 36 time state for each class. At each time state, one front view and side view of the posture
are obtained. To test the performance of the trained posture classifier, the image of each posture taken from both camera views are recognized individually. The recognition result is shown in Table 4.5. Each column refers to a posture instance that is classified into the corresponding class. An accuracy of 78.7% is obtained.

The classifier distinguishes most of the testing postures correctly. But for the postures belongs to class "Fist", a fair amount of postures is classified into class "Point", also a few number of postures "Six" is misclassified into class "Fist". The reason for this is these three postures all contain a similar hand part of three finger curling together. Therefore, they share more similar features. But for these postures which are well distinguished with each other, their recognition accuracies are higher.

Also, if we look into the misclassified postures, not well-segmented hand postures can cause the misclassification as well. Figure 4.11 shows three samples of misclassified postures. Background noises which are segmented as hand area (Figure 4.11a and 4.11b) is
Table 4.5: Result matrix for posture recognition on the testing set

<table>
<thead>
<tr>
<th>Testing Postures</th>
<th>Eight</th>
<th>Fist</th>
<th>Palm</th>
<th>Point</th>
<th>Six</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eight</td>
<td>71</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Fist</td>
<td>0</td>
<td>46</td>
<td>0</td>
<td>15</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Palm</td>
<td>0</td>
<td>0</td>
<td>61</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Point</td>
<td>0</td>
<td>17</td>
<td>3</td>
<td>52</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Six</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>43</td>
<td>1</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>11</td>
<td>67</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>98.6%</td>
<td>63.9%</td>
<td>84.7%</td>
<td>72.2%</td>
<td>59.7%</td>
<td>93.06%</td>
</tr>
</tbody>
</table>

(a) (b) (c)

Figure 4.11: Examples of misclassified postures.

A cause of misclassification. Also due to hand appearance changing with viewpoint, some posture cannot be recognized when the appearance changes too much. For instance, in Figure 4.11c, the index finger of posture "Eight" is total invisible from the camera view, and it was misclassified into class "Fist".

Considering the fact that the postures taken at the same time state is the same posture but different view, the posture recognition results from both camera view are associated with each other in the following way. At the same time state, if the recognition results of the postures taken from two cameras are different, then this posture is taken as ambiguous and discarded. The postures that are recognized as the same class from both camera views will be kept for gesture recognition in later steps. For each posture class, the recognition and abandon rate are shown in Table 4.6. Although nearly thirty percent of the testing data is abandoned, this scheme increases the recognition accuracy from 78.7% to 94.7%.

55
### 4.3 Gesture Recognition

The results so far only have focused separately on the trajectory and posture recognitions. In this thesis, we focus also on hand gestures which are defined by hand moving trajectory and the hand postures made along the trajectory. Therefore, we try to combine the results of posture and trajectory recognition into a feature vector we call gesture vector. A gesture vector is extracted from each gesture video and used for classification by SVM. The details of how to define a feature vector are introduced in the following. Also, the performance of gesture recognition is evaluated this section.

#### 4.3.1 Gesture Vector

A gesture vector is constructed from two components: posture elements $p_{ij}$ and trajectory element $T$. $T$ indicates the recognized trajectory class. Depending on the complexity of gestures, each gesture can be separated into $i$ segments to deal with posture variations. $p_{ij}$ represents the occurrence number of the recognized posture class $j$ in segment $i$. Equation 4.7 shows a gesture vector with $i$ segments and $j$ posture class.

$$v_g = [p_{11}, p_{12}, p_{13}, ..., p_{ij}, ..., p_{mn}, T] \quad (4.7)$$

The gesture made in different speed would generate a big number difference on recognized postures. Because the frame rate is fixed for cameras, the faster the hand is moving, the fewer postures can be taken. Therefore, the posture elements in gesture vector are normalized by the total number of recognized posture for each section $P_m$. Equation 4.7 is

<table>
<thead>
<tr>
<th>Postures</th>
<th>Recognition accuracy</th>
<th>Abandon number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eight</td>
<td>35/35</td>
<td>1</td>
</tr>
<tr>
<td>Fist</td>
<td>16/23</td>
<td>13</td>
</tr>
<tr>
<td>Palm</td>
<td>27/27</td>
<td>9</td>
</tr>
<tr>
<td>Point</td>
<td>17/18</td>
<td>18</td>
</tr>
<tr>
<td>Six</td>
<td>17/17</td>
<td>19</td>
</tr>
<tr>
<td>V</td>
<td>31/31</td>
<td>5</td>
</tr>
<tr>
<td><strong>Recognition Accuracy</strong></td>
<td><strong>Abandoned Rate</strong></td>
<td></td>
</tr>
<tr>
<td>94.7%</td>
<td>30.1%</td>
<td></td>
</tr>
</tbody>
</table>
rewritten as:

\[ \begin{bmatrix} p_{11} & p_{12} & p_{13} & \ldots & p_{ij} & \ldots & p_{mn} & T \end{bmatrix} \]  

(4.8)

The gesture vector is extracted for each hand gesture and adopted for gesture classification.

### 4.3.2 Defining Gesture Classes and Gesture Vectors

To test the recognition performance, we defined eight classes of hand gesture. Both one-hand and two-hand gestures are included. Although we only defined eight classes of gesture for evaluation, more gesture classes can always be added to the system with some training data and gesture definitions. Figure 4.12 shows the combination of hand moving trajectory and posture. The blue arrows represent the hand moving trajectory, and the posture changing is also shown in this figure.

A list of the gesture name and its component of posture and trajectory is listed in Table 4.7. Since the defined gestures only contain one or no posture change, each trajectory is segmented into two sections which make the gesture vector \(2 \times 6 + 1 = 13\) elements long for one-hand gesture. In two-hand case, the gesture vector for each hand is extracted for each hand and then combined together which make the two-hand gesture vector is twice long than one-hand case, and contains \(2 \times 6 + 1 + 2 \times 6 + 1 = 26\) elements.

**Table 4.7: The names of gesture classes and their compositions**

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Posture</th>
<th>Trajectory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grab</td>
<td>Palm → Fist</td>
<td>+ Forward</td>
</tr>
<tr>
<td>Hit</td>
<td>Fist</td>
<td>+ Forward</td>
</tr>
<tr>
<td>Call</td>
<td>Six</td>
<td>+ Circle</td>
</tr>
<tr>
<td>Poke</td>
<td>Point</td>
<td>+ Forward</td>
</tr>
<tr>
<td>Zoom</td>
<td>Eight</td>
<td>+ Left</td>
</tr>
<tr>
<td>Move</td>
<td>V → Eight</td>
<td>+ Right</td>
</tr>
<tr>
<td>Push</td>
<td>Right and Left Palms</td>
<td>+ Forward</td>
</tr>
<tr>
<td>Collision</td>
<td>Right and Left fists</td>
<td>+ Left and Right</td>
</tr>
</tbody>
</table>

### 4.3.3 Training Stage for Gesture Classifier Using SVM

In the training stage, 10 gestures for each class, 80 gestures in total, are collected for the training stage among four people. The one-hand and two-hand posture models
Figure 4.12: The example of eight gesture classes. The beginning and ending frames are shown, and the moving directions are shown in blue arrows.
are trained separately by SVM. The ground truth of trajectory and the recognition result of hand postures are utilized to compose the gesture vector. A linear kernel SVM is trained which obtained a 5-fold cross-validation accuracy at 94% for one-hand gestures and 100% percent for two-hand gestures. The training accuracy for hand gestures is listed in Table 4.8.

Table 4.8: The cross-validation accuracy on training gestures

<table>
<thead>
<tr>
<th>Gesture</th>
<th>one-hand gesture</th>
<th>two-hand gesture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>94%</td>
<td>100%</td>
</tr>
</tbody>
</table>

4.3.4 Testing Stage for Gesture Recognition

For this section, another 80 gestre are collected for testing. These 80 gestures, 10 for each gesture class, are collected among four people. Based on how many hand posture detected in the initial areas, the gesture would be classified as one-hand or two-hand gesture automatically. Then, based on the trajectory and posture recognition for each hand, the trained SVM model would recognize each hand gesture based on its gesture vector. Figure 4.13 shows the testing pipeline. Table 4.9 shows the recognition results and accuracies for each class.

Figure 4.13: The testing pipeline of gesture recognition.

All the postures are well recognized except for class "Grab" and "Hit". This is because, at posture recognition stage, the trained classifier could misclassify "Fist" into "Point" (Table 4.5) due to the similarity of these two postures. Therefore, there is a high chance of misclassification for this class. For the gestures that involve posture changing because there would be a period of interval postures appear which are not defined in the posture
Table 4.9: Gesture recognition results of the testing set

<table>
<thead>
<tr>
<th>Testying Gestures</th>
<th>Grab</th>
<th>Hit</th>
<th>Call</th>
<th>Poke</th>
<th>Zoom</th>
<th>Move</th>
<th>Push</th>
<th>Collision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grab</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hit</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call</td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Poke</td>
<td>2</td>
<td>2</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zoom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Push</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Collision</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Accuracy</td>
<td>60%</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

92.5% (74/80)

classes. With our posture recognition scheme, such posture would have a high chance to be discarded while the posture recognition results are not consistent. For instance in the class "Move" the interval posture for changing posture "V" to "Eight" are discarded and acquire a high recognition accuracy.

### 4.4 Discussions

In this chapter, we utilize DMP on trajectory recognition. The recognition is done by SVM based on the weight vector that extracted by DMP from each gesture. Posture recognition is also presented in this chapter. The recognition result that only rely on one camera view is 78.7%. By adding a second camera view, although some posture frames are discarded, the accuracy will increase to 94.7%. An innovative gesture recognition method is proposed. By combining the posture and trajectory information, a gesture vector could be extracted from each gesture video and classified by SVM. For the current stage, a gesture vector is consist by the recognition results of hand moving trajectory and postures. In the future, the features extracted from trajectories and postures can be used to consist the gesture vector for recognition.
Chapter 5

Conclusions and Future Work

This thesis has introduced a framework for vision-based hand gesture recondition with multiple cameras. The results we have achieved so far can be a foundation for an advanced and more comprehensive multi-camera hand gesture recognition system. In this chapter, we conclude this thesis by each module that constitute the system. Also, the suggestions for future work is provided for a more robust and efficient gesture reigation system.

5.1 Conclusions

For vision-based gesture recognition system, segmenting hand area from clutter background is always a challenging problem. This system does not require the user to wear any additional device or making gestures in front of a clean or steady background. The fixed-range CbCr is adopted for hand area segmentation. Other background noises such as other skin-color-liked area and face and neck area are also included in the segmentation results. By adding size and position constraint on the segmented blobs, the hand area can be located.

A Kalman Filter is adopted for hand tracking and occlusion. With the initialization step hand a constraint on hand area increase rate, the background noise can be further excluded. KF can track hands successfully with a short time occlusion or overlap with background noise.

A multi-calibrated-camera system captures the hand gesture from two viewpoints. Based on the camera geometry, the hand location in the world space can be recovered from the
pixel values on the two image planes that are obtained by KF tracking. The 3D hand moving trajectory can be reconstructed based on the tracking results. When hand area is overlapped with background noises and gives a pixel value that will reconstruct hand location off the trajectory. Such error points can be eliminated by trajectory smoothing which is done by RLOESS in our work.

Hand trajectory involves both temporal and spatial patterns that vary among different individuals. DMP as a method that could preserve the spatio-temporal information in a weight vectors. Topological similar trajectories would have similar weight vectors. Therefore, such vectors can be used as feature vectors for trajectory classification which is done by SVM. With only a few trajectory training data, the recognition can be achieved with an accuracy of 88.0%.

The feature points on each hand posture clipped from gesture video are detected by SIFT. Then a bag-of-words approach is employed to represent each posture into an uni-sized histogram vector. Such histogram vector is used for posture recognition by SVM. Unlike other vision-based posture recognition that capture hand image just from one point of view, we adopt two cameras to capture hand image from different view points. The training postures contain both front and side views taken by the cameras. The hand posture only is considered as being recognized when the recognition result from both camera views are consistent. With such scheme, although some of the postures are taken as not-recognized and discarded (30%), the recognition rate can reach 94.7%.

Gesture recognition combines the results of both trajectory and posture recognition, and integrates them into a gesture vector. Gesture vector counts the portion of recognized posture for each posture class along the gesture and includes the recognized trajectory class. In this way, each gesture is represented by a gesture vector and treated as a whole. Gesture recognition is done by gesture vectors with SVM and earned an accuracy of 92.5%.

5.2 Future Work

This proposed gesture recognition system is a comprehensive system including hand segmentation, hand tracking, 3D trajectory reconstruction, trajectory recognition and posture recognition. However, It is still at the preliminary stage and needs to be improved and extended in the future. Possible future work can be defined as follows:
• For now, due to the processing time of posture feature detected by SIFT, the system cannot achieve gesture recognition result in real-time. A faster hand area segmentation and posture feature detection methods are needed to be explored.

• To eliminate background noise, obtain well-segmented hand area, and also more robust to illumination changes, a more reliable hand segmentation area is needed, such as, an adaptive skin color model for skin color segmentation and hand detection based on context.

• For the same posture, the front view and side view recognition accuracy are different. To reduce the variation in recognizing different posture classes, in the future, the correlation between posture recognitions can be explored by attributing different weight on different posture view posture to improve recognition accuracy.

• The hand gesture in this thesis is on word-level and do not need to segment target trajectory from a serial of movement. For further work, such step is essential to sign language recognition.

• Here only two simple gestures involving two hand are explored. The trajectory of each hand are done separately. For later work, the movement relationship between two hands is another key information that need to be considered.

• With the technology develops, depth camera such as Kinect can be obtained with a much lower price and smaller volume. The 3D information of hand shape and trajectory can be achieved easily and accurately which will make the recognition result more reliable.
Bibliography


66


[50] Albert Mehrabian. Silent messages. 1971. 1


68


69
[87] Jiawei Zhou. Visual tracking and region alignment in surgical and medical education, 2014. 25

Appendix A

Conversions of Color Space

Color space is an abstract mathematical model to describe color, typically as three values of color components. RGB, HSV and YCbCr are the color space we have used. Color space conversion is the translation of a color from one color space to another color space. This usually occurs in the context of converting a color space of an image.

The default color format in OpenCV is referred to as RGB. The conventional ranges for R, G, and B channel values are $0 \sim 255$ for 8-bit integer images. The conversions from RGB color space to other formats ranging of HSV and YCbCr are presented below.

A.1 RGB to HSV

HSV color space provides an intuitive method for color selection. The hue parameter $H$ defines the color information. The saturation parameter $S$ indicates the range of grey in the color space. The value parameter $V$ is the brightness of the color. The transformation to convert from RGB to HSV is

$$V = \max(R, G, B)$$

$$s = \begin{cases} \frac{V - \min(R,G,B)}{V} & \text{if } V = G \\ 0 & \text{otherwise} \end{cases}$$

$$H = \begin{cases} 60(G - B)/(V - \min(R,G,B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - \min(R,G,B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - \min(R,G,B)) & \text{if } V = B \end{cases}$$

If $H < 0$ then $H = H + 360$. On output, we have

$$\begin{cases} 0 \leq V \leq 1 \\ 0 \leq S \leq 1 \\ 0 \leq H \leq 360 \end{cases}$$
The values are the converted to the destination data type. For 8-bit images, the values are converted as

\[
\begin{align*}
V &= 225V \\
S &= 225S \\
H &= H/2
\end{align*}
\]

### A.2 RGB to YCbCr

For a YCbCr image, the luminance information is represented by a single component, Y, and color information is stored as two color-difference components, Cb and Cr. Component Cb is the difference between the blue component and a reference value, and component Cr is the difference between the red component and a reference value. The transformation to convert from RGB to YCbCr for 8-bit images is

\[
H = \begin{cases}
Y = 0.299R + 0.587G + 0.114B \\
Cb = 128 - 0.169R - 0.31G + 0.5B \\
Cr = 128 + 0.5R - 0.419G - 0.081B
\end{cases}
\]
Appendix B

Multi-Camera Geometry

B.1 Pinhole Camera Model

Camera maps the 3D world into a 2D image. A simple but useful model of how this happens is the pinhole camera model [10]. In an idealized pinhole camera, a point in 3D space is projected onto an image surface. It is shown in Figure B.1, where \( f \) is the focal length of the camera, and \( Z \) is the distance from the camera to the point.

![Figure B.1: A idealized pinhole camera model. The light rays from point Q in the 3D space images q onto the image plane through a pinhole.](image)

To make the math coming out easier, the model can be rearranged into a new form that showed in Figure B.2. In the figure, we can see by similar triangles that

\[
\frac{u}{f} = \frac{X}{Z}, \frac{v}{f} = \frac{Y}{Z}
\]  

(B.1)

Three coordinate references systems are defined in this model for later convenience: the world coordinate system, the image plane and the camera coordinate system. The world coordinate system \( O - X - Y - Z \) is a standard 3D Cartesian coordinate frame that can be chosen arbitrarily. Simplifying the math for later steps is the principle of how to chose the world coordinate system. The image plane \( o - u - v \) is a coordinate system to describe the location of projected points on the image plane. It is always parallel to the
Figure B.2: The image plane is rearranged in front of the pinhole. A point \( Q = (X, Y, Z) \) is projected onto the image plane and the resulting point is \( q = (u, v) \).

The camera coordinate system \( C_c - X_c - Y_c - Z_c \) plane of the camera coordinate system \( C_c - X_c - Y_c - Z_c \) which always centers at the optical center \( C_c \) (the pinhole). Besides, the center \( c \) of the image plane is on the axis of \( C_c - Z_c \) which is known as the optical axis and the distance between the \( c \) and \( C_c \) is always \( f \).

### B.2 Single-Camera Calibration

Based on the pinhole camera model, equation B.1 represents the ideal case. But in practical, the center of the image plane and the pinhole are not perfectly aligned. Two new parameters \( c_u \) and \( c_v \) are introduced to model this possible displacement. The relationship between the \( Q \) and \( q \) now become:

\[
u = f_u \frac{X}{Z} + c_u, \quad v = f_v \frac{Y}{Z} + c_v
\]

(B.2)

Note that the unit of focal lengths \( f_u \) and \( f_v \) and displacement parameters \( c_u \) and \( c_v \) in equation B.2 is pixel. Therefore the value of \( u \) and \( v \) is also in pixel unit which is very convenient for following experiment and result analysing.

The projection of the points in the physical world into the image plane can be represented by the following equation:

\[
kq = PQ, \text{ where } q = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, Q = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad P = K[R|t], K = \begin{bmatrix} f_u & s & u_o \\ 0 & f_v & v_o \\ 0 & 0 & 1 \end{bmatrix}.
\]

(B.3)

Point \( Q \) in the world coordinate and its projected point \( q \) on the image plane are represented by homogeneous coordinates. \( P \) is a \( 3 \times 4 \) perspective projection matrix which
composed by intrinsic matrix $K$ and extrinsic matrix $[R|t]$. $k$ is an arbitrary scale factor defined up to $P$ and make the equation B.3 establishing.

In intrinsic matrix $K$, $f_u$ and $f_v$ are the camera focal length expressed in pixel units along $u$ and $v$ directions, respectively. $s$ denotes a skew coefficient between $u$ and $v$ axes on the image plane. Usually $s$ equals to zero if the camera pixel grid is well aligned. $[u_o, v_o]$ is the coordinate of center $c$ on the image plane.

In extrinsic matrix $[R|t]$, $R$ is the rotation matrix, and $t$ is the translation vector to relate the world coordinate system to the camera coordinate system.

The purpose of single-camera calibration is to estimate the parameters in $K$ that define the camera (i.e., $f_u, f_v, u_o$ and $v_o$) and the extrinsic parameters in $[R|t]$.

### B.3 Multi-Camera Calibration

For a multi-camera system, besides determining the intrinsic and extrinsic parameters for each camera, camera calibration also need to estimate the geometrical relationship between cameras. So the cameras could collaborate with each other with location information.

A typical dual-camera model with overlapping FOV is demonstrated in Figure B.3

![Figure B.3: A typical dual-camera model with overlapping FOV.](image)

$q_1, q_2$ are two projected points of the same point $Q$ onto two image planes respectively. $T_2^1$ is a $4 \times 4$ matrix that describes the relative relationship in position and orientation between two camera system, $C_1$ and $C_2$. 

75
B.4 3D Coordinate Reconstruction

The objective of 3D coordinate reconstruction is to recover the 3D coordinate information of a point in the physical world based on one or more of its images. Recall the pinhole camera model is referred to as a many-to-one mapping, which means that all points lines on the same projecting ray through a 3D scene point and the optical center are projected onto the same pixel of the image plane. In other words, the depth information \( Z \) a 3D point \( Q \) in the physical world is lost when mapping it onto an image plane. Figure B.4 represents the schematic in an intuitive way. The point \( q \) on the image plane can be the projection of any point \( Q \) lies on the line through the center of projection. Therefore, recover depth information of a point from one image is an underdetermined problem.

![Figure B.4: As a projected point on the image plane, \( q \) lost the depth information of point \( Q \) in the physical world. The point \( q \) can be the projection of any world point \( Q \) along the projecting ray \( CQ \).](image)

To recover the depth information of \( Q \), only one projected point \( q \) leaves the problem under-determined. Another restriction such as projected point on another image plane is necessary, as depicted in Figure B.5. \( Q \) is now restricted to the intersection point of a project ray \( C_1Q \) and \( C_2Q \).

Based on Figure B.5 and equation B.3, the relationship between a point \( Q = [X, Y, Z, 1]^T \) in the world coordinate system and its image \( q_1 = [u_1, v_1, 1]^T \) on the image plane of a camera \( C_1 \) can be established as equation B.4. Similarly, the relationship between \( Q \) and its image \( q_2 = [u_2, v_2, 1] \) on the image plane of a camera \( C_2 \) is given in equation B.5.
Figure B.5: A point in world space $Q$ is restricted based on the projected points on two images planes.

\[
k_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = f_{u1} \begin{bmatrix} f_{u1} & s_1 & u_{o1} \\ 0 & f_{v1} & v_{o1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{11} \\ r_{12} & r_{12} & r_{13} & t_{12} \\ r_{13} & r_{13} & r_{13} & t_{13} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},
\]

(B.4)

\[
k_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = f_{u2} \begin{bmatrix} f_{u2} & s_2 & u_{o2} \\ 0 & f_{v2} & v_{o2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{21} & r_{21} & r_{21} & t_{21} \\ r_{22} & r_{22} & r_{22} & t_{22} \\ r_{23} & r_{23} & r_{23} & t_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},
\]

(B.5)

Where $r_{ij}$ and $t_j$ are the elements of rotation matrices and translation vectors. $Q$ is assumed to be in the overlapping area. Observing equation B.4 and B.5, there are 4 equations with only 3 unknowns which is the coordinate of $Q$: $[X, Y, Z]^T$. The key points of recover the 3D coordinate of $Q$ is to solve the depth information $Z$ that is lost during the projecting process, which is included in the arbitrary scale factors $k$. Equation B.4 and B.5 can be rewritten as:

\[
k_1 q_1 = K_1 [R_1 | t_1] Q.
\]

(B.6)

\[
k_2 q_2 = K_2 [R_2 | t_2] Q.
\]

(B.7)

\[
k_1 q_1 = K_1 R_1 Q + K_1 t_1.
\]

(B.8)
\[ k_2q_2 = K_2R_2Q + K_1t_2. \]  \hspace{1cm} (B.9)

\[ k_1q_1 - K_1t_1 = K_1R_1Q. \]  \hspace{1cm} (B.10)

\[ k_2q_2 - K_2t_2 = K_2R_2Q. \]  \hspace{1cm} (B.11)

\[ R_1^{-1}K_1^{-1}(k_1q_1 - K_1t_1) = Q = R_2^{-1}K_2^{-1}(k_2q_2 - K_2t_2) \]  \hspace{1cm} (B.12)

\[ R_1^{-1}K_1^{-1}k_1q_1 - R_1^{-1}t_1 = R_2^{-1}K_2^{-1}k_2q_2 - R_2^{-1}t_2 \]  \hspace{1cm} (B.13)

Equation B.14 can be rewritten into a matrix form as:

\[
\begin{bmatrix}
    r_{211} & r_{212} & r_{213} \\
    r_{221} & r_{222} & r_{223} \\
    r_{231} & r_{232} & r_{233}
\end{bmatrix}
\begin{bmatrix}
    f_u & s_1 & u_0 \\
    f_v & v_0 & 1
\end{bmatrix}
^{-1}
\begin{bmatrix}
    u_1 \\
    v_1
\end{bmatrix}
= 
\begin{bmatrix}
    r_{111} & r_{112} & r_{113} \\
    r_{121} & r_{122} & r_{123} \\
    r_{131} & r_{132} & r_{133}
\end{bmatrix}
\begin{bmatrix}
    t_11 \\
    t_12 \\
    t_13
\end{bmatrix}
\]  \hspace{1cm} (B.14)

Equation B.14 can be rewritten into a matrix form as:

\[
\begin{bmatrix}
    m_1 \\
    m_2 \\
    m_3
\end{bmatrix}
\begin{bmatrix}
    k_1 \\
    k_2
\end{bmatrix}
= 
\begin{bmatrix}
    n_1 \\
    n_2 \\
    n_3
\end{bmatrix}
\]  \hspace{1cm} (B.15)

The method of ordinary least square can be used to find an approximate solution to such equation system. For the system \( Ax = b \), the solution can be written as the normal equations,

\[ x = (A^TA)^{-1}A^Tb. \]  \hspace{1cm} (B.16)

Substituting \( A \) with \( [m_1, m_2m_3]^T \), \( b \) with \( [n_1, n_2, n_3] \) and \( x \) with \( [k_1, k_2] \), we can determine the values of \( k_1 \) and \( k_2 \).
Appendix C

Appendices: K-means Clustering

For a data set \( x_1, x_2, \ldots, x_N \) consisting of \( N \) data points. Each data is a point in a \( D \)-dimensional Euclidean space. To separate these data into \( K \) clusters where \( K \) is given, an objective function is given by

\[
J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2
\] (C.1)

which represent the sum if the square of the distance of each data point to its cluster center \( \mu_k \), and \( r_{nk} \in \{0, 1\} \) is a binary indicator variables transcribing which of the \( K \) cluster the data point \( x_n \) is assigned to. If data point \( x_n \) is assigned to cluster \( k \) the \( r_{nk} = 1 \) and \( r_{nj} = 0 \) for \( j \neq k \). The goal for k-means cluster is to find the values for \( r_{nk} \) and \( \mu_k \) so as to minimize \( J \). An iteration procedure which involves two successive steps corresponding to successive optimization with respect to \( r_{nk} \) and \( \mu_k \).

First some initial values for the \( \mu_k \) are randomly selected. Then in the first phase, \( \mu_k \) is fixed, and \( J \) is minimized respect to \( r_{nk} \). In the second phase, \( r_{nk} \) is fixed, and \( J \) us minimized respect to \( \mu_k \). This two-stage-optimization is repeated until convergence. Figure C.1 illustrates K-means algorithm in a two-dimensional Euclidean space.
Figure C.1: Illustration of the K-means algorithm. (a) Blue points denote the data set in a two-dimensional Euclidean space. The initial choices for centres $\mu_1$ and $\mu_2$ are shown by the red and green crosses, respectively. (b) In the initial step, each data point is assigned either to the red cluster or to the green cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d) and (e) show successive these two steps through to final convergence of the algorithm [5].
Appendix D

Appendices: Support Vector Machine

A support vector machine (SVM) is a classifier formally defined by a separating hyperplane. Based on given labeled training data, the algorithm outputs an optimal decision boundary that categorized new examples. To find the optimal decision boundary, we introduce a concept called margin which is defined to be the smallest distance between the division boundary and any of the samples. In SVM, the decision boundary is chosen to be the one for which the margin is maximized. All lines in Figure D.1 are decision boundary, but only the line in red maximized the margin and taken as the optimal decision boundary. The location of this boundary is determined by a subset of the data points known as support vectors that are indicated by the circles.

![Figure D.1: Illustration of margin, the maximum margin and support vectors. The margin is defined as the perpendicular distance between the decision boundary and the closest of the data point. Maximizing the margin leads to a particular choice of a decision boundary which called optimal decision boundary. The location of this boundary is determined by support vectors that are circled in brown.](image)

The decision boundary is noted as:

\[ y(x) = w^T x + b \]  

(D.1)

where \( w \) is known as the weight vector and \( b \) is bias.
The optimal decision boundary can be represented in an infinite number of different ways by scaling of \( w \) and \( b \). As a matter of convention, the decision boundary is chosen as:

\[
|w^T x + b| = 1
\]  

(D.2)

where \( x \) is the support vector that is the training examples closest to the boundary. The distance \( D \) between a point \( x \) and a boundary is given by

\[
D = \frac{|w^T x + b|}{||w||}.
\]  

(D.3)

For the support vectors, the distance \( D_{SV} \) can be rewritten as

\[
D = \frac{|w^T x + b|}{||w||} = \frac{1}{||w||}.
\]  

(D.4)

The maximized margin \( M \) is twice as the distance to the closest data points.

\[
M = 2D_{SV} = \frac{2}{||w||}
\]  

(D.5)

The problem of maximizing \( M \) is equivalent to the problem of minimizing a function \( L(w) \) subject to some constraints (equation ).

\[
\min_{w,b} L(w) = \frac{1}{M^2} = \frac{||w||^2}{2}
\]  

subject to

\[
a_i(w^T x + b) \geq 1, \forall i
\]  

(D.7)

where \( a_i \) is the labels of each training examples. This is a problem of Lagrangian optimization that can be solved using Lagrange multipliers to obtain the weight vector \( w \) and the bias \( b \) of the optimal decision boundary. For more details of SVM can be found in [5].
Appendix E

Appendices: Locally Weighted Scatterplot Smoothing (LOESS)

Assume there are \( N \) data points, the observation \( y_i \) and the corresponding measurement \( x_i \) are related by

\[
y_i = g(x_i) + \epsilon_i \tag{E.1}
\]

where \( g \) is the regression function and \( \epsilon_i \) is a random error. LOESS is to fit linear or quadratic regression function to the data points within a chosen neighbourhood of the point \( x_i \). The radius of each neighbourhood is chosen so that the neighbourhood contains a specified percentage of the data points. Bigger weight gives to the neighbourhood that are close to \( x_i \) and smaller weight to those points that are further.

The neighbourhood for any \( x \) is determined by the nearest neighbours. \( x_r \) the \( r \)th closet \( x_i \) to \( x \) where \( r < d \). The neighbourhood is a d-dimensional sphere about \( x \) whose radius is the distance from \( x \) to \( x_r \). Let \( w = (x_r - x)/d \) the fraction of points in the neighbourhood. Then the local regression model specifies \( g \) through the specification of \( w \) and fitted locally.
Appendix F

Appendices: Dynamic Movement Primitives

This appendix shows an example of generating weight vector by DMP. Figure F.1 shows a hand moving trajectory. Figure F.2a shows the hand location in $x$ axis along time. The velocity and acceleration of the hand are shown in Figures F.2b and F.2c. The blue line in Figure F.2 shows the original hand location, and the green line is the fitting trajectory by DMP. The difference of between those two lines is small which means DMP reserves the hand moving information.

![Figure F.1: Hand moving trajectory](image)

This example fits the trajectory showed in F.2a with a weight vector with $D_w = 10$. Figure F.3a shows the Ten Guassian-liked basis functions used to fit $f$ in Equation 4.3. The weight of each function is shown in F.3b.
Figure F.2: Hand position, velocity and acceleration in $x$ direction. The horizontal axis indicate the duration time of hand gesture and the vertical axis shows the hand location along $x$ direction. (a) Hand moving trajectory in $x$ direction. (b) Hand velocity in $x$ direction. (c) Hand acceleration in $x$ direction. The blue line shows the original hand location, and the green line is the reconstructed trajectory by DMP.

Figure F.3: (a) Ten Gaussian-like basis functions. (b) The weight of each function.
Appendix G

Appendices: Hidden Markov Models

The HMM is a collection of states connected by transitions. Each transition has a pair of probabilities: a transition probability (which provides the probability for taking the transition) and an output probability (which defines the conditional probability of emitting an output symbol from a finite alphabet given a state). A formal characterization of HMM is as follows:

- \( \{s_1, s_2, ..., s_N\} \)-- A set of \( N \) states. The state at time \( t \) is denoted by the random variable \( q_t \).
- \( \{v_1, v_2, ..., v_M\} \)-- A set of \( M \) distinct observation symbols. The observation at time \( t \) is denoted by the random variable \( Q_t \). The observation symbols correspond to the physical output of the system being modeled.
- \( A = \{a_{ij}\} \)-- An \( N \times N \) matrix for the state transition probability distribution where \( a_{ij} \) is the probability of making a transition from state \( s_i \) to \( s_j \):
  \[
a_{ij} = P(q_{t+1} = s_j | q_t = s_i). \tag{G.1}
\]
- \( B = \{b_j(K)\} \)-- An \( N \times M \) matrix for the observation symbol probability distributions where \( b_j(k) \) is the probability of emitting \( v_k \) at time \( t \) in state \( s_j \):
  \[
b_j(K) = P(O_t = v_k | q_t = s_j). \tag{G.2}
\]
- \( \pi = \{\pi_i\} \)-- The initial state distribution where \( \pi_i \) is the probability that the state \( s_i \) is the initial state:
  \[
  \pi_i = P(q_1 = s_i). \tag{G.3}
\]

Since \( A, B \) and \( \pi \) are probability, they must satisfy the following constraints:

\[
\sum_j a_{ij} = 1, \forall i, a_{ij} \geq 0. \tag{G.4}
\]
\[ \sum_{k} b_j(K) = 1, \forall i, b_j(k) \geq 0. \]  \hspace{1cm} (G.5)

\[ \sum_{i} \pi_i = 1, \pi_i \geq 0. \] \hspace{1cm} (G.6)

Following the convention, a compact notation \( \lambda = (A, Bm\pi) \) is used which includes only probabilistic parameters. More details can be found in [5].