

Multi-Signal Gesture Recognition Using Body and Hand Poses

by

Yale Song

B.S. Computer Science and Engineering, Hanyang University (2008)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

© Massachusetts Institute of Technology 2010. All rights reserved.

Author.....
Yale Song
Department of Electrical Engineering and Computer Science
September 3, 2010

Certified by.....
Randall Davis
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by.....
Professor Terry P. Orlando
Chair, Department Committee on Graduate Students

Multi-Signal Gesture Recognition Using Body and Hand Poses

by

Yale Song

Submitted to the Department of Electrical Engineering and Computer Science
on September 3, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We present a vision-based multi-signal gesture recognition system that integrates information from body and hand poses. Unlike previous approaches to gesture recognition, which concentrated mainly on making it a single signal, our system allows a richer gesture vocabulary and more natural human-computer interaction. The system consists of three parts: 3D body pose estimation, hand pose classification, and gesture recognition. 3D body pose estimation is performed following a generative model-based approach, using a particle filtering estimation framework. Hand pose classification is performed by extracting Histogram of Oriented Gradients features and using a multi-class Support Vector Machine classifier. Finally, gesture recognition is performed using a novel statistical inference framework that we developed for multi-signal pattern recognition, extending previous work on a discriminative hidden-state graphical model (HCRF) to consider multi-signal input data, which we refer to Multi Information-Channel Hidden Conditional Random Fields (MIC-HCRFs). One advantage of MIC-HCRF is that it allows us to capture complex dependencies of multiple information channels more precisely than conventional approaches to the task. Our system was evaluated on the scenario of an aircraft carrier flight deck environment, where humans interact with unmanned vehicles using existing body and hand gesture vocabulary. When tested on 10 gestures recorded from 20 participants, the average recognition accuracy of our system was 88.41%.

Thesis Supervisor: Randall Davis

Title: Professor of Computer Science and Engineering

Acknowledgments

I would like to express the deepest appreciation to my advisor Professor Randall Davis for his invaluable insights and thoughtful suggestions throughout this work. I would have never been able to finish this thesis without his help. In addition, thanks to David Demirdjian for his profound comments and guidance that helped me to achieve the completion of my thesis. I also thank Professor Mary L. Cummings for providing me with the opportunity to work on this exciting project.

Many thanks to all other members of Multimodal Understanding Group for their friendship and hospitality, as well as being the greatest office mates. Also, I sincerely appreciate all of the KGSA (especially Korgrad07) members for their encouragement and support ever since I came to MIT. I cannot imagine my life here without each one of you, and hope our friendship lasts forever.

Most of all, I would like to thank my parents who are supporting me with their endless and unconditional love. I would also like to thank my brother who has always been an inspirational mentor in my life and good friend of mine.

Contents

- 1 Introduction 23**
 - 1.1 Preface: “The future is gestural.” 23
 - 1.2 Goal and Motivations 24
 - 1.2.1 Goal 24
 - 1.2.2 Motivations 25
 - 1.3 Gesture Recognition Pipeline 26
 - 1.4 Contribution 28
 - 1.5 Thesis Overview 29

- 2 LUVI Project 31**
 - 2.1 Background 31
 - 2.2 NATOPS: Constraints and Challenges 32
 - 2.3 Carrier Flight Deck: Constraints and Challenges 34
 - 2.4 LUVI System Design Criteria 35

3	Image Pre-Processing	37
3.1	3D Vision Cameras	37
3.1.1	Time-of-flight Camera	38
3.1.2	Stereo Vision Camera	39
3.2	Depth Map Calculation	40
3.3	Background Subtraction	42
3.3.1	The Codebook Approach	43
3.3.2	Refinement Using Depth Information	45
4	3D Body Pose Estimation	47
4.1	Generative Model of Human Upper Body	48
4.1.1	Skeletal Model	48
4.1.2	3D Shape Model	57
4.2	Feature Extraction	58
4.2.1	Model Feature Extraction	60
4.2.2	Image Feature Extraction	63
4.3	Estimation Framework	65
4.3.1	Particle Filter	66
4.3.2	Likelihood Function	68
4.3.3	Initialization	71
4.3.4	Estimation Under Constraints	72

4.4	Experiment	74
4.4.1	Qualitative Analysis	75
4.4.2	Quantitative Analysis	80
4.5	Related Work	84
4.5.1	Model-Free Approaches	84
4.5.2	Model-Based Approaches	86
5	Hand Pose Classification	87
5.1	Goal and Assumptions	88
5.2	Hand Pose Vocabulary	89
5.3	Dataset	90
5.3.1	Data Collection	90
5.3.2	Feature Extraction: Histogram of Oriented Gradients	90
5.4	Classification Framework	92
5.4.1	Multi-class Support Vector Machine Classifier	92
5.4.2	SVM Parameter Selection Using Grid Search	94
5.4.3	Search Region	94
5.4.4	Clustering	96
5.4.5	Output Feature Type	97
5.5	Experiment	98
5.5.1	Quantitative Analysis	99

5.5.2	Qualitative Analysis	99
5.6	Related Work	100
6	Gesture Recognition	103
6.1	Generative vs. Discriminative Models	104
6.2	Discriminative Graphical Models	105
6.2.1	Conditional Random Field	105
6.2.2	Hidden Conditional Random Field	106
6.3	Multi Information-Channel Hidden Conditional Random Field	107
6.3.1	Motivation	107
6.3.2	Formulation	108
6.3.3	MIC-HCRF For Gesture Recognition	110
6.4	Experiments	112
6.4.1	Dataset	112
6.4.2	Does Combining Body Pose And Hand Pose Really Help?	114
6.4.3	What Is The Best Feature To Use For Gesture Recognition?	118
6.4.4	How does MIC-HCRF Perform Over HCRF?	128
6.5	Related Work	131
7	Conclusion	133
7.1	Summary of Contribution	134

7.2	Future Work	135
7.2.1	Refining Body Pose Estimation And Hand Pose Classification Using Results From One Another	135
7.2.2	Real-Time Gesture Recognition System	135
7.2.3	Using Context Information	136
7.2.4	Providing Feedback To Human	137
A	NATOPS Gesture Dataset	139

List of Figures

- 1-1 Hitachi gesture remote control television (left) and Microsoft Kinect video game console (right). 24
- 1-2 Gesture recognition pipeline. 27
- 2-1 Some of NATOPS gestures: (a) brake on, (b) brake off, (c) insert chocks, (d) remove chocks 34
- 3-1 Various kinds of time-of-flight cameras. (a) SwissRanger SR-4000, (b) 3DV ZCam, (c) PMD Vision CamCube 3.0 39
- 3-2 Bumblebee2 stereo vision camera from Point Grey Research Inc. 40
- 3-3 A simplified stereo camera setting to illustrate the basic idea of computing a depth map. The depth Z to a point P is calculated using the principle of similar triangles, using a baseline length T , focal length f , and principal points c_x^l and c_x^r 41
- 3-4 A codebook is made up of several intensity bounds that grows to capture a history of intensity values in a particular range. The top image show a history of intensity values over time, and the bottom image shows a set of disjoint intensity bounds.(reproduced from [10]) 44

3-5	An overview of the image pre-processing step. The top-left image is a color image obtained from a stereo camera, and the bottom-left image is a depth map. Two images on the right shows background subtracted images. . . .	45
4-1	Two coordinate frames $o_{i-1}x_{i-1}y_{i-1}z_{i-1}$ and $o_ix_iz_i$, constructed following the Denavit-Hartenberg convention. Note that the two assumptions in the Denavit-Hartenberg convention are illustrated (DH1 and DH2). DH1 shows axis x_i is perpendicular to axis z_{i-1} , and DH2 shows axis x_i intersects axis z_{i-1} . (reproduced from [65]).	50
4-2	A skeletal model of human upper body. There are 8 limbs (neck, trunk, L/R collar bones, L/R upper arms, L/R lower arms) and 9 joints (head, chest, navel, L/R shoulders, L/R elbows, L/R wrists).	52
4-3	A kinematic chain of the human upper body. The coordinate frames are assigned as follows: o_{11} - head, o_0 - chest, o_{12} - navel, $(o_1, o_2, o_3)/(o_6, o_7, o_8)$ - left/right shoulder, o_4/o_9 - left/right elbow, o_5/o_{10} - left/right wrist.	53
4-4	Simplified shoulder model. After joint positions are determined using the forward kinematics, φ is calculated with positions of chest (C), shoulder (S), and elbow (E) joint points, using the law of cosines.	56
4-5	An overlaid image of the right shoulder's movement. Note that the height of the shoulder joint point goes up and down, depending on the right elbow's position.	57
4-6	A skeletal model and shape model of human upper body. Contour points are shown in the white color, while visible surface points are shown in the gray color.	59

4-7	Motion history images (bottom) generated from pairs of surface point clouds (top). In the bottom row gray pixels indicate where the arms moved from, white pixels indicate where the arms moved to.	63
4-8	Input image feature extraction: (a-left top) color image, (b-right top) depth map, (c-left bottom) surface point cloud, and (d-right bottom) contour point cloud.	64
4-9	Motion history images (bottom row) extracted from pairs of input images (top row). At the bottom row, gray pixels indicate where the arms moved from, and white pixels indicate where the arms moved to.	65
4-10	Participants were asked to make a T-shape pose for initialization.	72
4-11	Initialization results collected from 20 subjects. Although each subject has made a T-pose for the initialization, the shoulder angles were not always 90 degree to the backbone, preventing us to assume that the chest point is always at the middle of two wrist points.	73
4-12	A data visualization tool. (1,3): original input image and depth map, (2,4) input image and depth map after background subtraction, (5,9,13): 3D surface point cloud of input image from three different viewpoints (front, top, lateral), (6,10,14): 3D surface point cloud of estimation result from three different viewpoints (front, top, lateral), (7,8): 3D contour point cloud of input image and estimation result, (11,12): motion history image of observation and estimation result, (15): hand pose estimation result (Chapter 5), (16): synthesized view of body and hand pose estimation	76

4-13	Gesture #4 (spread wings) contains a shrugging gesture, which causes a major portion of estimation errors for this gesture. The pose estimator tracks the shrugging gesture correctly for a few frames at the beginning, but fails when arms get too close to the body. Note that it quickly finds the correct pose when there is no more self-occlusion.	78
4-14	Estimation fails when the bending direction of an elbow (red arrows) is incorrect. (Rectangles around hands are hand pose estimation result, which will be discussed in the next chapter).	79
4-15	Pixel distance errors of body pose in gesture #2 (affirmative) and #3 (negative).	82
4-16	Pixel distance errors of body pose in gesture #4 (spread wings) and #5 (fold wings).	82
4-17	Pixel distance errors of body pose in gesture #10 (remove chocks) and #11 (insert chocks).	83
4-18	Pixel distance errors of body pose in gesture #18 (engage nosegear steering) and #19 (hot brakes).	83
4-19	Pixel distance errors of body pose in gesture #20 (brakes on) and #21 (brakes off).	84
5-1	Canonical hand poses used in NATOPS standard aircraft handling signals. The four hand poses used in this work are shown in red boxes.	89
5-2	Hand pose dataset. Positive examples (first four rows) were sampled and labeled manually by cropping 32 x 32 images, and applying affine transformations to scale and rotate them. Negative examples (bottom row) were sampled randomly after positive examples were sampled in each image. . .	91

5-3	A graph showing the result of grid search over the two parameters C and γ . The optimal values of parameters (C, γ) are searched over the grids of (C, γ) values; x-axis shows $\log(C)$ and y-axis shows $\log(\gamma)$. Lines with the same colors indicate the parameter settings that lead to the same accuracy performance. The grid search result shows the optimal parameter values were $C=4.0$ and $\gamma=0.03125$, which gave the overall accuracy 99.86% (the X mark on the graph).	95
5-4	Hand tracking is performed only in two small search regions around estimated wrist positions. Given a 56x56 pixel search region (gray rectangle around each wrist position) and 32 x 32 pixel sliding window moving at 2 pixel interval, we compute the HOG features repeatedly.	96
5-5	Three time-consecutive images (top) and probability estimates (bottom) showing hand pose classification results. The small circles inside the rectangle are detected hand locations, and the rectangle is a clustered result where its location and probability estimates are obtained by averaging neighboring classification results. The colors indicate classified hand pose classes: blue for the "opened palm" and red for the "closed palm." A hand pose with the highest probability estimate is chosen as a categorical result value (shown in bold text in the table.)	97
6-1	Discriminative hidden-state graphical models. (a) HCRF, (b) MIC-HCRF.	109
6-2	A set of 10 gestures used in this work. #2: Affirmative, #3: Negative, #4: Spread wings, #5: Fold wings, #10: Remove chocks, #11: Insert chocks, #18: Engage nosegear steering, #19: Hot brakes, #20: brakes on, #21: brakes off.	113

6-3	Recognition accuracy rates comparing two conditions, a) body pose only (BodyOnly) and b) body and hand pose combined (BodyHand), under the four different body pose features.	116
6-4	Recognition accuracy rates for each gesture class (for body pose features, only dT and dP were used).	117
6-5	Gesture recognition accuracy graph of 6 feature combinations, averaging over all of the 10 gesture classes and 3 numbers of hidden states (3, 4, and 5). Regarding hand pose feature types, S performed significantly better than H in all of our test cases ($t(178)=2.24, p=.018$). Regarding body pose feature types, although there was no significance in the accuracy differences, dP performed the best.	119
6-6	Recognition accuracy rates for each gesture class (for hand pose features, only S was used).	120
6-7	Recognition accuracy rates of the dPS feature combination –a combination of derivatives of joint coordinates (dP) and vectors of probability estimates (S)– for each number of hidden states (3, 4, 5, and 6).	121
6-8	Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #2 (affirmative) and #3 (negative), averaged over all trials from 20 participants.	123
6-9	Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #4 (spread wings) and #5 (fold wings), averaged over all trials from 20 participants.	124

6-10	Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #10 (remove chocks) and #11 (insert chocks), averaging over all trials from 20 participants.	125
6-11	Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #18 (engage nosegear steering) and #19 (hot brakes), averaging over all trials from 20 participants.	126
6-12	Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #20 (brakes on) and #21 (brakes off), averaging over all trials from 20 participants.	127
6-13	Recognition accuracy rates of HCRF ($ \mathcal{H} =5$) and MIC-HCRF ($ \mathcal{H}^1 =3$, $ \mathcal{H}^2 =2$) when the dPS feature combination was used.	129
A-1	General Aircraft Handling Signals (Sheet 1 of 6).	140
A-2	General Aircraft Handling Signals (Sheet 2 of 6).	141
A-3	General Aircraft Handling Signals (Sheet 3 of 6).	142
A-4	General Aircraft Handling Signals (Sheet 4 of 6).	143
A-5	General Aircraft Handling Signals (Sheet 5 of 6).	144
A-6	General Aircraft Handling Signals (Sheet 6 of 6).	145

List of Tables

4.1	Parameters for each joint. Joint angle variables θ_i are determined during body pose tracking, while the other values are fixed once initialized ($\overline{o_i o_j}$ is the length between joints o_i and o_j).	55
4.2	Possible conditions for computing $\epsilon_{MHI}(I_{MHI}(z_t, z_{t-1}), I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})]))$. Note that, for the first two columns, the value 0 means there has been no object in the pixel, the value 127 means there was an object in the pixel but it has moved, and the value 255 means there is an object. Therefore, by thresholding the absolute subtracted values with the cutoff value 128, we can ignore the mistakes happened at $t - 1$ and concentrate on the mistakes that happened at time t	71
4.3	Joint angle range constraints.	74
4.4	A qualitative analysis result.	77
4.5	Joint displacement errors.	85
5.1	Hand pose classification accuracy rates.	100
6.1	Statistics for the recognition accuracies comparing two conditions, body pose only (BodyOnly) and body and hand pose combined (BodyHand), under the four different body pose features.	116

6.2	Mean and standard deviations of recognition accuracy rates for each feature combinations.	119
6.3	Mean and standard deviations of recognition accuracy rates of the dPS feature combination (a combination of derivatives of joint coordinates (dP) and vectors of probability estimates (S)) for each number of hidden states (3, 4, 5, and 6).	121
6.4	Recognition accuracy rates of HCRF ($ \mathcal{H} =5$) and MIC-HCRF ($ \mathcal{H}^1 =3$, $ \mathcal{H}^2 =2$) when the dPS feature combination was used.	129

Chapter 1

Introduction

1.1 Preface: “The future is gestural.”

Controller-based interaction (e.g., keyboard, mouse, joystick, etc) has been a primary way to deal with computer systems. It has allowed us to interact with complex systems efficiently, especially when one needs fine control of a system (e.g., photo/video editing, computer programming, music composition, etc.). However, this type of interaction is often both labor intensive (e.g., getting familiar with input devices and learning all the provided functionalities of the system) and requires an awkward interaction modality or an unintuitive definition of a functionality.

Gesture-based interaction, on the other hand, is a paradigm shifting approach to interacting with computer systems. It allows users to interact with systems by waving their arms, articulating specific gestures, directly touching a digital object, making facial expressions, etc., skills that most of us have naturally learned and used since birth. Compared to the conventional controller-based interaction, it does not require users to learn additional skill sets to control specially-designed input devices or to memorize the functionalities that are designed to be used with the input devices, allowing the users to concentrate on the task itself with less cognitive effort. Gesture-based user interfaces that are carefully designed

to resemble how humans naturally interact with objects in the real-world, therefore, can open a new horizon of future computing.

This shift –from the controller-based interaction to the gesture-based interaction– is getting growing attention in many research and industrial fields. In consumer electronics, for example, we see recent prototypes such as gesture-controlled televisions and interactive video game consoles (Figure 1-1). In these prototypes, gesture recognition allows people to use their own body parts to give commands via a sequence of articulated body poses. Therefore, in order to control a television or play an interactive video game, users no longer need to sit in front of a system holding a remote control or a joystick; they can simply wave their arms or hands, or hold a specific body pose in a predefined manner.



Figure 1-1: Hitachi gesture remote control television (left) and Microsoft Kinect video game console (right).

1.2 Goal and Motivations

1.2.1 Goal

A successful gesture recognition technology has a great potential in many application areas, including visual surveillance, virtual reality, home appliances, interactive video games, and natural human-computer/robot interactions.

The goal of this thesis work is to design and implement a multi-signal gesture recognition system that attends to multiple information channels, specifically, a combination of body and hand poses. Along with this, to avoid obtrusive and unnatural interaction, the system was built not to require any marker to be attached to the human body, but to perform motion tracking solely based on a single stereo camera and various computer vision techniques.

To make these goals possible, our system performs 3D body pose estimation and hand pose classification in a unified fashion, and uses results from the both to recognize gestures. In order to provide a principled statistical inference framework, we developed a novel approach for pattern recognition that considers input data from multiple information channels, capturing dependencies within the input data more precisely than conventional approaches to gesture recognition.

1.2.2 Motivations

How do humans gesture when communicating?

The first motivation comes from our intuition about the way humans gesture to communicate: we often make specific body poses as well as hand poses, and there are even occasions when it is impossible to understand the meaning of a gesture without seeing them both, indicating their importance. However, many of the current approaches to gesture recognition consider either body pose or hand pose alone, limiting their practicality for many real-world problems. In this work, we combine body and hand poses, allowing gesture recognition to deal with a richer gesture vocabulary, extending its practicality.

How do humans interpret multi-signal gestures?

The second motivation comes from the uncertainty about the way humans interpret gestures: when humans perceive gestures involving both body and hand poses, it is not clear

whether they consider body and hand poses dependently or independently (i.e., attempting to capturing or ignoring the dependencies between each input modality). Nor is it clear when information fusion occurs, that is, when we combine information on body and hand poses to extract meaning (i.e., before or after each input modality is processed, or both).

Previous work on multi-signal gesture recognition falls into one of the following categories, depending on its information fusion scheme: *feature-level fusion* performs recognition based on a single input vector combining all signals from different information channels, assuming the signals are independent; *decision-level fusion* performs recognition with each individual signal first, then makes the final decision based on the recognition results by, for example, selecting the one with the highest probability or majority vote.

In this work, instead, we take a new approach by not specifying the information fusion scheme, but rather letting the system decide how to handle dependencies among input signals or when to fuse them. We believe this has a potential to capture the complex nature of how human recognizes multi-signal gestures.

1.3 Gesture Recognition Pipeline

Figure 1-2 shows a pipeline diagram of our gesture recognition system¹. The system starts by receiving pairs of time-synchronized images recorded from a stereo camera. For the first part in the pipeline, *image pre-processing*, 3D range images (depth maps) are calculated, and images are background subtracted using depth information and a codebook background model. For the second part, *3D body pose estimation*, a particle filtering estimation framework is used to perform body pose estimation, comparing features extracted from both a parametric model and input images. For the third part, *hand pose classification*, the estimated wrist positions are used to guide hand pose classification. A multi-class

¹This diagram purposely hides implementation details as much as possible, while also trying to give a general overview of the gesture recognition system, so that the readers can refer to it for context as they read each chapter.

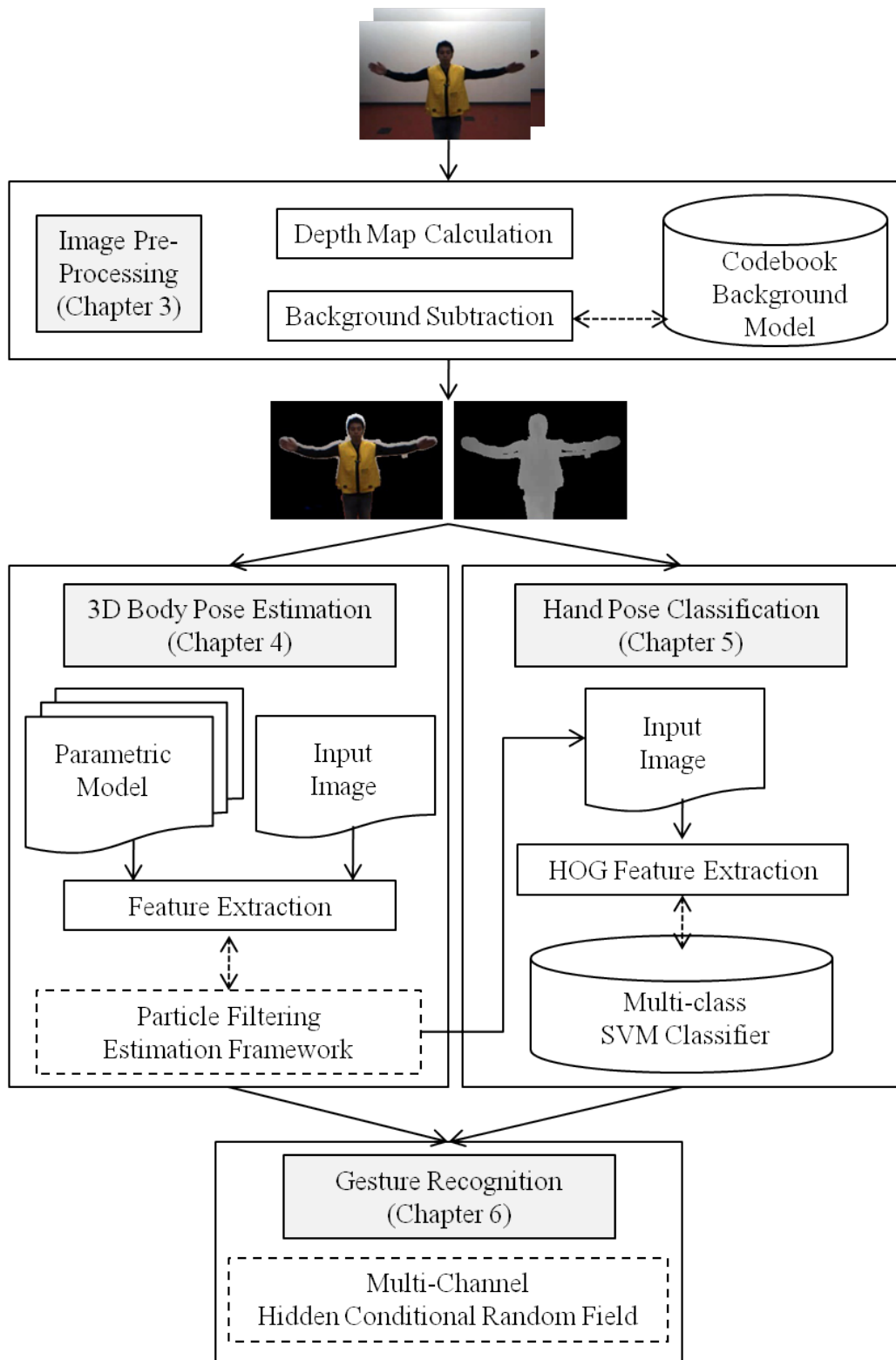


Figure 1-2: Gesture recognition pipeline.

Support Vector Machine (SVM) classifier is trained off-line using HOG features extracted from manually-segmented images of hands. Then hands are searched for small regions around the estimated wrist positions. In the last part, *gesture recognition*, we perform multi-signal gesture recognition using a combination of body and hand pose information. Multi Information-Channel Hidden Conditional Random Fields (MIC-HCRFs), a new approach we developed in this work, is trained off-line using a supervised gesture data set, and is used to perform gesture recognition.

1.4 Contribution

Our main contribution in this thesis work is three-fold:

- We designed and implemented a vision-based multi-signal gesture recognition system that takes into account both body and hand poses, thus allowing a richer set of gesture vocabulary and more natural human-computer interaction.
- We developed a novel statistical inference framework for multi-signal pattern recognition that is capable of capturing complex dependencies of multiple information channels in a principled manner, which we refer to Multi Information-Channel Hidden Conditional Random Fields (MIC-HCRFs).
- We applied our multi-signal gesture recognition system to an interesting real-world problem: an aircraft carrier flight deck environment. We showed that (a) the design of our system is well suited to the flight deck environment, (b) the recognition performance of our system is comparable to that of human pilots when tested on a subset of aircraft handling signals, and (c) although still in the proof-of-concept phase, our system has a great potential for deploying unmanned vehicles onto the deck environment.

1.5 Thesis Overview

Figure 1-2 shows an overview of this thesis. Chapter 2 introduces the context for this work, Chapter 3 describes image pre-processing, Chapter 4 describes 3D body pose estimation, Chapter 5 describes hand pose classification, and Chapter 6 describes gesture recognition framework. Chapter 7 concludes this thesis, listing contributions and suggesting directions for future work.

Chapter 2

LUVI Project

2.1 Background

This thesis work is a part of a multiple lab-wide project, “*Integrating Global and Local Situational Awareness in Distributed Unmanned and Manned Ground Operations*”, sponsored by the Office of Naval Research (ONR). A team of faculty members and students at MIT is working toward developing a next-generation aircraft carrier deck environment where manned and unmanned vehicles exist together. The project has three sub tasks, (a) developing global supervisory control decision support, (b) planning under uncertainty, and (c) enabling local unmanned vehicle interaction (LUVI). The concentration of this work is on the third sub task.

The goal of the LUVI project is to develop a framework that allows aircraft carrier flight deck personnel (hereafter ABHs—Aviation Boatswain’s Mate Handler) to interact naturally with unmanned combat aerial vehicles (UCAVs). The difficulty of the problem lies in the fact that there is no existing routine practice to study, i.e., there is no human-UCAV interaction in today’s carrier flight deck environment. The idea of having a natural interaction is also subtle and could easily be ill-posed. In this case, we specifically mean allowing deck personnel to interact with UCAVs in the same way they routinely interact with human

pilots. This in turn raises the question as to how ABHs currently communicate with pilots. Communication on the carrier flight deck is primarily gestural because jet engines are extraordinarily noisy (reaching up to 140 dB), and this led to defining a well established vocabulary of gestures that ABHs and Navy pilots can communicate each other. We believe a UCAV that understood the gesture vocabulary would fit in well, minimizing transition cost. This would also avoid the need for specially trained personnel (e.g., teleoperators) in a carrier flight deck environment that is already chaotic, again smoothing the integration of UCAVs into the existing environment.

Of particular interest in the LUVI project is the idea of allowing two-way communication between humans and UCAVs. It is important for UCAVs to be able to recognize human gestures. At the same time, it is also necessary for UCAVs to have an appropriate feedback mechanism, that is, UCAVs also have to be able to *gesture back*; just as a pilot would do in the same situation. Although both aspects are important, the concentration of this thesis work will be on gesture recognition by the UCAV, leaving the research on UCAV gesturing back to human as future work.

2.2 NATOPS: Constraints and Challenges

The Naval Air Training and Operating Procedures Standardization (NATOPS) manual standardizes general flight and operating procedures for the US naval aircraft. There are a large number of publications that belong to NATOPS; among the many, an aircraft signals manual (NAVAIR 00-80T-113 [53]) contains information about all aircraft systems including general aircraft and carrier flight deck handling signals.

Several things make it interesting to use the NATOPS aircraft handling signals as a gesture vocabulary for the LUVI project.

- First, since the aircraft handling signals are the ones that are currently used on the flight deck, the performance of the LUVI system can be measured in a realistic

scenario.

- Second, there are a lot of things going on in the carrier flight deck environment, which indicates that the aircraft handling signals should be designed to handle a wide range of situations to communicate. Also, the aircraft handling signals have been refined, modified, and optimized over the years, suggesting that the signals can be thought of as a well-defined gesture set.
- Third, the aircraft handling signals defined in the manual suggests interesting problems for gesture recognition. There are many similar gesture pairs that are having completely opposite meanings. For example, the “brakes on” and “brakes off” gestures are performed by raising both hands, with either open palms that are closed (“brakes on”), or closed hands that are opened (“brakes off”) (see the top row in the Figure 2-1). Here, the role of hand pose is crucial to differentiating these two very similar gestures with completely opposite meanings.

As a more subtle case: the “insert chocks” and “remove chocks” gestures are performed with both arms down and waving them in/outward (see the bottom row in the Figure 2-1). The only difference is the position of thumbs (inward and outward). The velocity of waving arms might be another indicator for differentiating the two gestures (going faster for the direction of thumbs), but it is not obvious whether even human eyes can catch the differences.

- Fourth, gestures performed by ABHs need to be perceived in 3D space. Many gestures defined in the manual are performed with self-occluding body poses, which are in general hard to reconstruct with a 2D image. Moreover, some gestures include directional information with pointing actions, which are lost in a 2D image.



Figure 2-1: Some of NATOPS gestures: (a) brake on, (b) brake off, (c) insert chocks, (d) remove chocks

2.3 Carrier Flight Deck: Constraints and Challenges

In addition to the challenges in recognizing gestures in the NATOPS manual, there are many constraints and challenges on the carrier flight deck environment that make the problem of gesture recognition even more interesting.

- First, for safety reasons, no active sensor is allowed on the flight deck. This includes any type of camera that uses an infrared light source. Therefore, many existing marker-based or active sensor-based motion tracking methods would not be acceptable.
- Second, we cannot rely on the system detecting colors for locating body parts in captured images, although flight deck personnel wear color-coded helmets and jerseys (e.g., yellow helmet and jersey for aircraft directors, as in Figure 2-1). This is because lighting conditions change through the day and the colors usually wear out.
- Third, since the camera will be mounted on a moving UCAV, the system should be viewpoint-invariant, i.e., robust to scale and rotation changes, and able to locate the

object of interest within a moving environment¹.

- Lastly, gestures performed by ABHs differ from person to person and from time to time: senior ABHs often simplify the standard gestures when they communicate with well experienced pilots; the fatigue effect simplifies the standard gestures as well. Therefore, we need a gesture recognition system that is robust to individual variations.

2.4 LUVI System Design Criteria

This work is aimed at the near future, where UCAVs are operating in close proximity to manned vehicles and deck personnel. In such a scenario, enabling the UCAVs to *see* and *understand* the aircraft handling-signals performed by ABHs, would mean using the same communication as between Navy pilots and ABHs, smoothing the integration of unmanned vehicles into the existing carrier flight deck environment.

The gesture recognition system described in this thesis is designed to work well with the constraints and challenges described above. The system does not require any marker to be attached to the tracking objects, and users do not have to wear specially designed clothes or gloves. The tracking is performed solely based on a single stereo camera, easing the installation of the system. Body poses are reconstructed using a generative model-based approach, which is relatively robust to viewpoint changes compared to other approaches. The approach is also capable of reconstructing body poses in 3D space, since the system takes advantage of stereo vision. Lastly, the statistical inference framework for gesture recognition, which is newly proposed in this work, is capable of learning complex dependencies of gestures from multiple input channels (e.g., body pose and hand pose), establishing the dependencies more systematically.

¹Viewpoint-invariant tracking is not implemented in this work but left as a future work.

Chapter 3

Image Pre-Processing

In this chapter, we describe image pre-processing, the first part in the gesture recognition pipeline. A stereo camera is used to capture pairs of time-synchronized images (Chapter 3.1), and depth maps are calculated using existing stereo vision techniques (Chapter 3.2). Then background subtraction is performed using a combination of codebook approach [34] and depth information, segmenting foreground object (human body) from the background (Chapter 3.3). The output of this step includes three types of images: color images, depth maps, and foreground-background mask images.

3.1 3D Vision Cameras

Being able to capture and process images in 3D is becoming crucial to many application areas including visual surveillance, interactive video games, navigation, and robotics. For that reason, many technologies have been developed to make a 3D vision camera that is capable of producing depth maps –an image in which each pixel value represents a distance to an object. We briefly review two types of 3D cameras currently widely used for human motion tracking: a time-of-flight camera and a stereo vision camera. We compare their advantages and disadvantages, and explain why we chose a stereo vision camera for this

work.

3.1.1 Time-of-flight Camera

Time-of-flight (ToF) cameras are becoming popular in these days for their ability to process images quickly and accurately¹ (Figure 3-1). The camera illuminates objects with high speed modulated light, often using infrared light to be unobtrusive, and captures the light reflected back at it. Then it calculates the distance to an object using the speed of light and the time the light took to travel back to the camera.

There are two main approaches to measuring distance using the ToF cameras: one method directly measures the time a light pulse traveled using arrays of single-photon avalanche diodes (SPADs) [61]; the other method uses amplitude modulated light to measure the phase difference between a reference signal and the reflected signal [39]. While the first approach suffers from complex readout schemes and low frame rates, the second approach has already been successfully implemented in industry.

The second approach, based on the phase-shift principle, uses the recent advances in semiconductors (i.e., CCD/CMOS imaging sensors) to measure for each pixel the phase of the returned modulated signal. The distance for each pixel is determined by the phase-shift principle, i.e., measuring a fraction of the one full cycle of the modulated signal [39].

Although the phase-shift approach allows us to obtain pixel-level accuracy depth maps, there are two main disadvantages of this approach. First, because of the modulation frequency cycle, there is a range limit on measuring distance without ambiguity, typically ranging around 16 feet (or 5 meters). This limit can be extended by lowering the modulation frequency, but this degrades the distance measurement accuracy. Second, the pixel resolution is low, usually about 320x240 pixels or less, so an object needs to be placed

¹Some of this popularity comes from the fact that Microsoft has released their new video game product, Kinect (formerly known as the project Natal,) which is a human body motion tracking device allowing users to use their body to control their avatars.

close to the camera to be captured with enough detail. A comprehensive evaluation of a CCD/CMOS ToF camera can be found in [17].



Figure 3-1: Various kinds of time-of-flight cameras. (a) SwissRanger SR-4000, (b) 3DV ZCam, (c) PMD Vision CamCube 3.0

3.1.2 Stereo Vision Camera

Stereo cameras, on the other hand, have been extensively used in many motion tracking approaches. They capture a pair of time-synchronized images from two horizontally aligned lenses with a fixed baseline. From the pair of images, they find matching pairs of points in the two images and compute a depth map using the principle of similar triangles (described in the next section).

For this work, we use a Bumblebee2 stereo vision camera from Point Grey Research Inc.² (Figure 3-2). There are a variety of reasons for us to choose a stereo vision camera over a time-of-flight camera. First, stereo vision cameras offer a higher resolution image compared to ToF cameras³. In general, deck personnel must keep a certain distance from the UCAVs (at least 50 feet or 15 meters) to ensure their safety; in such a case we would expect that low resolution images will not have enough detail to perform gesture recognition. Second, again for safety, everyone on the flight deck must wear a color-coded jersey which has

²The camera used in this work is BB2-03S2C-38; 640x480 resolution, two 3.8mm color lenses with 12cm baseline, 48 FPS, 65-degree HFOV.

³Bumblebee2 stereo vision cameras offer a maximum resolution of 1280x960 pixels, which is 16 times larger than the state-of-the-art ToF cameras (320x240 pixels)

reflectors on it. However, distance data obtained from a ToF camera can be significantly degraded if a scene contains materials with high reflectivity (e.g., mirror or reflectors) or light emitting objects (e.g., the Sun) [28]. Therefore, unless we change the current design of the jerseys, using ToF cameras on the flight deck will be problematic. Lastly, ToF cameras use infrared light to illuminate objects of interest. However, as described in Chapter 2, any source of infrared light is undesirable on the carrier flight deck, as it is easily detectable from reconnaissance.



Figure 3-2: Bumblebee2 stereo vision camera from Point Grey Research Inc.

3.2 Depth Map Calculation

In this section, we briefly review the process of obtaining a depth map from a pair of time-synchronized images captured using a stereo camera. To build an intuition, let's take the simplest example as shown in Figure 3-3. In this setting, we assume that

- captured images are undistorted;
- focal length f and baseline T are known and fixed;
- two image planes are coplanar, have parallel *principal rays* (the ray from the center of projection O through the *principal point* c_x), and horizontally aligned (same points always appear on the same row in both images);

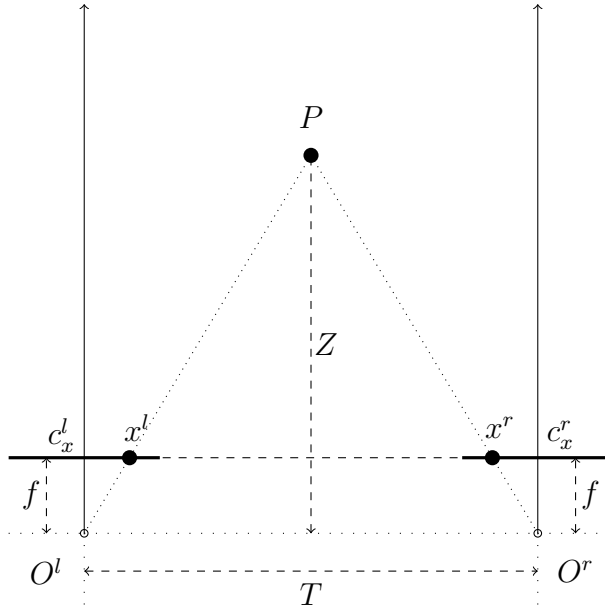


Figure 3-3: A simplified stereo camera setting to illustrate the basic idea of computing a depth map. The depth Z to a point P is calculated using the principle of similar triangles, using a baseline length T , focal length f , and principal points c_x^l and c_x^r .

- the principal points c_x^l and c_x^r are calibrated to have the same coordinate system in their respective images;
- a matching pair of the point P is found in both images, shown as x^l and x^r .

With this simplified setting, the depth Z (i.e., the distance to the point P) can be calculated using the principle of similar triangles, that is

$$\frac{Z}{T} = \frac{Z - f}{T - (x^l - x^r)}, \quad Z = \frac{fT}{x^l - x^r} \quad (3.1)$$

where $x^l - x^r$ is called *disparity*.

In the real situation, however, many of the above assumptions usually do not hold. The surface of lens is spherical, not parallel to the projected image plane; so we need to correct lens distortions in the image, called *undistortion*. Also, the pairs of captured images are neither coplanar nor horizontally aligned; so we need to mathematically align the image

planes, called *rectification*. After the two processes are applied to a pair of images, we can compute a disparity map by solving the *correspondence problem*, which finds matching points between two images. Note that the rectification process horizontally aligns a pair of images, so it simplifies the correspondence problem to one-dimensional search (since points on a row in one image are expected to be in the same row in the other image). Lastly, after the disparity maps are obtained, we can calculate depth maps using the principle of similar triangles (Eq 3.1).

In this work, we captured pairs of time-synchronized 640 x 480 color images at 20 FPS, and used the manufacturer provided SDK [31] to obtain depth maps, which does all processes explained above.

3.3 Background Subtraction

Background subtraction is one of the most crucial preprocessing step in many computer vision problems. The goal of background subtraction is to learn a background model to segment out foreground objects given a sequence of images, usually captured from a camera at a fixed location⁴. This allows us to concentrate on the objects of interest and ignore the background, optimizing the use of available computational resources.

Many challenges make the problem of background subtraction complex, including the illumination condition, shadows, gradual or sudden background changes, etc. The definition of foreground objects is also subtle and largely depends on the context. Imagine a moving tree in the background; should it be considered as a background object or a foreground object? What if clouds pass by the Sun and the illumination condition changes? Can we make background subtraction adaptable to these sudden/gradual changes?

In this work, we perform background subtraction with a combination of the codebook approach [34] and depth information: input images are first background subtracted following

⁴Recent approaches also deal with moving backgrounds, such as [18].

the codebook approach, then refined using depth information, filtering out pixels where the distance is further from camera than a foreground object.

3.3.1 The Codebook Approach

The codebook approach assumes that a foreground object will have pixel intensity values that are noticeably different from the history of background pixel intensity values; hence the method works by learning a background model with a history of background images sampled for a period of time, then segmenting out the *outlier* pixels in an input image as a foreground object. The strength of the method comes from the fact that it models the history of each pixel's intensity values as a multimodal distribution, allowing us to capture dynamic variations of the background.

For each pixel, a multimodal distribution is constructed with a set of disjoint intensity bounds, where each intensity bound is determined by a new value observed for the pixel. If the value falls into or is close to one of existing bounds, it is modeled as a perturbation on that bound, making the bound grow to cover the perturbation of values seen over time; otherwise, a new intensity bound is created and added to the set. The set of disjoint intensity bounds are called *codebook*, and can be envisioned as several boxes located in RGB space, each box capturing a particular intensity range considered likely to be background (Figure 3-4).

It has been empirically shown that most of the variation in background is better represented along the brightness axis, not the color axis [10]. Therefore, we transform input images into the YUV space for background subtraction, where Y stands for the *luma* component (the brightness) and U and V are the chrominance (color) components.

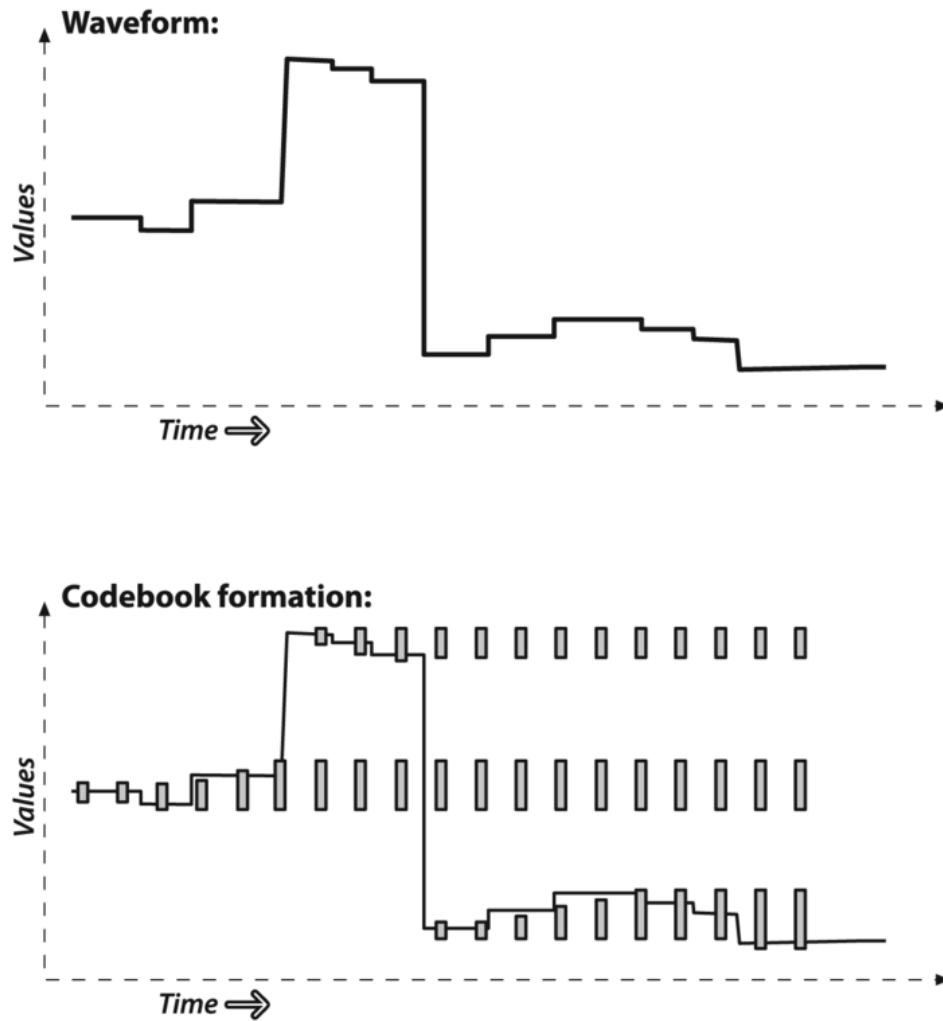


Figure 3-4: A codebook is made up of several intensity bounds that grows to capture a history of intensity values in a particular range. The top image show a history of intensity values over time, and the bottom image shows a set of disjoint intensity bounds.(reproduced from [10])

3.3.2 Refinement Using Depth Information

Using the codebook approach alone, a background subtracted image may not segment out the shadows created by foreground objects properly. This is because the codebook approach defines a foreground object as a set of pixels where their intensity values are noticeably different from the history of intensity values for background; so the shadows are foreground objects in the context of the codebook approach.

To remedy this, after an image is background subtracted using the codebook approach, we refine the result using depth information. We filter out pixels where the distance is further than a foreground object, with an assumption that the foreground object is located closest to the camera.

Figure 3-5 shows the result of image pre-processing step, including depth map calculation and background subtraction.

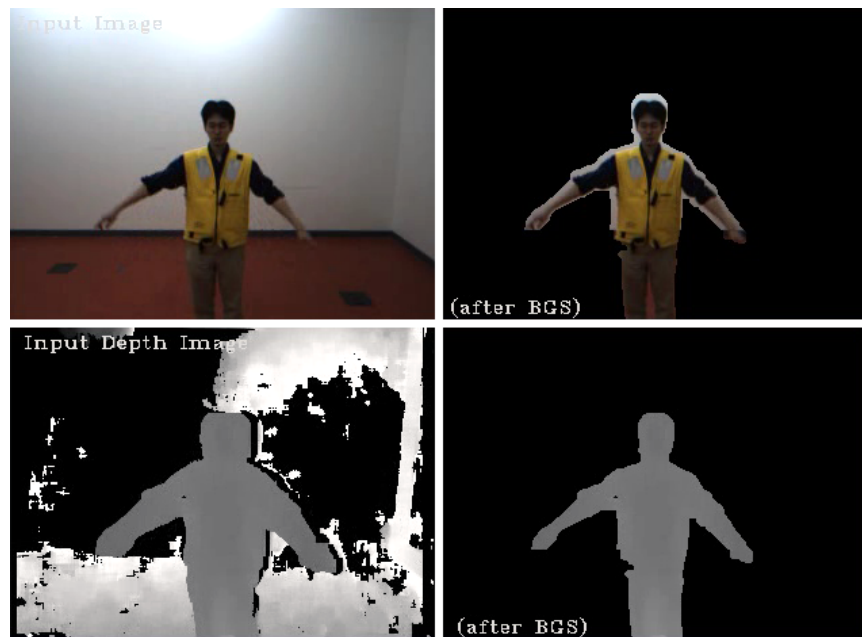


Figure 3-5: An overview of the image pre-processing step. The top-left image is a color image obtained from a stereo camera, and the bottom-left image is a depth map. Two images on the right shows background subtracted images.

Chapter 4

3D Body Pose Estimation

The problem of estimating 3D body pose is a long-standing challenge. Many things make the problem difficult. Among the first is the high-dimensionality of the human body's kinematic model. Representing a single rigid body in 3D space requires at least six variables (global translation and rotation); adding a joint to the model requires at least one additional variable (linear displacement or rotation about an axis). This leads to more than 30 variables needed to represent a full human body model. Searching in such a high dimensional space consumes substantial amounts of processing time, which in turn limits its practicality in real-time applications. The problem is compounded by many other challenges, such as self-occlusion, loose clothing, lighting conditions, and available sensor technology.

In this chapter, we describe a generative model-based 3D body pose estimation framework, the second part in the gesture recognition pipeline. A parametric model of the human upper body is constructed in 3D space, representing the skeletal model as a kinematic chain, and modeling the shape of the body with superellipsoids (Chapter 4.1). This model is then fitted to an input image by comparing several features extracted from an input image and generated from the parametric model (Chapter 4.2). Finally, a particle filter [33] is used to estimate 3D upper body pose in a high dimensional space (Chapter 4.3). The estimated

result has been evaluated both qualitatively (by visually comparing it to the input image) and quantitatively (by measuring pixel displacement errors using the ground-truth data obtained from a Vicon motion capture system) (Chapter 4.4). We conclude this chapter by reviewing other approaches to body pose estimation (Chapter 4.5).

4.1 Generative Model of Human Upper Body

4.1.1 Skeletal Model

We construct the skeletal model of 3D human upper body as a kinematic chain consisting of 6 body parts (head, trunk, upper and lower arms for both sides), parametrized by 14 variables (8 for angle rotation and 6 for global orientation). Coordinates of each joints are obtained by solving the forward kinematics problem following the Denavit-Hartenberg convention [21], which is a compact way of representing n -link kinematic structures. We improve on this basic model by building a more complex model of the human shoulder: in order to accurately capture the human arm movement while not including additional DOFs, the shoulder model is approximated analytically by computing the chest-to-shoulder angle relative to the elbow position.

Forward Kinematics Problem

The forward kinematics problem is concerned with determining the position and orientation of the end-effector, given the values of joint variables for the intermediary joints. A kinematic chain is defined as a set of links, and constructing it is typically done by constraining each joint to have a single free variable, either the angle of rotation for a revolute joint or the linear displacement for a prismatic joint¹.

¹More complex joints, such as a ball-and-socket joint with 3 DOF, can be thought of as a succession of 1 DOF joints with zero length links.

Let the i -th joint connect the $(i-1)$ -th link to the i -th link, and let A_i be a 4 x 4 homogeneous transformation matrix for the i -th joint. Then we can express any arbitrary coordinate frame $o_j x_j y_j z_j$ of the j -th joint with respect to the i -th joint as a cumulative effect of all matrices in between; this can be expressed as a single homogeneous transformation matrix T_j^i :

$$T_j^i = A_{i+1} \cdots A_{j-1} A_j \quad (i < j) \quad (4.1)$$

What remains to solve is to determine A_i for each joint. For a 2-link kinematic chain, this is not so complicated, since there is only one joint. However, it can get quite complex if a kinematic chain has more than two joints. The Denavit-Hartenberg convention [21] simplifies this by making two assumptions in constructing a kinematic chain (illustrated in Figure 4-1):

(DH1) Axis x_i is perpendicular to axis z_{i-1}

(DH2) Axis x_i intersects axis z_{i-1}

Under these assumptions, A_i can be expressed as a combination of rotation and translation matrices:

$$\begin{aligned} A_i &= Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,l_i} Rot_{x,\delta_i} \quad (4.2) \\ &= \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \delta_i & -\sin \delta_i & 0 \\ 0 & \sin \delta_i & \cos \delta_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \delta_i & \sin \theta_i \sin \delta_i & l_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \delta_i & -\cos \theta_i \sin \delta_i & l_i \sin \theta_i \\ 0 & \sin \delta_i & \cos \delta_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

where the four parameters θ_i , d_i , δ_i , l_i are measured as follows (illustrated in Figure 4-1).

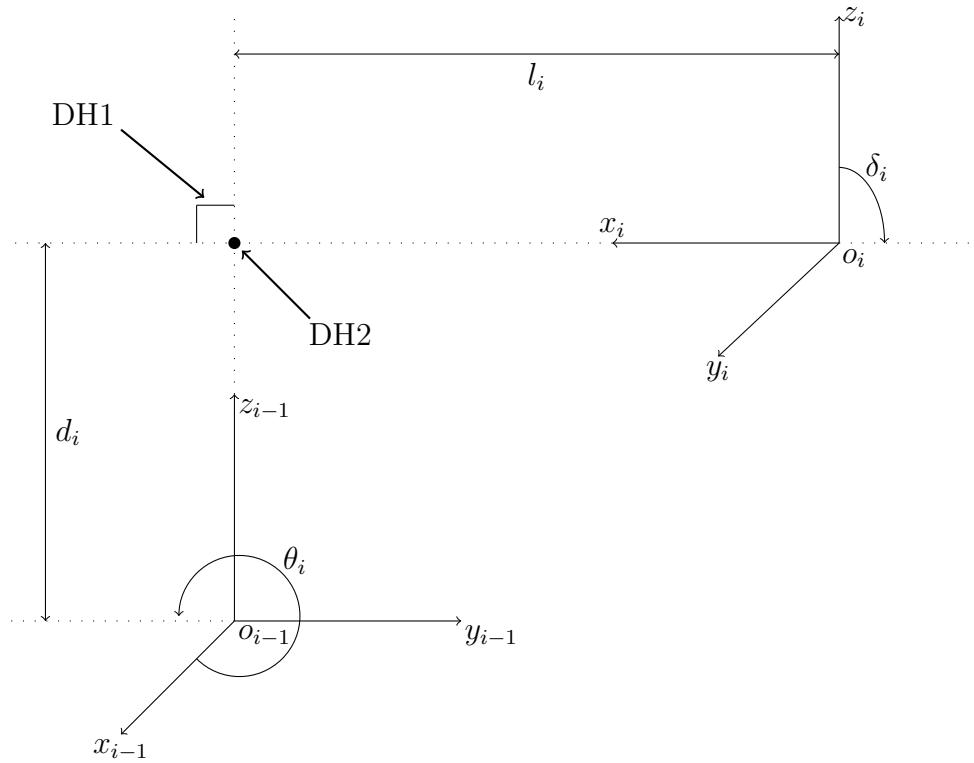


Figure 4-1: Two coordinate frames $o_{i-1}x_{i-1}y_{i-1}z_{i-1}$ and $o_i x_i y_i z_i$, constructed following the Denavit-Hartenberg convention. Note that the two assumptions in the Denavit-Hartenberg convention are illustrated (DH1 and DH2). DH1 shows axis x_i is perpendicular to axis z_{i-1} , and DH2 shows axis x_i intersects axis z_{i-1} . (reproduced from [65]).

- The parameter θ_i is the angle between the x_{i-1} axis and the x_i axis measured in a plane normal to z_{i-1} , following the right-hand rule (i.e., align the thumb to axis z_{i-1} and measure the angle counter-clockwise).
- The parameter d_i is the distance between the origin o_{i-1} and the intersection of the x_i axis with z_{i-1} axis, measured along the z_{i-1} axis.
- The parameter δ_i is the angle between the z_{i-1} axis and the z_i axis, measured in a plane normal to x_i , also following the right-hand rule.
- Lastly, the parameter l_i is the distance between the z_{i-1} axis and the z_i axis, measured along the x_i axis.

If we set the z_i axis to be the actuating axis for the next joint and construct a kinematic chain following the Denavit-Hartenberg convention, the parameter θ_i becomes the angle of rotation of a revolute joint and the parameter d_i becomes the linear displacement of a prismatic joint. Now that each joint is constrained to have a single free variable, we can obtain A_i simply as a function of either one of the two parameters: θ_i or d_i .

Skeletal Model Construction

Having explained the forward kinematics problem, now we describe the way we construct a skeletal model of human upper body.

We first need to define modules to build a skeletal model of a human upper body. As seen in Figure 4-2, there are 17 modules: 8 limbs (neck, trunk, L/R collar bones, L/R upper arms, and L/R lower arms) and 9 joints (head, chest, navel, L/R shoulders, L/R elbows, and L/R wrists). A shoulder is modeled as a ball-and-socket joint with 3 DOF and an elbow is modeled as a revolute joint with 1 DOF.

With the above definitions of modules, the skeletal model is constructed by choosing the chest joint as the base frame and constructing four kinematic chains for the rest of the

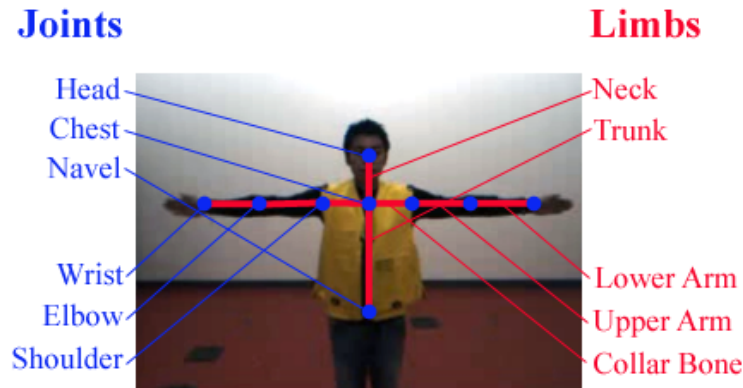


Figure 4-2: A skeletal model of human upper body. There are 8 limbs (neck, trunk, L/R collar bones, L/R upper arms, L/R lower arms) and 9 joints (head, chest, navel, L/R shoulders, L/R elbows, L/R wrists).

body – the head, trunk, and each arm – following the Denavit-Hartenberg convention (Figure 4-3).

The head and the trunk parts are constructed using one prismatic joint, and both arms are constructed using a series of joints; one prismatic joint connects the collar bone, three successive revolute joints model a ball-and-socket joint for the shoulder, one prismatic joint connects the upper arm, one revolute joint models the elbow, and one one prismatic connects the lower arm.

Although values for the prismatic joints (i.e., the limb lengths) are adjustable, our skeletal model assumes that these values are fixed once they are initialized. Therefore, once limb lengths are initialized, the local figure of the kinematic chain can be parametrized with a total of 8 joint angle variables:

- θ_2 : the left shoulder Z-axis rotation;
- θ_3 : the left shoulder Y-axis rotation;
- θ_4 : the left shoulder X-axis rotation;

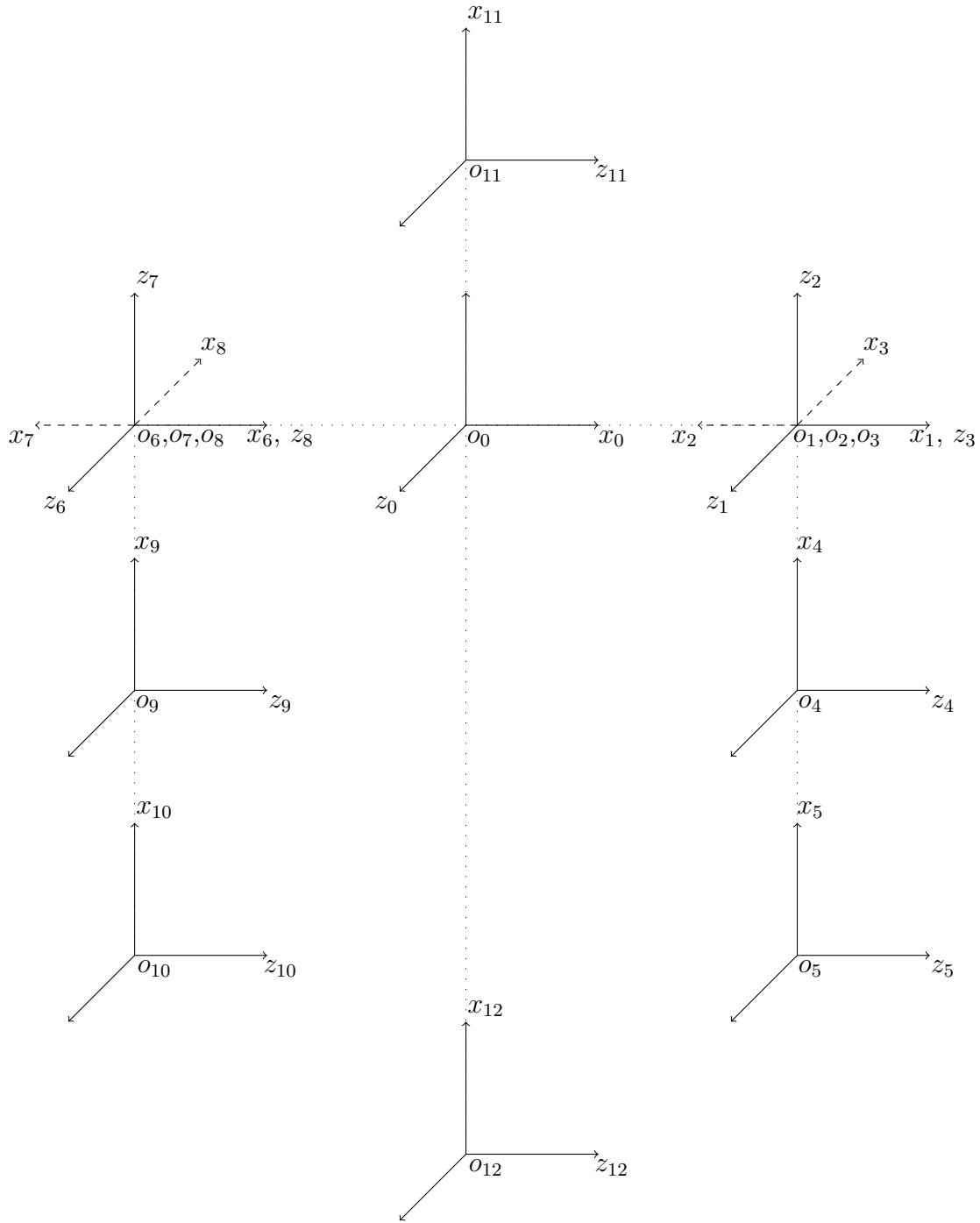


Figure 4-3: A kinematic chain of the human upper body. The coordinate frames are assigned as follows: o_{11} - head, o_0 - chest, o_{12} - navel, $(o_1, o_2, o_3)/(o_6, o_7, o_8)$ - left/right shoulder, o_4/o_9 - left/right elbow, o_5/o_{10} - left/right wrist.

- θ_5 : the left elbow rotation;
- θ_7 : the right shoulder Z-axis rotation;
- θ_8 : the right shoulder Y-axis rotation;
- θ_9 : the right shoulder X-axis rotation;
- θ_{10} : the right elbow rotation.

In addition to these 8 variables, 6 variables are added to control the global orientation of the kinematic chain (3 for translation and 3 for rotation).

After constructing the four kinematic chains, we initialize values for all the variables defining the kinematic chains except for the 8 joint angle variables.

- Values for d_i and δ_i are directly derived from the way we construct the skeletal model.
- Values for l_i are set at the initialization step (described in 4.3.3).
- Values for θ_i are set in the following way:
 - θ_1 (chest-to-left shoulder Z-axis rotation) and θ_6 (chest-to-right shoulder Z-axis rotation) are set to 0 degree and 180 degree, respectively;
 - values for θ_{11} (chest-to-head Z-axis rotation) and θ_{12} (chest-to-naval Z-axis rotation) are set to 90 degree;
 - the rest are the 8 joint angle variables.

Table 4.1 shows the initial values for the four parameters θ_i , d_i , δ_i , l_i .

During the tracking, whenever any of the joint angle variables is changed, we re-calculate the transformation matrices A_i and update the kinematic model.

Joint	θ_i	d_i	δ_i	l_i
1	0	0	0	$\overline{o_0o_1}$
2	θ_2	0	90	0
3	θ_3	0	90	0
4	θ_4	0	90	$\overline{o_3o_4}$
5	θ_5	0	90	$\overline{o_4o_5}$
6	180	0	90	$\overline{o_0o_6}$
7	θ_7	0	90	0
8	θ_8	0	90	0
9	θ_9	0	90	$\overline{o_8o_9}$
10	θ_{10}	0	90	$\overline{o_9o_{10}}$
11	90	0	90	$\overline{o_0o_{11}}$
12	90	0	90	$\overline{o_0o_{12}}$

Table 4.1: Parameters for each joint. Joint angle variables θ_i are determined during body pose tracking, while the other values are fixed once initialized ($\overline{o_i o_j}$ is the length between joints o_i and o_j).

Human Shoulder Model

The human shoulder has historically been the most challenging part for human body modeling [24]. It has a complicated anatomical structure, with bones, muscles, skin, and ligaments intertwined, making modeling of the shoulder movement difficult. One can easily see this by tracking the pivot point of a shoulder while raising an arm or doing a shrugging gesture; the pivot point is not fixed, but changes position in 3D space as the arm moves.

Although having a high fidelity shoulder model is the basis for a successful body pose tracking, many approaches in the generative model-based body pose estimation sacrifice some accuracy for simplicity, usually modeling the shoulder as a single ball-and-socket joint. In the biomechanics community, there have been many approaches to more sophisticated shoulder models (see [26] for a survey). In that community, most approaches have used a model with 5 to 9 DOF to model the shoulder accurately. Although these models offer high fidelity, having a higher DOF model is not desirable in body pose estimation, as it makes the estimation problem undesirably more difficult.

In this work, we approximate the human shoulder movement analytically by computing

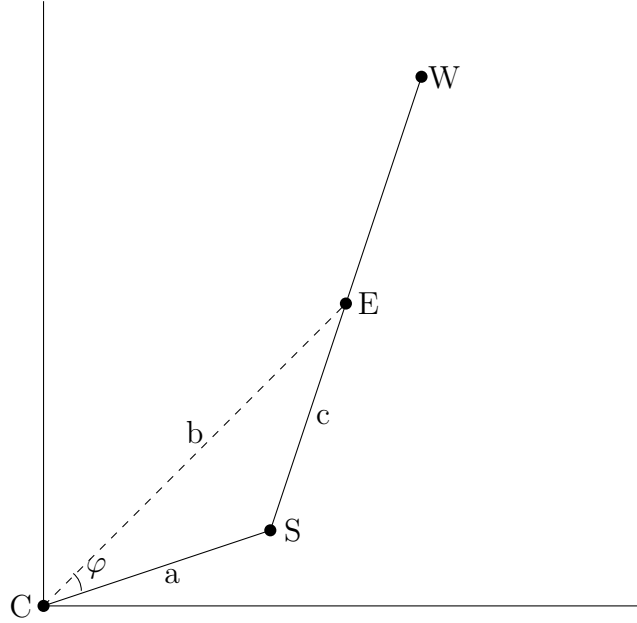


Figure 4-4: Simplified shoulder model. After joint positions are determined using the forward kinematics, φ is calculated with positions of chest (C), shoulder (S), and elbow (E) joint points, using the law of cosines.

the angle φ using the position of elbow joint points (Figure 4-4). First, using the law of cosines (Eq. 4.3), the angle φ between the line \overline{CE} and the line \overline{CS} is calculated. Then the chest-to-shoulder angle θ in the kinematic chain (θ_1 for the left and θ_6 for the right shoulder) is updated using Eq. 4.4, where θ_{MAX} and θ_{MIN} are constants that determine the maximum and minimum possible values for shoulder angle adjustments, respectively. Figure 4-5 shows several shoulder angles analytically computed using our shoulder model.

$$\varphi = \arccos \frac{a^2 + b^2 - c^2}{2ab} \quad (4.3)$$

$$\theta' = \begin{cases} \theta + \frac{\varphi}{\theta_{MAX}} & \text{if elbow is higher than shoulder} \\ \theta - \frac{\varphi}{\theta_{MIN}} & \text{otherwise} \end{cases} \quad (4.4)$$

One advantage of our shoulder model is its simplicity: the shoulder is modeled to have 3 DOF, so it still maintains a low DOF, while efficiently approximating the shoulder move-

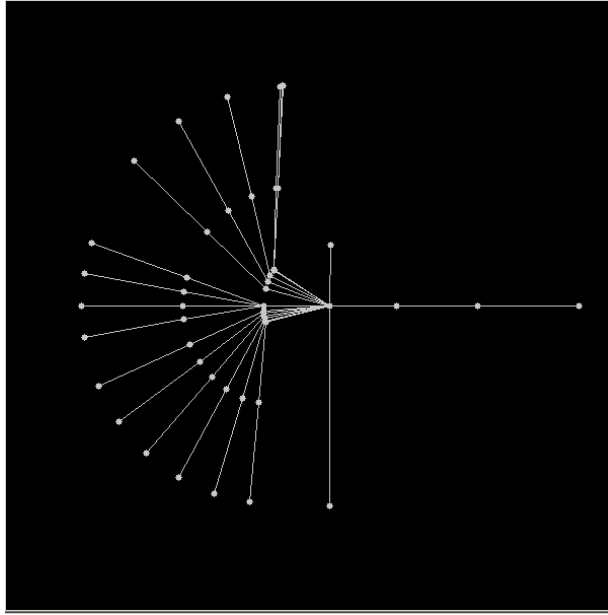


Figure 4-5: An overlaid image of the right shoulder’s movement. Note that the height of the shoulder joint point goes up and down, depending on the right elbow’s position.

ment. This simplified model is not a perfect reconstruction of the real human shoulder of course, because it only mimics shoulder movement in one-dimension: up and down. Nevertheless, this method works quite well if a human being tracked is facing the camera and does not rotate the entire body to other directions. In our scenario, ABHs are assumed to be facing the pilot (or the camera) for the most of the time, so we can expect that this method will help tracking body pose successfully (more detailed discussion of this can be found in Chapter 5.1).

4.1.2 3D Shape Model

The volumetric model of each body part can be based on various 3D shape primitives, including 3D cones, cylinders, ellipsoids, or superellipsoids. Superellipsoids [8] allow a compact representation of various 3D shapes while at the same time offering more flexibility than other shape primitives. In this work, therefore, superellipsoids are used to model the body shape (Figure 4-6).

A superellipsoid is defined by a function

$$\left(\left| \frac{x}{r_x} \right|^{2/e_2} + \left| \frac{y}{r_y} \right|^{2/e_2} \right)^{e_2/e_1} + \left| \frac{z}{r_z} \right|^{2/e_1} = 1 \quad (4.5)$$

where e_1 and e_2 are the shape parameters (e_1 for the z -axis and e_2 for the x - y plane) and r_x , r_y , and r_z are the scaling parameters for each axis. A point on the surface is expressed by the equation

$$\mathbf{p}(\alpha, \beta) = \begin{bmatrix} r_x \cos^{e_1}(\alpha) \cos^{e_2}(\beta) \\ r_y \cos^{e_1}(\alpha) \sin^{e_2}(\beta) \\ r_z \sin^{e_1}(\alpha) \end{bmatrix} \quad (4.6)$$

and the surface normal is expressed by the equation

$$\mathbf{n}(\alpha, \beta) = \begin{bmatrix} \frac{1}{r_x} \cos^{2-e_1}(\alpha) \cos^{2-e_2}(\beta) \\ \frac{1}{r_y} \cos^{2-e_1}(\alpha) \sin^{2-e_2}(\beta) \\ \frac{1}{r_z} \sin^{2-e_1}(\alpha) \end{bmatrix} \quad (4.7)$$

where α and β represent the longitude $-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$ and the latitude $-\pi \leq \beta \leq \pi$.

As seen in Figure 4-6, we use six superellipsoids to build up a volumetric model of the upper body. Except for the head, the shape parameters e_1 and e_2 of all superellipsoids are configured so that the eccentricity along the z -axis is higher than the eccentricity along the x - y plane.

4.2 Feature Extraction

Three kinds of features are extracted from the parametric model and an input images: a visible surface point cloud, a contour point cloud, and a motion history image. These features are then compared to calculate the likelihood function in the pose estimation framework.

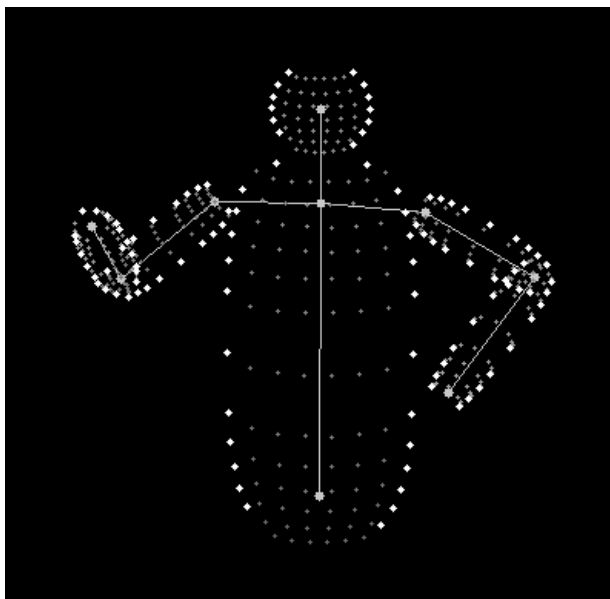


Figure 4-6: A skeletal model and shape model of human upper body. Contour points are shown in the white color, while visible surface points are shown in the gray color.

The two features, a visible surface point cloud and a contour point cloud, allow us to evaluate how well the shape of the parametric model fits to the actual human body shape in an input image, by computing the sum of closest distances between points from one to another.

The third feature, a motion history image [9], is a static image where each pixel value is a function of the recency of motion in a sequence of images. This often gives us useful information about motion, as it indicates *where* and *how* the motion occurred in a sequence of images. Therefore, by comparing two motion history images obtained from the parametric model and input images, we expect to capture discrepancies in the dynamics of motion, as compared to the point clouds where we capture point-to-point discrepancy between the parametric model and input images.

In the following two sections, we explain the feature extraction methods for the parametric model (4.2.1) and for input images (4.2.2).

4.2.1 Model Feature Extraction

Surface Point Cloud and Contour Point Cloud

We want to collect point clouds both from the parametric model and the input image, so that we can make a comparison of an estimated body pose and the actual human body pose in the image, quantitatively measuring how similar they are. Ideally, the two point clouds will have a well-balanced density of points, since the comparison will include calculating the distances between a pair of closest points from one to another.

A naive way to obtain the parametric model’s surface point cloud would be for each superellipsoid to sample points at a uniform longitude and latitude interval. Note that in this approach the sampling is performed *before* the parametric model is projected onto the image plane; this will cause more points to be sampled where the curve is changing rapidly, because after the surface points are projected onto an image plane we will see more points gathered on edges of a superellipsoid, i.e., on a sharply curved surface. This can be problematic when comparing the parametric model’s surface points to the input image’s surface points, as in the later case more points are sampled from the center of an object (since the sampling is performed *after* an object is projected onto an image plane).

Another way to obtain the parametric model’s surface point cloud would be to collect surface points with a probability inversely proportional to the distance to the less curved area, i.e., lower probability on the edge points. This will cause more points to be collected where the curve is changing slowly; the property that we want. Therefore, we use this approach to collect a surface point cloud from the parametric model.

The sampling procedure is as follows. A set of longitude (α) and latitude (β) values are randomly selected with the probability described above, and a set of points are sampled by applying the selected α and β values to Eq. 4.6. Note that our supersllipoids for the trunk and arms are configured so that the eccentricity along the z -axis is higher than the eccentricity along the x - y plane. Thus it is the angle of longitude (α) that primarily affects

whether the points are sampled near the equator, while the angle of latitude (β) can be set at a uniform interval. Pseudocode for the sampling procedure is shown in Algorithm 4.1.

Algorithm 4.1 Pseudocode for sampling surface points

Require: $|\alpha|$ (number of longitude lines), $|\beta|$ (number of latitude lines), and N (number of multiples the longitude lines to be sampled) are given

```

 $d\alpha \leftarrow \frac{\pi}{N|\alpha|}$ 
 $d\beta \leftarrow \frac{2\pi}{|\beta|}$ 
for  $i=1$  to  $N|\alpha|$  do
     $\alpha[i] \leftarrow -\pi/2 + (i-1) * d\alpha$  //Sample N more data for the longitude angle
end for
for  $i=1$  to  $|\beta|$  do
     $\beta[i] \leftarrow -\pi + (i-1) * d\beta$ 
end for
accepted points  $\leftarrow 0$ 
while accepted points  $< |\alpha| |\beta|$  do
    for  $i=1$  to  $N|\alpha|$  do
         $v \leftarrow \text{uniform}(0,1)$ 
         $\text{prob}(\alpha[i]) \leftarrow \frac{\pi/2 - \alpha[i]}{\pi/2}$ 
        if  $v \geq 1 - \text{prob}(\alpha[i])$  then
            Remove  $\alpha[i]$  from the set  $\alpha$ 
            for  $j=1$  to  $|\beta|$  do
                add  $\alpha[i], \beta[j]$  to the sampled point set
                accepted points  $\leftarrow$  accepted points + 1
            end for
        end if
    end for
end while

```

Once the surface point cloud is sampled, we need to check each point's visibility to ensure that the points extracted from the cloud contain only points that are visible in the input image: this can be checked with the dot product of $\vec{\mathbf{n}}$ (the point's surface normal) and $\vec{\mathbf{v}}$ (the vector from the camera to the point) as the following:

$$\vec{\mathbf{n}} \cdot \vec{\mathbf{v}} \begin{cases} < 0 : \text{invisible} \\ = 0 : \text{on the edge} \\ > 0 : \text{visible} \end{cases} \quad (4.8)$$

Sampling the contour point cloud requires a bit more care than just using the dot product rule in Eq. 4.8, since it will be computationally inefficient to directly compute the exact values of α and β that set the dot product value to zero.

We collect the contour point cloud by scanning longitude and latitude values with a uniform interval, searching for a pair of longitude (or latitude) values that changes the sign of the dot product value. Changing the sign would mean that there is a contour point somewhere in the middle of the pair; for each pair we compute the geometric mean of the two points, and add that point to the contour point cloud. The resulting visible surface point and contour point cloud is seen in Figure 4-6.

Motion History Image: Parametric Model

We compute a motion history image for the parametric model in the following way. For each time step t we render an 8-bit unsigned integer binary image I_{cur} (pixel values span 0 to 255) using the collected visible point cloud; we draw each point from the cloud on I_{cur} with its value set to 255. This image will be stored in memory until the next time step $t + 1$. Once we have two time-consecutive images I_{prev} and I_{cur} rendered using the previous and the current visible surface points, we compute a motion history image I_{MHI} using the following rule:

$$I_{MHI}(I_{cur}, I_{prev}) = thresh(I_{prev} - I_{cur}, 0, 127) + thresh(I_{cur} - I_{prev}, 0, 255) \quad (4.9)$$

where $thresh(I, \alpha, \beta)$ is a binary threshold operator that sets each pixel value to β if $I(x, y) > \alpha$, and set to *zero* otherwise; the values 127 and 255 are chosen to indicate the time information of those pixels (this will become clear in Section 4.3.2). The first term in Eq. 4.9 captures the pixels that were occupied at the previous time step but not in the current time step, while the second term captures the pixels that are newly occupied in the current time step.

To give a more concrete example, imagine an arm moved while the body trunk stayed still.

A motion history image will capture the history of the arm's movement by setting previous pixel position values to 127, the current pixel position values to 255, and the rest to zero. Note that this method allows us to construct an image that concentrates on the moved regions (e.g., arms) only, while ignoring the unmoved parts (e.g., trunk). This situation is depicted in Figure 4-7.

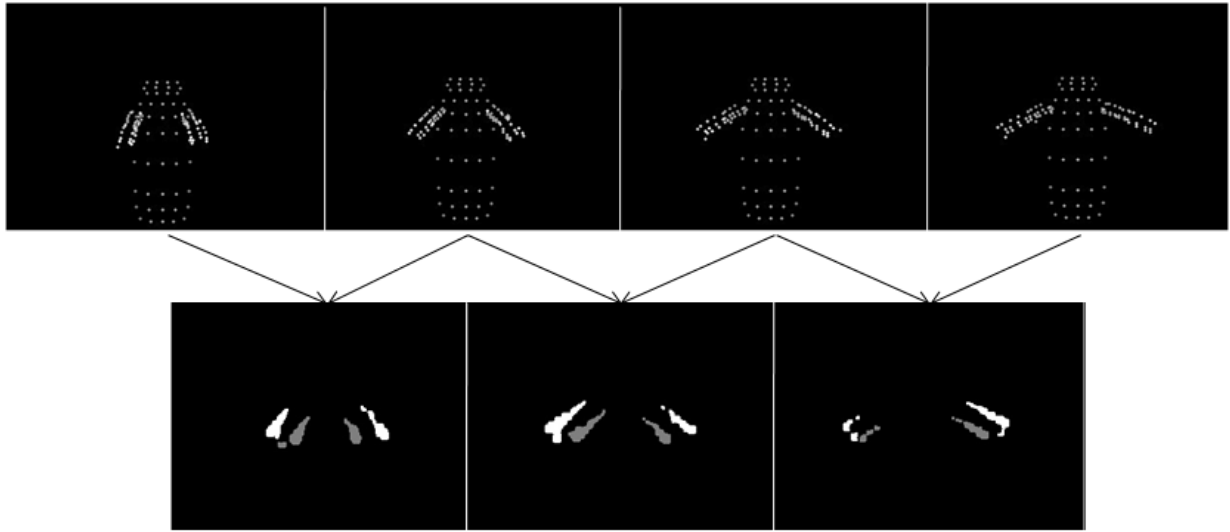


Figure 4-7: Motion history images (bottom) generated from pairs of surface point clouds (top). In the bottom row gray pixels indicate where the arms moved from, white pixels indicate where the arms moved to.

4.2.2 Image Feature Extraction

Surface Point Cloud And Contour Point Cloud

Collecting surface points from an input image is simpler than the one we did for the parametric model. We obtain surface points from a depth map that has been background subtracted, subsampling with a fixed interval in both the x and y direction (see Figure 4-8 (c)).

To collect points on the contour, we first perform edge detection, where by 'edge' we mean

a sharp change in depth. Hence, depth maps are used to collect contour points. The sampling procedure is as follows: we reduce noise in the depth image by using several morphological transformations; both closing and opening transformations are applied to a depth map²; Gaussian smoothing is applied to further remove the noise; Canny edge detection [14] is performed on the depth data. Finally, contour points are subsampled with the same method used for sampling surface points (see Figure 4-8 (d)).

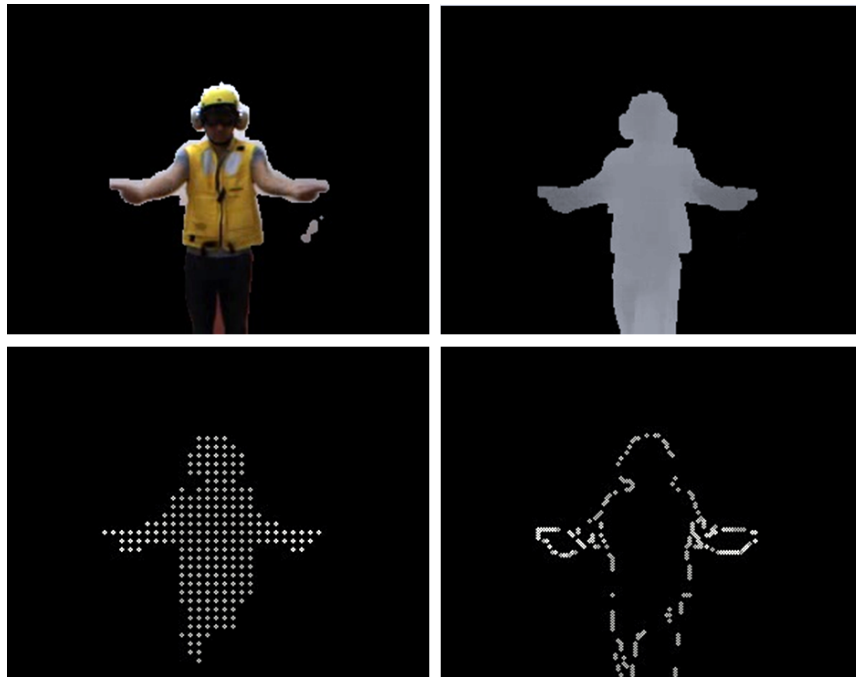


Figure 4-8: Input image feature extraction: (a-left top) color image, (b-right top) depth map, (c-left bottom) surface point cloud, and (d-right bottom) contour point cloud.

Motion History Image: Input Image

To compute a motion history image from input images, a background subtracted color image at each time t is converted to a gray color image. Two time-consecutive images are then used to compute a motion history image following the same method that we used for

²Closing (dilate and erode) and opening (erode and dilate) transformations are often used in connected component analysis: a closing transformation helps to remove unwanted elements caused by noise, while an opening transformation helps to connect nearby large regions.

the parametric model (Eq. 4.9).

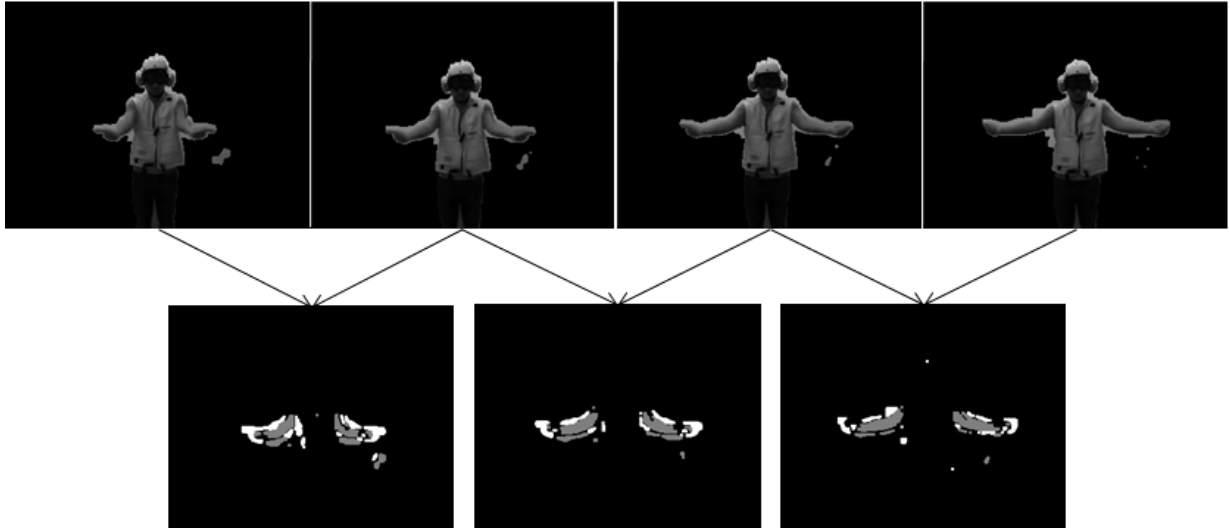


Figure 4-9: Motion history images (bottom row) extracted from pairs of input images (top row). At the bottom row, gray pixels indicate where the arms moved from, and white pixels indicate where the arms moved to.

4.3 Estimation Framework

The goal here is to estimate a vector with the 8 joint angle variables that define a body pose at each time t in a time sequence. Human upper body movements can be highly unpredictable, so an estimation framework that assumes that its random variables form a single Gaussian distribution can fall into a local minima or completely loose track. In this work, we solve the body pose estimation problem using the particle filter method [33], which assumes the underlying distribution to be multimodal and non-Gaussian. We briefly review the particle filter method, then explain how we apply the method to estimate upper body pose.

4.3.1 Particle Filter

The particle filter was introduced to the computer vision community by Isard *et al.* [33], named as the CONDENSATION (CONditional DENsity propaGATION) algorithm. Although originally introduced as an algorithm to detect and track the contour of objects moving in a cluttered scene, the algorithm has become one of the seminal works for generative model-based body pose estimation (e.g., [22, 43]).

The most prominent feature of the particle filter is that it models the state density $p(x_t)$ as a multimodal non-Gaussian distribution. One advantage of this approach is its ability to maintain multiple hypotheses during inference. When the underlying dynamic process governing the observation is highly unpredictable, this approach often leads to more robust estimation result as compared to single hypothesis filtering methods (e.g., the Kalman filter [72]).

We will denote an observation (the input image) at time t as z_t , and the state of model (a vector of 8 joint angle values in this work) at time t as x_t . Also, we will denote a history of observations z_t up to time t as Z_t , and a history of states x_t up to time t as X_t ; that is, $Z_t = \{z_1, \dots, z_t\}$ and $X_t = \{x_1, \dots, x_t\}$.

The algorithm assumes that the observations Z_t are mutually independent given the states of the model X_t . It also assumes that X_t form a first-order Markov chain. Thus, it follows:

$$\begin{aligned}
 p(x_t, Z_{t-1} | X_{t-1}) &= p(x_t | X_{t-1}) p(Z_{t-1} | X_{t-1}) \\
 &= p(x_t | x_{t-1}) \prod_{i=1}^{t-1} p(z_i | x_i)
 \end{aligned}
 \tag{4.10}$$

where $p(x_t | X_{t-1}) = p(x_t | x_{t-1})$ is derived by the Markov assumption, and $p(Z_{t-1} | X_{t-1}) = \prod_{i=1}^{t-1} p(z_i | x_i)$ is derived by the independence of observations assumption.

The algorithm also assumes that $p_t(x_t)$, the state density at time t , can be captured from

a history of observations up to that point:

$$p_t(x_t) \equiv p(x_t | Z_t). \quad (4.11)$$

With the above assumptions, inference on posterior state density $p(x_t | Z_t)$ over discrete time steps is modeled using the following probability density propagation rule:

$$p(x_t | Z_t) = k_t p(z_t | x_t) p(x_t | Z_{t-1}) \quad (4.12)$$

where

$$p(x_t | Z_{t-1}) = \int_{x_{t-1}} p(x_t | x_{t-1}) p(x_{t-1} | Z_{t-1}) \quad (4.13)$$

and k_t is a normalization constant that does not depend on x_t .

Note that the conditional state density $p(x_t | Z_t)$ update in Eq 4.12 is modeled as the likelihood of an observation given a state $p(z_t | x_t)$ and the effective prior $p(x_t | Z_{t-1})$, which can be interpreted as the reactive effect expected from an observation z_t . Also note that the derivation of an effective prior $p(x_t | Z_{t-1})$ in Eq. 4.14 is modeled as a prediction taken from the previous step's posterior $p(x_{t-1} | Z_{t-1})$, onto which is superimposed one time step forward following the underlying dynamical process $p(x_t | x_{t-1})$.

The problem now is how to compute the terms in Eq. 4.12 efficiently. When it is complex to model the likelihood $p(z_t | x_t)$ and the conditional state density $p(x_t | Z_t)$ cannot be solved in a closed form, iterative sampling techniques are often used for approximation. The CONDENSATION algorithm uses the *factored sampling* method to approximate $p(x_t | Z_t)$, and is iteratively applied to a temporal sequence of images, extending the original method to the temporal domain.

The conditional state density $p(x_t | Z_t)$ is approximated as a set of N sample-and-weight

combinations: $\{(s_t^{(1)}, \pi_t^{(1)}), \dots, (s_t^{(N)}, \pi_t^{(N)})\}$, where each weight is computed as

$$\pi_t^{(n)} = \frac{p(z_t | x_t = s_t^{(n)})}{\sum_{j=1}^N p(z_t | x_t = s_t^{(j)})}. \quad (4.14)$$

This set can be obtained from the prior density $p(x_t | Z_{t-1})$, which can be derived using the previous time-step's output $\{(s_{t-1}^{(1)}, \pi_{t-1}^{(1)}), \dots, (s_{t-1}^{(N)}, \pi_{t-1}^{(N)})\}$ that represents $p(x_{t-1} | Z_{t-1})$, to which we apply Eq. 4.11 and Eq. 4.12 using the assumed underlying dynamical process $p(x_t | x_{t-1})$.

Once the N samples are obtained, the estimated result can be calculated as the weighted mean of samples $s_t^{(n)}$:

$$\mathbb{E}[f(x_t)] = \sum_{n=1}^N \pi_t^{(n)} f(s_t^{(n)}). \quad (4.15)$$

4.3.2 Likelihood Function

In order to evaluate the weights $\pi_t^{(n)}$ in Eq. 4.14, we need to define a likelihood function $p(z_t | x_t = s_t^{(n)})$ that best captures the likelihood of an observation z_t given the n -th sample $s_t^{(n)}$. This is typically done by fitting an observation z_t to the parametric model generated with each of $s_t^{(n)}$. Section 4.2 described our methods for extracting image features and parametric model features, including the surface point cloud, the contour point cloud, and the motion history image. In this section, we describe how we compute a fitting error that compares the image features to the model features.

Let $\mathcal{P}_{surface}(\cdot)$ and $\mathcal{P}_{contour}(\cdot)$ be a surface point cloud and a contour point cloud. Each cloud is collected both from an observation z_t and from the parametric model generated with the n -th sample $s_t^{(n)}$. Also, let $I_{MHI}(z_t, z_{t-1})$ be a motion history image of the observation at time t , and $I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})])$ be a motion history image of the parametric model at time t , where $\mathbb{E}[f(x_{t-1})]$ is an estimation result from the previous time-step $t-1$.

The likelihood function is defined as an inverse of an exponentiated fitting error $\epsilon(z_t, s_t^{(n)})$:

$$p(z_t | x_t = s_t^{(n)}) = \frac{1}{\exp\{\epsilon(z_t, s_t^{(n)})\}} \quad (4.16)$$

where $\epsilon(z_t, s_t^{(n)})$ is defined as the sum of errors computed from comparing the three extracted features

$$\begin{aligned} \epsilon(z_t, s_t^{(n)}) &= \epsilon_{SSD_surface}(\mathcal{P}_{surface}(z_t), \mathcal{P}_{surface}(s_t^{(n)})) + \\ &\quad \epsilon_{SSD_contour}(\mathcal{P}_{contour}(z_t), \mathcal{P}_{contour}(s_t^{(n)})) + \\ &\quad \epsilon_{MHI}(I_{MHI}(z_t, z_{t-1}), I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})])). \end{aligned} \quad (4.17)$$

Sum of Squared Distance Error

The first and second terms in Eq. 4.17 is based on a generalized function of the sum of squared distance error, defined as:

$$\epsilon_{SSD}(\mathcal{P}(z), \mathcal{P}(s^{(n)})) = \sum_{p_z \in \mathcal{P}(z)} \{d(p_z, \min(p_{s^{(n)}}))\}^2, \quad p_z \in \mathcal{P}(z) \quad \text{and} \quad p_{s^{(n)}} \in \mathcal{P}(s^{(n)}) \quad (4.18)$$

where $\mathcal{P}(z)$ is a point cloud from the observation and $\mathcal{P}(s^{(n)})$ is a point cloud from the parametric model that is generated with the n -th sample $s^{(n)}$. The function $d(p_1, \min(p_2))$ calculates the Euclidean distance between two points p_1 and p_2 , where the second point p_2 is selected to be the closest point to the first point p_1 in the Euclidean space. Eq. 4.18 is used to calculate the sum of squared distance error $\epsilon_{SSD}(\mathcal{P}_z, \mathcal{P}_{s^{(n)}})$ for the surface point cloud and for the contour point cloud.

Motion History Image Error

The third term in Eq. 4.17 is an error function for the motion history image, which is calculated as follows. We first generate two motion history images: a motion history image

of observation $I_{MHI}(z_t, z_{t-1})$ using two time-consecutive images z_t and z_{t-1} (Section 4.2.2), and a motion history image of model $I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})])$ using the n -th sample $s_t^{(n)}$ and an estimated result from the previous time-step $\mathbb{E}[f(x_{t-1})]$ (Section 4.2.1). With the two motion history images, an error function for the motion history image is defined as:

$$\begin{aligned} \epsilon_{MHI}(I_{MHI}(z_t, z_{t-1}), I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})])) = \\ \mathbb{C} \left[\mathit{thresh}\{ \mathit{abs}(I_{MHI}(z_t, z_{t-1}) - I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})])) \}, 128, 255 \} \right] \end{aligned} \quad (4.19)$$

This error function first subtracts $I_{MHI}(z_t, z_{t-1})$ from $I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})])$ and computes an absolute-valued image of it. Then it applies the binary threshold operator with the cutoff value set to 128 and result value set to 255 (i.e., set a pixel value to 255 if the original value of that pixel was greater than or equal to 128; set to 0 otherwise), and counts non-zero-valued pixels with an operator \mathbb{C} .

The intuition behind setting the cutoff value to 128 can be described as follows. Table 4.2 shows all possible cases of the absolute-valued image. Remember that for each motion history image (the first two columns), the pixel value 0 means there has been no change since $t-1$, 127 means there was an object at $t-1$ but has moved at t , and 255 means there was no object at $t-1$ but has appeared at t in that pixel. By subtracting these values, we get four possible results (the third column in Table 4.2):

- the pixel value 0 means the conditions in two motion history images are the same;
- the pixel value 127 means one of the motion history images indicates there was an object at $t-1$ while the other indicates there has been no object since $t-1$;
- the pixel value 128 means one of the motion history images indicates there was an object at $t-1$ while the other indicates there is an object at t ;
- the pixel value 255 means one of the motion history images indicates there has been no object at $t-1$ while the other indicates there is an object at t .

We want to penalize the conditions in which two motion history images do not match at

$I_{MHI}(z_t, z_{t-1})$	$I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})])$	$abs(I_{MHI}(z_t, z_{t-1}) - I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})]))$
0	0	0
0	127	127
0	255	255
127	0	127
127	127	0
127	255	128
255	0	255
255	127	128
255	255	0

Table 4.2: Possible conditions for computing $\epsilon_{MHI}(I_{MHI}(z_t, z_{t-1}), I_{MHI}(s_t^{(n)}, \mathbb{E}[f(x_{t-1})]))$. Note that, for the first two columns, the value 0 means there has been no object in the pixel, the value 127 means there was an object in the pixel but it has moved, and the value 255 means there is an object. Therefore, by thresholding the absolute subtracted values with the cutoff value 128, we can ignore the mistakes happened at $t - 1$ and concentrate on the mistakes that happened at time t .

the current time t , independent of the situation at $t - 1$. Therefore, by setting the cutoff value to 128, we can ignore the errors in the previous time-step and concentrate on the errors that are related to only the current time-step (see Table 4.2).

4.3.3 Initialization

Iterative methods needs a good initialization of the prior density $p(x_0 | z_0)$ for successful estimation. For this reason, we need initial values of the model parameters to be as accurate as possible (i.e., joint locations, limb lengths, and joint angles). The initial values are usually obtained assuming a person in the image is making a predefined body pose; then the problem becomes locating joints and finding limb lengths that best describe the observation.

In this work, as seen in Fig. 4-10, we assume that a person in the scene is making a T-shape pose. The initial parameter values are obtained by first locating head and wrist joints in the image (by scanning a background subtracted image from top-to-bottom, left-to-right,



Figure 4-10: Participants were asked to make a T-shape pose for initialization.

and right-to-left) then locating the rest of the joints based on these three joints; limb lengths can be obtained from the joint locations; joint angles are already known because we assume a specific body pose.

Note that, obtaining the location of chest joint as a geometric mean of left and right wrist joints can result in inaccurate initial parameter values, because a person can make an imperfect T-shape pose (thus resulting in a situation where the y -coordinates of the chest joint and both hands do not lie on the same row in the image).

We locate the chest joint by performing a grid search varying the (x, y) coordinate of chest joint and the Z-axis rotation angles of shoulder joints (which determines the up-and-down movement of an arm). This often gives more accurate initialization parameter values (see Figure 4-11). After the chest joint is located, limb lengths are calculated using the relative ratios of limbs published in an anthropometric data [68].

4.3.4 Estimation Under Constraints

There exists certain body poses that are anatomically implausible, so this should be reflected in the body pose estimation as well. That is, we should put some constraints on



Figure 4-11: Initialization results collected from 20 subjects. Although each subject has made a T-pose for the initialization, the shoulder angles were not always 90 degree to the backbone, preventing us to assume that the chest point is always at the middle of two wrist points.

Left Shoulder	Pitch	$180 \leq \theta_{Pitch}^{LS} \leq 360$
	Yaw	$200 \leq \theta_{Yaw}^{LS} \leq 330$
	Roll	$30 \leq \theta_{Roll}^{LS} \leq 250$
Left Elbow	Flexion	$210 \leq \theta_{Flexion}^{LE} \leq 360$
Right Shoulder	Pitch	$180 \leq \theta_{Pitch}^{RS} \leq 360$
	Yaw	$210 \leq \theta_{Yaw}^{RS} \leq 340$
	Roll	$30 \leq \theta_{Roll}^{RS} \leq 250$
Right Elbow	Flexion	$210 \leq \theta_{Flexion}^{RE} \leq 360$

Table 4.3: Joint angle range constraints.

the ranges of joint angles. There are two reasons for this: (a) estimated body poses should make only anatomically plausible body poses, and (b) in order to optimize the use of computational resources, the estimation framework should not be searching in the implausible parameter vector space. We set possible joint angle ranges based on an anthropometric data chart published by NASA [52], empirically modifying the ranges to add some flexibility (see Table 4.3).

4.4 Experiment

In any system we need to consider the trade-off between accuracy and computational speed. For example, if estimation results are used for controlling a 3D avatar, as in computer animation film making, highly accurate body poses are needed that match the input at the pixel displacement level. However, if estimation results are used for other higher-level tasks, such as the gesture recognition in this work, it is sufficient to show how well the estimated body poses resembles the input, visually comparing estimation results to the input.

For completeness, we performed both a *qualitative* and a *quantitative* analysis: our qualitative analysis visually checked whether estimated body poses resembled the actual human body poses (Section 4.4.1); our quantitative analysis measured pixel displacement errors between the estimated body pose and the ground-truth body pose data collected from the

Vicon motion capture system (Section 4.4.2).

We selected 10 gestures out of 24 that are good representative examples of aircraft handling signals (more discussion on this appears in Section 6.4.1). When collecting the dataset, each of 20 participants repeated each gesture 20 times, hence there were a total of 400 trials for each gesture. All gestures were recorded at 20 FPS; each gesture lasted about 3 seconds. Using 500 particles, the body pose estimation took about 0.7 seconds for each frame, on an Intel Xeon Dual Core 2.66GHz machine with 3.25GB of RAM.

4.4.1 Qualitative Analysis

Methods

A qualitative analysis was performed using a data visualization tool we created that displays various information on the process of body pose estimation (see Figure 4-12). Using the tool, a human evaluator visually checked the results frame-by-frame to see if the estimated body poses resembled the actual body poses shown in the recorded images, counting the number of erroneous estimation results.

Note that visually checking each of the 10 gestures means that we would have to look at approximately 240,000 image frames (roughly 24,000 for each gesture). In order to make this analysis tractable, we randomly selected for each gesture one trial sample out of 20, and checked only that trial's sequence of images. Also, we checked only even-numbered frames in each trial: this was a reasonable approximation since two consecutive frames tend to show similar results.

To set this in context, we need to clarify the evaluation criteria, that is, how close does an estimated body pose has to be to the actual body pose? Without a clear definition of such a criteria, the result might fall into the *subjective bias*³. Our answer is that an estimated

³Subjective bias is decision making or evaluation based on personal, poorly measurable, and unverifiable data or feelings that are improperly weighted against objective, unbiased data [23]

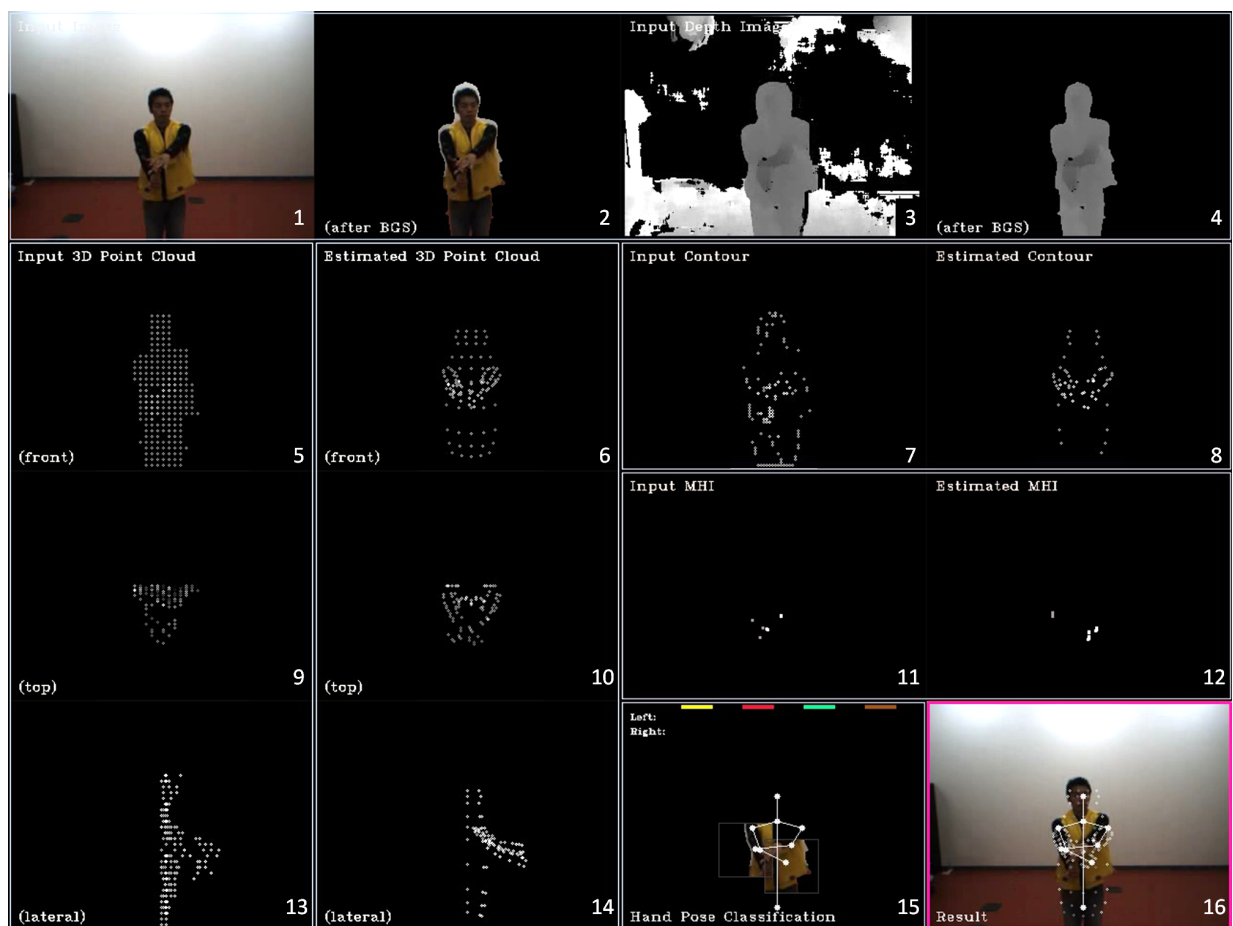


Figure 4-12: A data visualization tool. (1,3): original input image and depth map, (2,4) input image and depth map after background subtraction, (5,9,13): 3D surface point cloud of input image from three different viewpoints (front, top, lateral), (6,10,14): 3D surface point cloud of estimation result from three different viewpoints (front, top, lateral), (7,8): 3D contour point cloud of input image and estimation result, (11,12): motion history image of observation and estimation result, (15): hand pose estimation result (Chapter 5), (16): synthesized view of body and hand pose estimation

Gesture Index	Gesture Description	Error Rate
#2	Affirmative	7.82%
#3	Negative	6.83%
#4	Spread wings	12.83%
#5	Fold wings	6.80%
#10	Remove chocks	3.08%
#11	Insert chocks	1.50%
#18	Engage nosegear steering	7.73%
#19	Hot brakes	5.83%
#20	Brakes on	13.96%
#21	Brakes off	10.77%

Table 4.4: A qualitative analysis result.

body pose will be regarded as an error if it looks obviously different from the actual body pose (i.e., an arm pointed to a position that is 45 degree off from the original one) or is an unreasonable body pose (i.e., an arm twisted abnormally). However, disagreements due to small noise (i.e., an arm slightly shaking frame to frame) or near-misses (i.e., the estimated arm position does not align perfectly to its original position) will not be counted as an error.

Result

Table 4.4 shows error rates for body poses in each gesture sequence. The average accuracy rate of the body pose estimation was 92.285%. Note that there is no benchmark result for comparison, so it is hard to judge how good or bad the result is. However, considering the fact that 6 of 10 gestures included 3D body poses (gesture #4, #5, #10, #11, #18, #19), we believe that the reliability of the pose estimator is at a reasonable level to be used for gesture recognition, which will in turn be shown in Chapter 6.

Discussion

In general, more errors occurred on the gestures that included self-occluded body poses. Gesture #4 (spread wings), for example, contained body poses where both arms were located in front of the body at a close distance: it started with making a shrugging gesture, with both arms kept close to the body and moving to the chest (as seen in Figure 4-13). Most errors on this gesture occurred during tracking the shrugging part of the gesture. This seems to be caused by the way the system estimates self-occluded body poses. Note that two of the features we extracted (a surface point cloud and a contour point cloud) were highly related to the depth information. Therefore, we can expect that the estimation accuracy may be degraded if the depth of an arm does not differ noticeably from the rest of the body (as in the case of shrugging gesture).

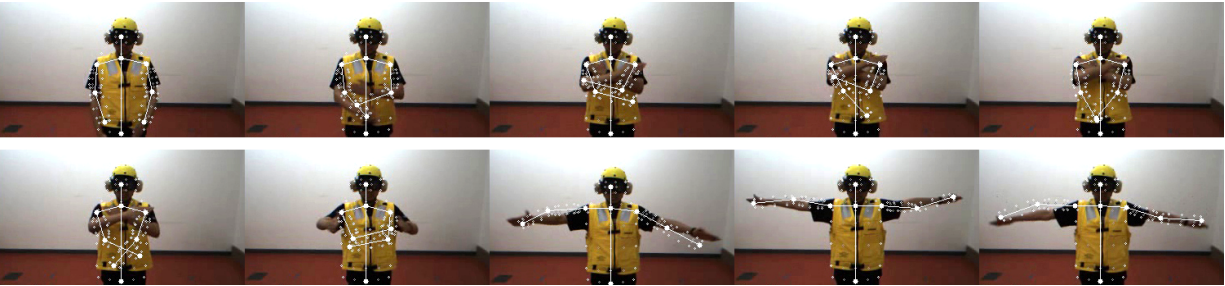


Figure 4-13: Gesture #4 (spread wings) contains a shrugging gesture, which causes a major portion of estimation errors for this gesture. The pose estimator tracks the shrugging gesture correctly for a few frames at the beginning, but fails when arms get too close to the body. Note that it quickly finds the correct pose when there is no more self-occlusion.

Also, error rates of gesture #20 (brakes on) and #21 (brakes off) were, although still low, relatively higher than the others. Note that, as depicted in Figure 4-14, these two gestures included raising both arms outwards (making the T-pose) and bending both elbows so that both hands point upward. A majority of errors on these two gestures were that the estimated bending direction of elbows were opposite to the actual one. This type of errors can be explained by the Monte Carlo stochastic process of the particle filter, and the recovery process with a dynamical process of the model and range limits on joint angles.

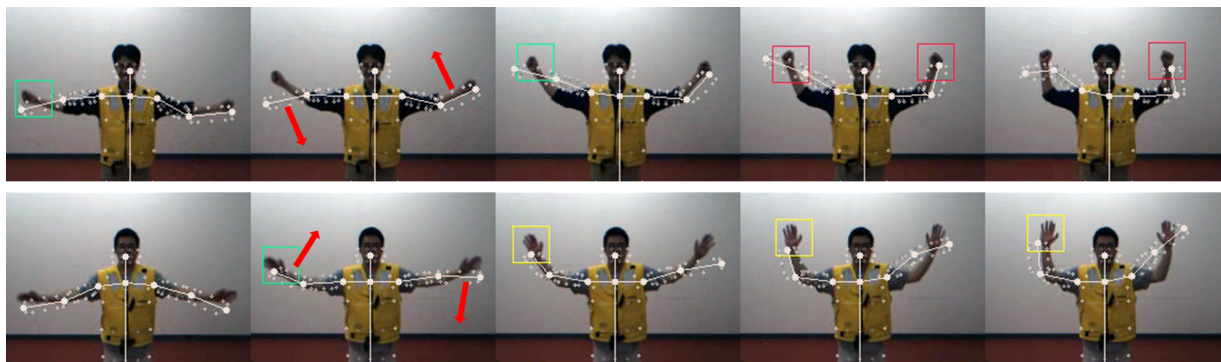


Figure 4-14: Estimation fails when the bending direction of an elbow (red arrows) is incorrect. (Rectangles around hands are hand pose estimation result, which will be discussed in the next chapter).

To build an intuition of what is causing this error: imagine an actor standing at attention, wearing a big jacket with his hands hidden inside it. Your task is to guess in which direction the actor's elbow can bend. The actor slowly raises an arm, keeping the elbow straight; at this point, you have no idea where the elbow can bend, so you just have to guess the bending direction. After the actor starts bending his elbow in a certain direction, you start to have a better idea, either confirming that your guess was right or overriding the erroneous knowledge.

Note that this is very similar to how our body pose estimation is performed, except that for body pose estimation the recovery process is constrained with a dynamical process of the model and range limits on the joint angles.

- A dynamical process of the model determines how fast each of the 8 joint angle values can change⁴. In our recorded videos, joint angles tended to change in a small value in each frame, because the videos were captured at 20 FPS. Therefore, the dynamical process was configured so that their permitted changes are small at each frame. However, to change an elbow's bending direction to the opposite, one of the

⁴Remember that, each time a particle filter framework goes through the diffusion step, we apply a dynamical process (essentially a vector of Gaussian random values) to each particle, changing the values of the 8 joint angle variables. Therefore, a configuration of the dynamical process (the mean and variance of the Gaussian distribution) affects the speed of the joint angle change: the bigger the mean is, the faster the permitted changes are.

three shoulder angles should turn 180 degrees; this may not be possible given the dynamical process of the model.

- The range limits on the joint angles prevent a joint from bending in an abnormal direction, i.e., an elbow bending outwards. What this means is that, although the skeletal model can bend an elbow to the negative direction, the range limit prevents this from being an acceptable solution.

The errors in Figure 4-14 happened when most of the particles had elbow joints set to bend downwards, that is, when there were not enough particles that caused the elbows to bend upward. To recover from this, erroneous particles should be able to change their joint angle values rapidly and appropriately. However, due to the reasons described above, although in general they make estimation reliable, the dynamical process and the range limits prevented the errors to be recovered appropriately.

4.4.2 Quantitative Analysis

Method

A quantitative analysis was performed by measuring pixel displacement errors between joints of estimated body poses and joints of the ground-truth data, collected using the Vicon motion capture system⁵ [70].

The Vicon system records, using multiple cameras, the movement of several reflectors attached to predefined positions on the body, with each camera emitting infrared light and capturing the light reflected back from the reflectors. Using the recorded movements of each reflector, the system solves the inverse kinematics problem to reconstruct a skeletal model, and outputs trajectories of 8 joint points (chest, navel, left/right shoulder, left/right elbow, left/right wrist).

⁵We used the Vicon motion capture system from the MIT CSAIL holodeck room (16 cameras, 120 Hz frequency, 1 mm precision).

To get the ground-truth data, one participant was selected and recorded using both a stereo camera and the Vicon system simultaneously. From the Vicon captured images, trajectories of 8 joint points (chest, navel, left/right shoulder, left/right elbow, left/right wrist) were obtained using a manufacture provided toolkit [70]. The obtained trajectories were superimposed onto images captured from the stereo camera, scaled and translated properly so that they align with the coordinate system that the estimated body pose is in. Finally, 2D pixel displacement errors were calculated for each joint on a 320x240 pixel frame.

Result and Discussion

Figures 4-15, 4-16, 4-17, 4-18, 4-19 show graphs of joint displacement errors measured in pixels, paired with similar gestures. In each graph, joint displacement errors are shown with each line (blue: left elbow, green: right elbow, yellow: left wrist, purple: right wrist), indicating how much fluctuation of displacement errors happened for the joint. The X-axis indicates frame index numbers, which shows the 20 repetitions of the same gesture over a period of time. Also, Table 4.5 shows mean and standard deviation pixel values of joint displacement errors. Note that, both in the figures and the table, we show elbow and wrist joint errors only, since the position of a body trunk is fixed once initialized and the position of the shoulder is analytically determined.

There are two major findings from the result of this analysis. First, as discussed in Section 4.4.1, gestures involving 2D body poses only (#2, #3, #20, and #21) showed more reliable estimation result than others (#4, #5, #10, #11, #18, and #19). Second, although some graphs showed higher error rates than others, we can see that estimation was performed consistently in the sense that errors mostly occurred at the same phase in a gesture. To see this, following our discussion in Section 4.4.1, let's take the graph of gesture #4 (spread wings) for example (shown in Figure 4-16). The high error peaks of WristL (yellow line) and Wrist R (purple line) occurred when a person was making the shrugging part of the gesture. Although the errors were high on average, the fluctuation

of estimation error repeated 20 times as the time goes by (following the 20 repetitions of the same gesture), which explains that the estimation performed consistently without too much variation.

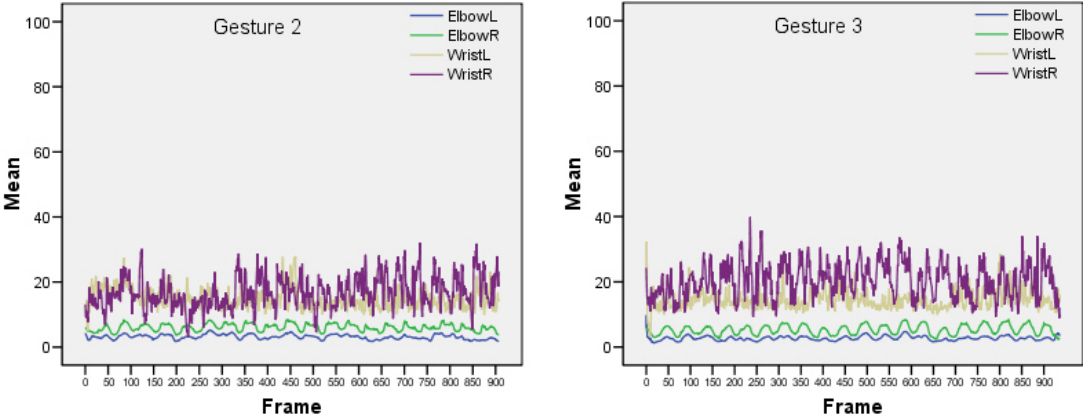


Figure 4-15: Pixel distance errors of body pose in gesture #2 (affirmative) and #3 (negative).

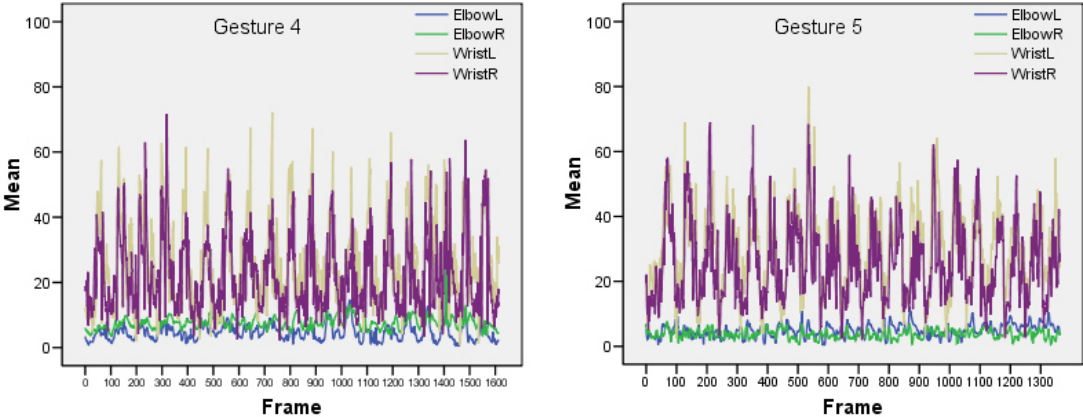


Figure 4-16: Pixel distance errors of body pose in gesture #4 (spread wings) and #5 (fold wings).

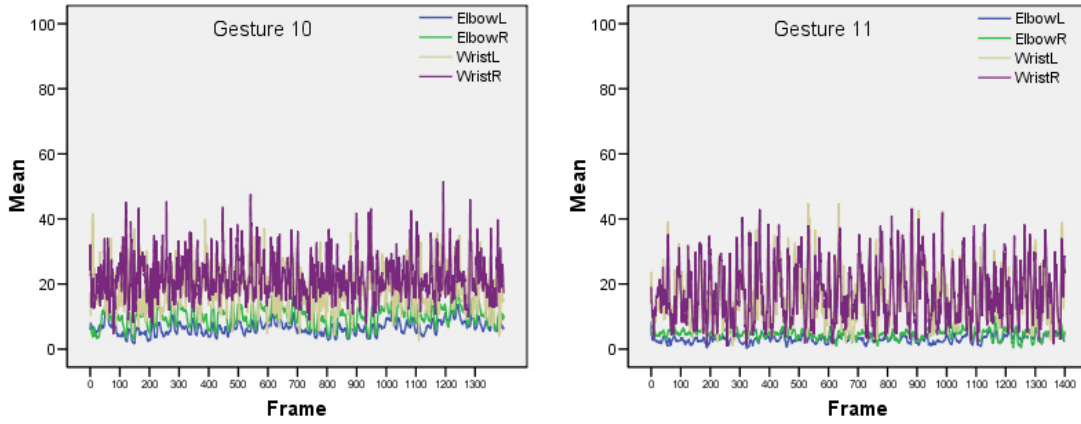


Figure 4-17: Pixel distance errors of body pose in gesture #10 (remove chocks) and #11 (insert chocks).

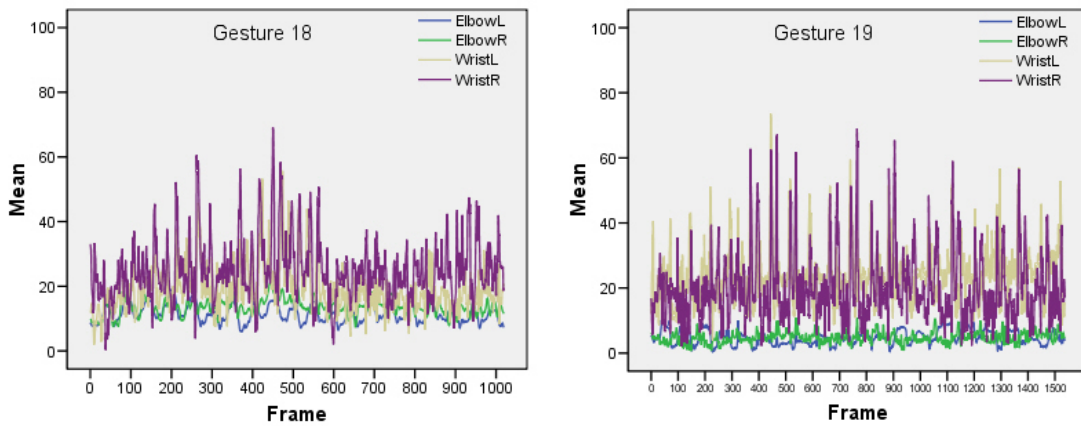


Figure 4-18: Pixel distance errors of body pose in gesture #18 (engage nosegear steering) and #19 (hot brakes).

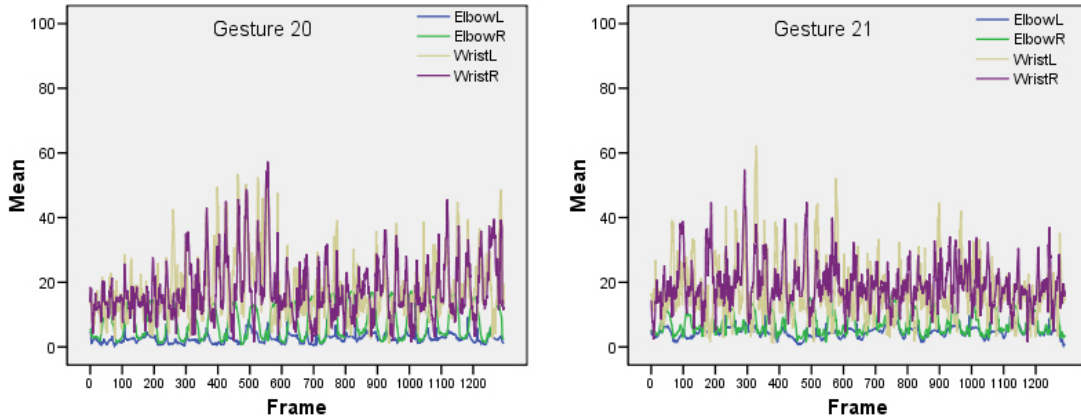


Figure 4-19: Pixel distance errors of body pose in gesture #20 (brakes on) and #21 (brakes off).

4.5 Related Work

The problem of body or hand pose estimation has been an active research area (see the review articles [25][49][50][58]), and many approaches were proposed, which can be roughly categorized into a *model-free* approach and a *model-based* approach.

4.5.1 Model-Free Approaches

The model-free approach establishes a direct correlation between an observed image and a pose. Two broad classes of work in this approach are a *learning-based* method and a *example-based* method. The learning-based method builds a statistical model that learns a mapping function from image space to pose space [5][29][67], where the example-based method performs a similarity search in a collected pose database [12][51][62].

Problems in the model-free approach include the large numbers of possible poses, and the viewpoint sensitivity, i.e., the same pose can look different when viewed from different viewpoints. Therefore, the model-free approach requires a large training dataset that contains various poses from many camera viewpoints, which makes it pragmatically impos-

Gesture Index	Gesture Description	Joint	Mean (px)	Std. Dev (px)
#2	Affirmative	ElbowL	3.15	.72
		ElbowR	6.02	1.10
		WristL	14.89	3.09
		WristR	16.73	4.89
#3	Negative	ElbowL	2.71	.71
		ElbowR	5.30	1.45
		WristL	14.32	2.80
		WristR	20.36	5.41
#4	Spread wings	ElbowL	4.61	2.35
		ElbowR	7.58	1.86
		WristL	25.61	13.67
		WristR	22.51	11.62
#5	Fold wings	ElbowL	4.89	1.91
		ElbowR	3.75	1.35
		WristL	30.33	12.47
		WristR	25.59	12.26
#10	Remove chocks	ElbowL	6.60	2.07
		ElbowR	10.28	3.02
		WristL	19.15	6.19
		WristR	20.02	6.33
#11	Insert chocks	ElbowL	3.19	1.17
		ElbowR	4.65	1.38
		WristL	15.89	7.76
		WristR	17.28	8.86
#18	Engage nosegear steering	ElbowL	11.64	2.71
		ElbowR	13.60	2.61
		WristL	18.99	7.16
		WristR	24.84	9.51
#19	Hot brakes	ElbowL	4.78	2.13
		ElbowR	4.81	1.69
		WristL	24.56	8.79
		WristR	20.30	10.89
#20	Brakes on	ElbowL	2.56	1.34
		ElbowR	7.91	5.08
		WristL	17.88	9.39
		WristR	17.41	9.01
#21	Brakes off	ElbowL	4.61	1.61
		ElbowR	7.58	3.51
		WristL	16.96	8.71
		WristR	19.19	6.98

Table 4.5: Joint displacement errors.

sible to collect and label all the possible pose images. In addition, this approach generally estimates 2D poses only.

4.5.2 Model-Based Approaches

The model-based approach, on the other hand, used for body pose estimation in this work, constructs a parametric model of the human body from the observed image, by evaluating the parametric model's likelihood given the observed image [20][22][40][64]. This approach is not affected by a camera viewpoint, does not require a training dataset, and is generally more robust in pose estimation. Plus, 3D acquisition of body pose is possible. Although this approach requires a good initialization of model parameters for satisfactory estimation, several methods for automatic initialization methods have been proposed [20][37].

Chapter 5

Hand Pose Classification

In this chapter we describe hand pose classification, the third part in the gesture recognition pipeline. The task here is to find and classify canonical hand poses made contemporaneously with gestures. To this end, we first defined a vocabulary of canonical hand poses used in NATOPS aircraft handling signals (Section. 5.2). The training data set was collected by manually segmenting images of hands, and extracting the Histograms of Oriented Gradients (HOG) features [19] (Section 5.3). Then a multi-class Support Vector Machine (SVM) classifier [69] was trained using the data set (Section 5.4). Note that exhaustive search for hands in the image is not necessary because the computed body pose supplies information on possible wrist positions; a small search region was defined around each of the estimated wrist positions, and hand tracking was performed only on those regions, dramatically reducing the amount of computation needed (Section 5.4.3).

When tested on segmented images that included positive and negative examples of all hand poses, the multi-class SVM hand pose classifier accomplished near-perfect recognition accuracy (99.94%) (Section 5.5). The chapter concludes with a review of some other approaches to this task (Section. 5.6).

5.1 Goal and Assumptions

The goal here is to classify canonical hand poses by distinguishing their *appearances*, i.e., how they look, not by building a *model* of them, i.e., reconstruct the anatomical structure, as was done for the body pose estimation in Chapter 4.

The appearance of hand poses depends heavily on the viewpoint. In our task, however, a person is assumed to be facing the camera, so the viewing angle is also assumed to be relatively static. Note that this resembles the realistic scenario: on the carrier flight deck, personnel are required to make an eye contact with pilots when they communicate, and if the vehicle moved so far that it is hard for a pilot to see the person directly, another person gets the control of the vehicle and continues to gesture to the pilot, again making direct eye contact when communicating. Therefore, it seems reasonable to assume that the viewing angle of hand images collected during gesture recognition will not vary significantly, in which case the appearance-based approach can work reliably¹.

Another assumption is that hand poses in aircraft handling signals tend to be coarse and can be summarized with a handful of canonical poses. Note that this is inevitable on the carrier flight deck because of the standing distance between pilots and personnel: on the carrier flight deck, personnel usually keep back 50 feet (15 meters) from the aircrafts for safety. This makes it hard for pilots to recognize hand poses accurately if there are many similar ones, which implies there cannot be many similar hand poses and we can define a handful of canonical hand poses for NATOPS gesture recognition. There are, in fact, only seven canonical hand poses in NATOPS standard aircraft handling signals that have specific meanings and play a key role defining the meaning of a gesture (thumb up/down/left/right, palm open/close, and pointing); this is not true for body poses, which

¹Taking an appearance-based approach for hand pose classification may sound contradictory to some readers: in body pose estimation we took a model-based approach although the same assumption still holds. The main reason that we took model-based approach for body pose estimation was to be able to reconstruct body pose in 3D space, and 3D information is needed to classify the NATOPS gestures reliably [53]. An appearance-based approach for body pose estimation has been mainly used for 2D body pose estimation, such as in [12]. However, when it comes to 3D body pose estimation, model-based approaches are more used, as discussed in Section 4.5.

can be seen accurately from further away, and hence use a larger vocabulary.

5.2 Hand Pose Vocabulary

As discussed in the previous section, there are seven distinct hand poses used in the 24 gestures studied in this work. Of these, we selected four (thumb up/down, palm opened/closed) (Figure 5-1), that we believe represent canonical hand poses that capture important visual cues during gesticulation. These hand poses play important roles in defining meanings of the gestures: “Affirmative” (thumb up), “Negative” (thumb down), “Brakes on/off” (palm opened/closed) (Figure 5-1).



Figure 5-1: Canonical hand poses used in NATOPS standard aircraft handling signals. The four hand poses used in this work are shown in red boxes.

The three other hand poses (thumb left/right and pointing) were excluded from this work for a variety of reasons. For the thumb left/right hand poses, in most cases they were in front of the body, so the sampled images were cluttered with other textures, such as jerseys. Therefore, we would need additional background removal routines to acquire hand images, i.e., perform skin color detection as was done in [13]. For the pointing hand pose, the orientation of the pointing hand can vary in 360 degrees, and we would need to train a hand pose classifier with much more data representing such a huge rotational variation. However, we speculate that extending the framework to classify these two hand poses can be done, and we leave this for future work.

5.3 Dataset

5.3.1 Data Collection

A hand pose dataset was collected from the recorded video data of the first 10 participants (out of 20). Positive examples were sampled by manually cropping 32 x 32 pixel images and labeling them, and negative examples were sampled by choosing two random locations and cropping the same sized images. Note that the positive examples were sampled before sampling the negative examples to ensure that the negative examples do not contain any hand images. Also, when selecting locations for the negative examples, the foreground-background mask images were used to ensure that no background images are included in the samples, which will not be searched during the tracking in any case.

After collecting the sample images, affine transformations were applied to the positive examples to scale and rotate the original samples into poses with a wide range of additional orientations and sizes. This was done to make the classifier more robust to scaling and rotational variations, and to increase and balance the number of positive sample sets across hand pose classes. After applying the transformations, the size of positive dataset for each hand pose class was well-balanced (about 12,000 samples per class) (Figure 5-2).

5.3.2 Feature Extraction: Histogram of Oriented Gradients

Now we turn our attention to the specific mechanism to compute image features. The Histograms of Orient Gradients (HOG) method was introduced to the computer vision community as an image feature descriptor for pedestrian detection, and since then it has been one of the most popular methods to extract image features, as it usually outperforms other approaches [19].

The HOG features are image descriptors based on dense and overlapping encoding of image regions. The central assumption of the method is that the appearance of an object

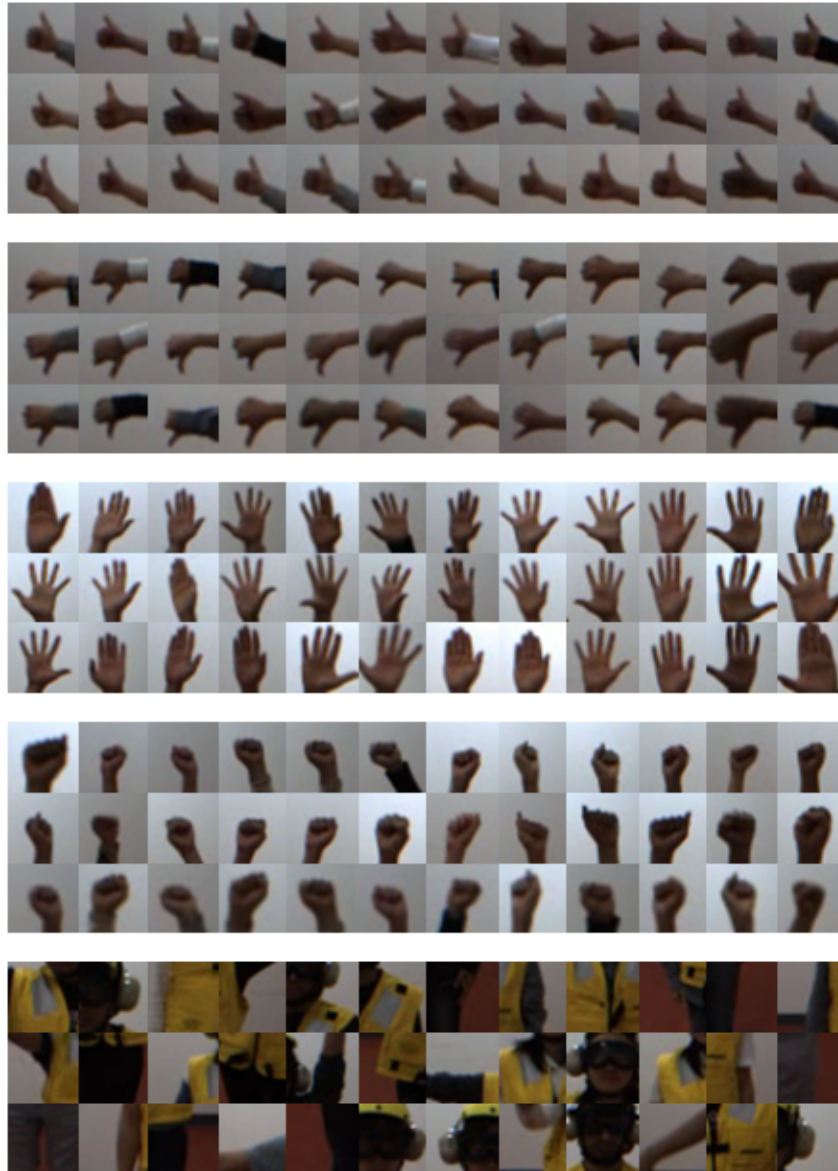


Figure 5-2: Hand pose dataset. Positive examples (first four rows) were sampled and labeled manually by cropping 32 x 32 images, and applying affine transformations to scale and rotate them. Negative examples (bottom row) were sampled randomly after positive examples were sampled in each image.

is rather well characterized by locally collected distributions of intensity gradients or edge orientations, even without having the knowledge about the corresponding gradient or edge positions that are globally collected over the image.

To compute the HOG features, an image window is divided into a grid of small regions, hereinafter "*cells*", and each cell's gradient orientations are quantized into a number of bins to create a histogram. To make the features more invariant to illumination and shadowing effects, the same image window is again divided into a grid of larger regions, hereinafter "*blocks*", and for each block the computed histograms are accumulated so that each cell within the block can be normalized with it. The histograms over the normalized blocks are referred to the HOG features.

In this work, we assume that each of the selected hand poses has a unique pattern of intensity gradients or edge orientations that distinguishes it from the others. This can be expected from the appearance of each hand pose. For example, an open palm has distinctive local appearances that look different from other hand poses, e.g., edges along the fingers. Since our vocabulary of hand poses has unique patterns of gradient orientations, we decided to use the HOG feature as an image descriptor.

5.4 Classification Framework

5.4.1 Multi-class Support Vector Machine Classifier

To classify the HOG features, we trained a multi-class Support Vector Machine (SVM) classifier [69] using an existing library (LIBSVM [16]). SVM is a discriminative framework for performing classification and regression tasks that can handle multiple categorical and continuous variables, by constructing hyperplanes that separate regions of different class

labels in a multi-dimensional space. It solves the following optimization function:

$$\min \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi_i \quad (5.1)$$

subject to the constraints:

$$y_i (\mathbf{w}^T \cdot \phi(x_i) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for } i = 1, \dots, N \quad (5.2)$$

where \mathbf{w} is a parameter vector, C is a penalty trade-off variable, N is the number of training examples, ξ_i are slack variables, $\phi()$ is a kernel function for transforming input data to the feature space, and b is an offset value. Because the image features extracted using HOG descriptors are high-dimensional, the parameter vector space might not be linearly separable. Therefore, to make the classification more robust, we use the Radial Basis Function (RBF) kernel function, defined as

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\gamma}{2} \|\mathbf{x} - \mathbf{x}'\|\right), \gamma > 0. \quad (5.3)$$

The multi-class SVM was trained following *one-against-one* method [35] for fast training, while obtaining comparable accuracy to *one-against-all* method [30]. The one-against-one method constructs a multi-class classifier based on a combination of $k(k-1)/2$ binary classifiers, where each classifier is trained on data from two classes.

Each time a sliding window moves to a new position within a search region, the HOG features are computed to form an input feature vector. Then the trained SVM classifier examines the input feature vector and returns the result as a vector of $k+1$ probability estimates for k hand classes and one negative class (i.e., no hand pose). If the highest probability estimate is from one of the k hand poses, we store the window's position as a classified hand pose position.

5.4.2 SVM Parameter Selection Using Grid Search

The multi-class SVM classifier requires us to perform parameter selection of two variables, C and γ , which often affect the performance of a trained engine significantly. In this work, we use a grid search approach to perform parameter selection following [16]. It performs training and testing repeatedly, changing values of C and γ at uniform intervals. We varied both C and γ from -5 to 5, increasing the values in each step by 1 (all in the log scale).

Figure 5-3 illustrates the result of the grid search. The optimal parameter values of C and γ are searched over the grids of (C, γ) values in log scale, x-axis showing $\log(C)$ and y-axis showing $\log(\gamma)$. In the figure, lines with the same colors indicate the parameter settings that lead to the same accuracy performance. The optimal parameter values were $C=4.0$ and $\gamma=0.03125$, which gave the overall accuracy 99.86%.

5.4.3 Search Region

Given an input image, one way to find the hands is to perform an exhaustive search: slide a small search window over the whole image area; compute features every time the window moves to a new position; then report back the positions where the hands were classified. However, this approach wastes a substantial amount of time computing features from meaningless regions, i.e., the background or front body regions. Obviously, an efficient approach would be to compute features only in the regions most likely to contain the hands.

In this work, we use the information of estimated wrist positions (computed in body pose estimation) to narrow down search regions. Around each of the estimated wrist positions, we create a small search region, slightly larger than the size of the actual hand image. Then the HOG features are computed from the search region, from top-left to bottom-right, by sliding a window with a fixed size interval. This process allows us to reduce the computation time dramatically. The specific size settings used in this work are 56 x 56

Best $\log_2(C) = 2.0$ $\log_2(\gamma) = -5$ accuracy = 99.8656%
 $C = 4.0$ $\gamma = 0.03125$

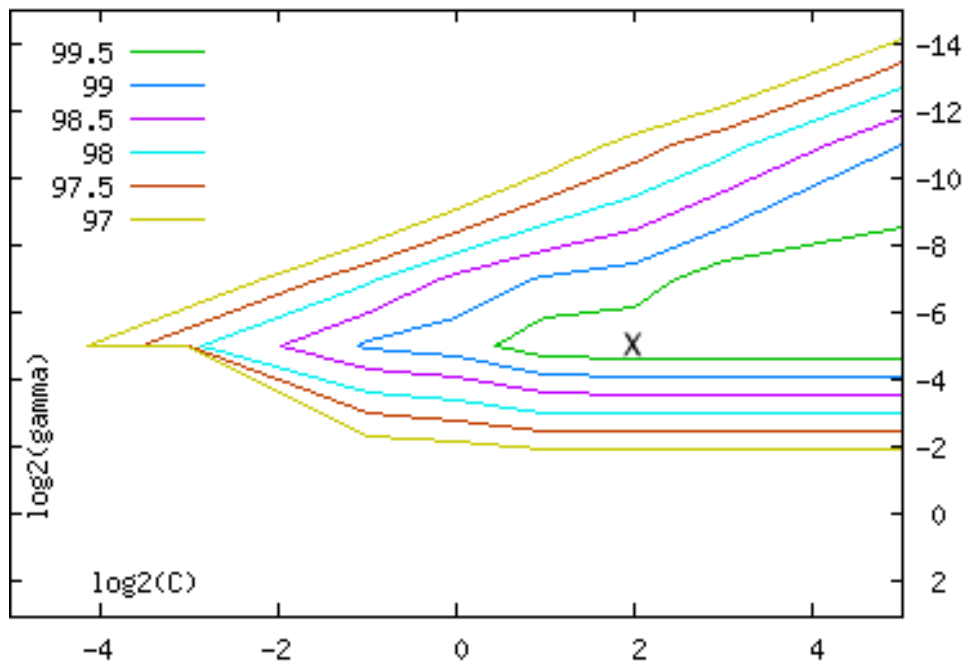


Figure 5-3: A graph showing the result of grid search over the two parameters C and γ . The optimal values of parameters (C, γ) are searched over the grids of (C, γ) values; x-axis shows $\log(C)$ and y-axis shows $\log(\gamma)$. Lines with the same colors indicate the parameter settings that lead to the same accuracy performance. The grid search result shows the optimal parameter values were $C=4.0$ and $\gamma=0.03125$, which gave the overall accuracy 99.86% (the X mark on the graph).

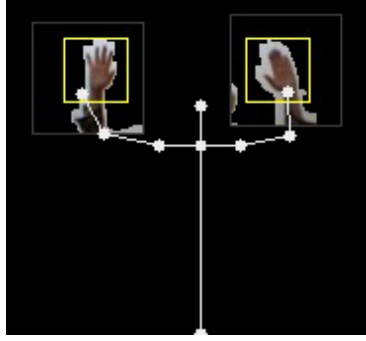


Figure 5-4: Hand tracking is performed only in two small search regions around estimated wrist positions. Given a 56x56 pixel search region (gray rectangle around each wrist position) and 32 x 32 pixel sliding window moving at 2 pixel interval, we compute the HOG features repeatedly.

pixels for the search region, 32 x 32 pixels for hand image window, and 2 pixels for an interval of the sliding window (depicted in Figure 5-4).

Note that body pose estimation results are not always accurate, in which case a search region might not contain a hand. We compensate for this error by using the position of the previously classified hand pose: if a hand pose is found at time t , for the next time $t + 1$ we set the center position of a search region as the geometric mean of the classified position at time t and the estimated wrist position at time $t + 1$ ².

5.4.4 Clustering

We would like to obtain a single classification result per hand, but performing hand tracking using a sliding window over a search region means there can be multiple classification results per hand. Therefore, we cluster all classified results within the same search region (hence a single classification result per hand), taking an average of the positions and probability estimates of the classification results to obtain a single classification result.

Figure 5-5 depicts this clustering process. The small circles inside a rectangle shows each

²Also note that we can refine the body pose estimation result using the classified hand pose position. We leave this as a future work.

individual classification results, where the rectangle is obtained by clustering. The figure also explains two types of classification results, which will be explained next.



Frame	Hand	Opened Palm (blue)	Closed Palm (red)	Nothing
t	Left	95%	0%	5%
	Right	87%	0%	13%
$t+1$	Left	66%	21%	13%
	Right	8%	78%	14%
$t+2$	Left	0%	96%	4%
	Right	1%	78%	21%

Figure 5-5: Three time-consecutive images (top) and probability estimates (bottom) showing hand pose classification results. The small circles inside the rectangle are detected hand locations, and the rectangle is a clustered result where its location and probability estimates are obtained by averaging neighboring classification results. The colors indicate classified hand pose classes: blue for the "opened palm" and red for the "closed palm." A hand pose with the highest probability estimate is chosen as a categorical result value (shown in bold text in the table.)

5.4.5 Output Feature Type

Now we have two types of output feature types: a vector of probability estimates, and a categorical value obtained by selecting the highest probability. There are advantages and disadvantages for each of the two output feature types.

An advantage of using a categorical value is that it helps the gesture recognition framework to have a lower dimensional input feature vector, as we only need a two-dimensional vector describing both hands. However, a disadvantage of it is that it might lose fine details of hand poses, which can be problematic when a probability estimate for hand poses are close to each other. To see this, look at the middle image in Figure 5-5. Both hands are in the transition from the opened palm to the closed palm, so it is not clear as to which hand poses they represent. A categorical value picks the one with the highest probability estimate, so the left hand pose is classified as an opened palm (66%) while the right hand pose is classified as a closed palm (78%).

Loosing fine details of probability estimates turns out to be problematic for gesture recognition; gesture recognition using probability estimates resulted in significantly higher accuracy rates than using the categorical results (we will show this in Section 6.4.3), and this is the advantage of using a vector of probability estimates. However, using a vector of probability estimates has its disadvantage as well: we need an eight-dimensional feature vector to describe two hand poses with the probability estimates, which in turn increases the complexity of gesture recognition.

We compare the performance of both hand pose feature types in Section 6.4.3, where we conduct various experiments on gesture recognition framework using combinations of body and hand poses.

5.5 Experiment

To calibrate and evaluate the accuracy of the hand pose classifier, we performed two experiments: the first one *quantitatively* measured the classification accuracy on manually-segmented images, and the second one *qualitatively* measured the classification accuracy on a subset of non-segmented images.

5.5.1 Quantitative Analysis

The first test was performed using a dataset containing manually-segmented images that included both positive and negative samples. We conducted a 10-fold cross validation (10-CV) analysis: the dataset was divided into 10 individual subsets, each subset containing images from a single participant only; evaluation was repeated 10 times, each time using one of the 10 subsets as a test dataset and the rest as a training dataset.

The 10-CV analysis resulted in 99.94% classification accuracy rate. As reported in [19], where they reported that the performance on pedestrian detection was near-perfect, we can see that the HOG features performs reliably on this object detection task as well.

5.5.2 Qualitative Analysis

What matters more is how well the hand pose classifier performs on our video recorded images, rather than on manually-segmented images. To exhibit this, the second test was performed similar to the qualitative analysis we did for body pose estimation (Section 4.4.2).

We randomly sampled a subset of image frames from four gestures that contained the canonical hand poses: #2 (affirmative), #3 (negative), #20 (brakes on), and #21 (brakes off). After classification was performed, the results were overlaid on the original images, for us to be able to visually compare the classification results to the actual hand poses in the images. Then we counted the number of *misses* (no classified result although there was a hand pose in an image) and *misclassifications* (classified result did not match the hand pose in an image), and combined them to obtain the number of erroneous results. For the simplicity of the evaluation, we used categorical values as the classification results (chosen as the one with the highest probability estimate).

Table 5.5.2 shows error counts and accuracy rates for the four gesture video sequences. Gesture #2 and #3 showed reasonable performances, with slightly less classification accuracy rates compared to the quantitative analysis result (99.94%) being resulted from some

Index	Description	Missed	Misclassified	Total	Accuracy
#2	Affirmative	11	6	379	95.51%
#3	Negative	12	5	368	95.91%
#20	Brakes on	84	0	804	89.55%
#21	Brakes off	155	0	787	80.30%

Table 5.1: Hand pose classification accuracy rates.

rotational variances (thumb up with 45 degree rotation). Gesture #20 and #21 showed much less successful performances compared to the other two gestures as well as to the qualitative analysis result. A majority of these errors were due to imperfect body pose estimation results, i.e., search regions that did not include hand images.

5.6 Related Work

Similar to body pose estimation, there are two main approaches to hand pose estimation: a model-based approach (construct a skeletal model and estimate joint angles) and an appearance-based approach (learn a direct mapping from images to labels, usually computing image descriptors). In a broad term, the two approaches are equally applicable to hand pose estimation. Since we have already mentioned the two approaches in Section 4.5, here we review some of the related work. For a comprehensive review of hand pose estimation, see a survey paper [25].

In [13], Buehler *et al.* performed hand pose estimation following an appearance-based approach. They first segmented the 'shape' of the hands using a graph cut method, with a potential given by the probability of each pixel being explained by skin or background color models. From the segmented shape of the hands, they extracted HOG descriptors, and applied a vector quantization method to represent a hand's HOG descriptor by its nearest 'exemplar' hand shape, where the exemplar set is determined by performing K-means clustering of the corresponding HOG descriptors. Hand poses are classified on-line by finding the nearest exemplar (measured by Euclidean distance between HOG descriptors).

In [36], Kolsch *et al.* performed view-dependent hand pose classification on eight canonical hand poses using a cascade of classifiers that was introduced by Viola and Jones [71]. For each hand pose, a cascade of binary classifiers were trained on a data set consisting of Haar-like features extracted from manually-segmented images. Evaluation was done for each hand pose, achieving 92.23% recognition rate on the *closed* hand pose as the highest one. However, their work did not include implementing a multi-class hand pose classifier.

Chapter 6

Gesture Recognition

This chapter describes multi-signal gesture recognition, the final step in the gesture recognition pipeline. The goal here is to predict a gesture label given a temporal sequence of multi-signal input data that consists of body and hand poses (described in Chapter 4 and Chapter 5, respectively). In order to capture the dependencies within the data from multiple channel inputs more precisely, we propose a new approach to learning a discriminative hidden-state graphical model: Multi Information-Channel Hidden Conditional Random Field (MIC-HCRF) is an extension to the previous work on HCRF [59], which generalizes the framework to accept multiple input vectors.

We first review previous work on discriminative graphical models (Section 6.1 and Section 6.2), and describe the MIC-HCRF in detail (Section 6.3). Next, a comprehensive set of experiments are performed: (a) we test our hypothesis that combining body pose and hand pose helps gesture recognition, (b) we compare the performance of various compositions of body pose and hand pose output formats, and (c) we compare the recognition accuracy of MIC-HCRF to HCRF (Section 6.4). Finally, we review some of the other approaches to multi-signal gesture recognition (Section 6.5).

6.1 Generative vs. Discriminative Models

Generative and discriminative models are two important approaches to building a statistical inference framework in machine learning research. Given a set of input variables $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and corresponding class labels $\mathbf{y} = \{y_1, \dots, y_N\}$, the goal is to learn a classifier $p(\mathbf{y} | \mathbf{x})$ that predicts a class label \mathbf{y} given a new observation \mathbf{x} .

Generative approaches construct the underlying probability distributions to be able to *generate* synthetic data points, while discriminative approaches focus on learning how to best *discriminate* the given feature vector space. To this end, generative models aim to learn a classifier by constructing class conditional probability distributions $p(\mathbf{x} | \mathbf{y})$ and a prior probability distribution $p(\mathbf{y})$ from the training data. Discriminative models, by contrast, aim to learn a classifier by constructing posterior probability distributions $p(\mathbf{y} | \mathbf{x})$ directly from the training data.

The main problem in the generative approach lies in the fact that it requires modeling $p(\mathbf{x} | \mathbf{y})$. In most cases, input features are highly dependent on each other, and accurately modeling such dependencies in $p(\mathbf{x} | \mathbf{y})$ is computationally intractable. To make the problem tractable, the generative approach simplifies the problem by making conditional independence assumptions, or the Naive Bayes assumption. That is, it assumes individual attributes describing an observation are independent of each other once the class label is known:

$$(\forall i, j) p(\mathbf{x}_i | \mathbf{x}_j, \mathbf{y}) = p(\mathbf{x}_i | \mathbf{y}). \quad (6.1)$$

Therefore, with the Bayes assumption, we can simplify describing $p(\mathbf{x} | \mathbf{y})$ as follows:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathbf{y}) = \prod_{i=1}^N p(\mathbf{x}_i | \mathbf{y}) \quad (6.2)$$

Where the conditional independence assumption holds, the generative approach can work surprisingly well (e.g., HMMs [60]). However, if the assumption does not hold, the asymptotic accuracy of the classifier is often worse than the accuracy of a discriminatively trained

classifier [47].

The discriminative approach assumes that $p(\mathbf{y}|\mathbf{x})$ can be represented in a certain parametric format, and aim to find the optimal parameter values that best describes the training data, not requiring modeling of $p(\mathbf{x} | \mathbf{y})$. This in turn allows us not to make the conditional independence assumption. Therefore, this approach can be more robust if the Naive Bayes assumption does not hold for the data. Along with this, it accounts for the whole input data sequence at once during training, allowing us to capture complex long-range dependencies within the input data.

6.2 Discriminative Graphical Models

6.2.1 Conditional Random Field

The Conditional Random Field (CRF) [38] is a framework for building probabilistic models to segment and label sequential data in a discriminative fashion. The CRF assumes an undirected graphical model, which is designed to capture dependencies of attributes in observations. It constructs a conditional model $p(\mathbf{y}|\mathbf{x})$ by assuming that, when conditioned on \mathbf{x} , the random variables \mathbf{y} obey the Markov property with respect to the underlying graph, forming a first order Markov chain. When the underlying graphical model has a tree structure (which forms a chain structure), its cliques are its edges and vertices. Then the conditional model $p(\mathbf{y} | \mathbf{x}; \theta)$ with a parameter vector θ can be constructed as follows:

$$p(\mathbf{y} | \mathbf{x}; \theta) = \frac{1}{Z(\mathbf{x}; \theta)} e^{\Psi(\mathbf{y}, \mathbf{x}; \theta)} \quad (6.3)$$

where

$$\Psi(\mathbf{y}, \mathbf{x}; \theta) = \sum_{v \in V} \theta_V \cdot f(v, \mathbf{y}|_v, \mathbf{x}) + \sum_{(i,j) \in E} \theta_E \cdot f((i,j), \mathbf{y}|_{(i,j)}, \mathbf{x}). \quad (6.4)$$

Here, \mathbf{x} is an input data sequence, \mathbf{y} is a class label sequence, $\mathbf{y}|_S$ is the set of components of \mathbf{y} associated with the vertices in subgraph S , and $Z(x; \theta)$ is a normalization constant that does not depend on the observation. The parameter vector $\theta = [\theta_V \theta_E]$ is estimated using iterative scaling algorithms.

6.2.2 Hidden Conditional Random Field

The Hidden Conditional Random Field (HCRF) [59] is an extension to the previous work on CRFs to incorporate hidden variables in the graphical model. The idea of introducing hidden variables has been empirically shown to improve classification performance in generative graphical models (e.g., HMM [60]). It assumes that the variables *observable* to the user form an undefined stochastic process, which is linked to *hidden* variables that govern the distribution of the observable variables.

The HCRF constructs the conditional model $p(y | \mathbf{x}; \theta)$ as the marginals over hidden variables, \mathbf{h} :

$$p(y | \mathbf{x}; \theta) = \sum_{\mathbf{h}} p(y, \mathbf{h} | \mathbf{x}; \theta) = \frac{1}{Z(y | \mathbf{x}; \theta)} \sum_{\mathbf{h}} e^{\Psi(y, \mathbf{h}, \mathbf{x}; \theta)} \quad (6.5)$$

where

$$\Psi(y, \mathbf{h}, \mathbf{x}; \theta) = \sum_{v \in V} \theta_V \cdot f(v, \mathbf{h}|_v, y, \mathbf{x}) + \sum_{(i,j) \in E} \theta_E \cdot f((i,j), \mathbf{h}|_{(i,j)}, y, \mathbf{x}). \quad (6.6)$$

Here, \mathbf{h} is a hidden state sequence, \mathbf{x} is an input data sequence, and y is a class label. Similar to the CRF formulation, $\mathbf{h}|_S$ is the set of components of \mathbf{h} associated with the vertices in subgraph S , and $Z(y | \mathbf{x}; \theta)$ is a normalization constant that does not depend on the hidden variables. Parameter estimation for $\theta = [\theta_V \theta_E]$ is performed using the belief propagation algorithm [57], with one restriction being made on the graphical model to form a tree structure.

6.3 Multi Information-Channel Hidden Conditional Random Field

6.3.1 Motivation

Previous work on discriminative hidden-state graphical models have concentrated on learning a direct mapping function from single-channel input feature vectors to class labels, regardless of whether the input signals are obtained from a single information-channel or multiple information-channels. However, consider a case where input signals are obtained from multiple information channels, with each channel having different characteristics (as in many multimodal interaction application scenarios, e.g., speech-and-gesture recognition system [63]). In such a case, considering our intuition about the way humans process multiple signals, it might be too restrictive to force input features to form a single-channel vector.

Also, performing a parameter estimation is essentially a way of finding out the best discriminator with the lowest error in a hyper-dimensional space; so, it must be followed with an assumption that the input feature space is separable with reasonable errors. However, by putting all features into one vector, we are implicitly forcing features with different characteristics to be in the same hyper-plane space. Therefore, it is not clear as to how or why it can be separable when a space contains the combined multi-channel signals.

Multi Information-Channel Hidden Conditional Random Fields (MIC-HCRFs) are designed to accept an arbitrary number of feature vectors, with each vector representing an individual input channel with its own characteristics. Therefore, compared to the previous work on HCRF for gesture recognition (e.g., [73]), we get more flexibility in modeling the graphical model. For example, the number of hidden states can be set individually for each channel, in which case we can make more sense of the dynamics of hidden states; the relationship between hidden states and observations can be customized in more detail, i.e., by specifying dependencies among multiple input channels or hidden states. In higher level,

we believe that this flexibility in modeling the graphical model will allow us to construct a more robust estimation framework for multi-signal pattern recognition tasks.

6.3.2 Formulation

The goal here is to learn a posterior distribution $p(y | \widehat{\mathbf{x}}; \theta)$ given a labeled training dataset $(\widehat{\mathbf{x}}_i, y_i)$. Here, $\widehat{\mathbf{x}}_i = (\mathbf{x}_i^{(1)}, \dots, \mathbf{x}_i^{(K)})$ is the i -th set of observation sequences collected from K -channels, with each $\mathbf{x}_i^{(k)} = (\mathbf{x}_{i,1}^{(k)}, \dots, \mathbf{x}_{i,T_i}^{(k)})$ is a sequence of observations from the k -th channel, and y_i is the i -th class label. For each set of observation sequences $\widehat{\mathbf{x}}_i$, there is assumed to exist a corresponding set of hidden state sequences $\widehat{\mathbf{h}}_i = (\mathbf{h}_i^{(1)}, \dots, \mathbf{h}_i^{(K)})$. Note that, each of the i -th set of sequences $\widehat{\mathbf{x}}_i$ can have different sequence length T_i , but each observation channel $\mathbf{x}_i^{(k)}$ within the set of sequences must have the same sequence length. Also note that, the number of hidden states $|\mathcal{H}^{(k)}|$ can differ from channel to channel.

The posterior distribution $p(y | \widehat{\mathbf{x}}; \theta)$ is defined as follows:

$$p(y | \widehat{\mathbf{x}}; \theta) = \sum_{\widehat{\mathbf{h}}} p(y, \widehat{\mathbf{h}} | \widehat{\mathbf{x}}; \theta), \quad (6.7)$$

$$p(y, \widehat{\mathbf{h}} | \widehat{\mathbf{x}}; \theta) = \frac{e^{\Psi(y, \widehat{\mathbf{h}}, \widehat{\mathbf{x}}; \theta)}}{\sum_{y', \widehat{\mathbf{h}}'} e^{\Psi(y', \widehat{\mathbf{h}}', \widehat{\mathbf{x}}; \theta)}}, \quad (6.8)$$

where $\Psi(y, \widehat{\mathbf{h}}, \widehat{\mathbf{x}}; \theta)$ is a potential function that specifies dependencies in the model. In this work, following previous work on HCRF [59], the underlying graph $G = (V, E)$ is assumed to have a chain structure (see Figure 6-1 (b)) with vertices V and edges E . Following the definitions of $\widehat{\mathbf{x}}$ and $\widehat{\mathbf{h}}$, we will use $\widehat{v} \in V$ to refer to a set of all vertices that are linked to each channels with a single observation.

Next, the potential function is defined as

$$\Psi(y, \widehat{\mathbf{h}}, \widehat{\mathbf{x}}; \theta) = \sum_{v \in V} \sum_{k=1}^K \phi(\mathbf{x}_v^{(k)}) \cdot \theta(\mathbf{h}_v^{(k)}) + \sum_{v \in V} \sum_{k=1}^K \theta(y, \mathbf{h}_v^{(k)}) + \sum_{(\widehat{v}_1, \widehat{v}_2) \in E} \theta(y, \widehat{\mathbf{h}}_{\widehat{v}_1}, \widehat{\mathbf{h}}_{\widehat{v}_2}) \quad (6.9)$$

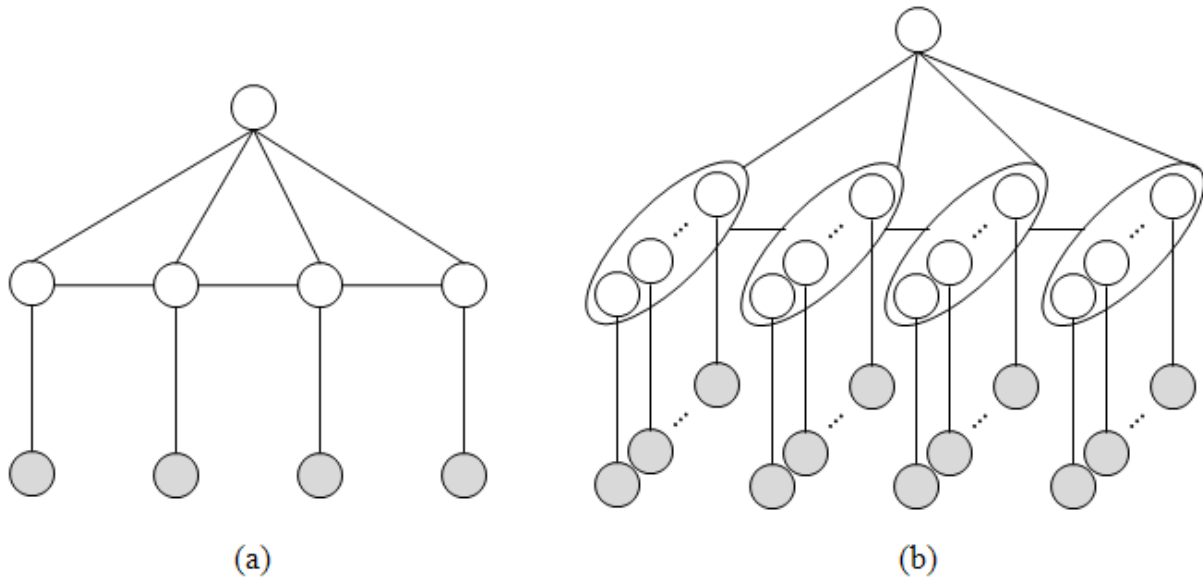


Figure 6-1: Discriminative hidden-state graphical models. (a) HCRF, (b) MIC-HCRF.

where $\phi(\mathbf{x}_t^{(k)})$ is a feature extraction function for the k -th channel input data. The main difference in the potential function of this work compared to HCRF is the use of $\hat{\mathbf{x}}$ and $\hat{\mathbf{h}}$. Note that we would like to capture dependencies between input features and hidden variables *per channel*. This is modeled in the first two terms in Eq. 6.9 by computing values for each individual channel k and marginalizing them over all channels.

In the third term, however, instead of marginalizing values over all channels, all hidden variables for each vertex $\hat{v} \in V$ are combined to create another combinatorial hidden state variable, and values are computed with the combinatorial hidden state variables over the connected edges. To see how the combinatorial hidden state variables are constructed, consider a case where the observation contains two channels with $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, and two corresponding hidden state variables $\mathbf{h}^{(1)} \in \mathcal{H}^{(1)}$ and $\mathbf{h}^{(2)} \in \mathcal{H}^{(2)}$. Then the combinatorial hidden state variables can be constructed by assigning unique values to each of the combinations in $\mathbf{h} \in (\mathcal{H}^{(1)} \times \mathcal{H}^{(2)})$.

Following the previous work on CRF [38], parameters are trained using the following ob-

jective function:

$$L(\theta) = \sum_{i=1}^n \log P(y_i | \hat{\mathbf{x}}_i, \theta) - \frac{1}{2\sigma} \|\theta\|^2 \quad (6.10)$$

where the second term is introduced for regularization. Then the optimal parameter values are obtained using the maximum log-likelihood estimator:

$$\theta^* = \arg \max_{\theta} L(\theta) \quad (6.11)$$

Finally, a class label for a new observation $\hat{\mathbf{x}}$ is classified as:

$$y^* = \arg \max_{y \in \mathcal{Y}} p(y | \hat{\mathbf{x}}; \theta^*) \quad (6.12)$$

To find the maximum log-likelihood estimator for Eq. 6.9 from the training dataset, we use the L-BFGS algorithm that works well for high-dimensional vectors. Note that the graph in Figure 6-1 does not contain any loops. Furthermore, computing gradients in Eq. 6.9 is a simple extension to the previous work [59]. Therefore, we can use the same method of computing the gradient of $L(\theta)$ using belief propagation as introduced in [59].

An important thing to notice in computing the gradients is that the added complexity of the potential function compared to the previous version of HCRF [59] is only $\mathcal{O}(n^2)$. The complexity of the first two terms increases in $\mathcal{O}(n)$ as the number of channels and the number of hidden channels increases, because we are just marginalizing over all channels. The complexity of the last term increases in $\mathcal{O}(n^2)$ which is square to the number of hidden states, because we are computing transitions between two combinatorial hidden state variables.

6.3.3 MIC-HCRF For Gesture Recognition

Now we explain the specific formulation of MIC-HCRF for gesture recognition used in this work. The input is a sequence of video images where each image contains two signals:

the body pose signal and the hand pose signal. The input vector can be expressed as $\widehat{\mathbf{x}}_i = (\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)})$ where the first channel is for the body pose signal and the second channel is for the hand pose signal.

Features for each channel $\phi(\mathbf{x}_t^k)$ are extracted in the ways described in the previous chapters (Chapter 4 for body pose and Chapter 5 for hand pose). Body pose features $\phi(\mathbf{x}_t^{(1)})$ can be any one of the features (or a combination of multiple features) extracted from the body pose estimation framework, including joint angles, angular speeds of joint angles, relative coordinates of joint movements, and velocities of joint movements. Similarly, hand pose features $\phi(\mathbf{x}_t^{(2)})$ can be either one of the two features (or both features) that were extracted from the hand pose classification framework, including the categorical classification results (which hand pose it was) and a vector of continuous values (probability estimates for being each of the hand poses in the vocabulary).

The three terms in Eq. 6.9 can be interpreted as follows. The first term, the inner products of input features $\phi(\mathbf{x}_t^k)$ and hidden state variables $\theta(\mathbf{h}_t^k)$, specifies dependencies between input features (body or hand signal) and the corresponding hidden states for each channel. The second term specifies dependencies between each channel’s hidden state and a class label. Finally, the third term specifies dependencies between a class label and all combinations of hidden states for two consecutive time frame.

Note that, the first two terms are defined *per channel*, so they aim to capture *intra*-dependencies (i.e., dependencies within each channel). The third term, on the other hand, is defined over all combinations of channels, so it aims to capture *inter*-dependencies (i.e., dependencies among multiple channels).

In HCRF, selecting the number of hidden states $|\mathcal{H}^{(k)}|$ can be subjective, especially when the input feature vector has multiple channels. However, this is a less subjective process in MIC-HCRF: we can set the number of hidden states more intuitively (e.g., count the number of canonical body poses or hand poses in a gesture) because we divide the input channels into individual channels with similar characteristics,

6.4 Experiments

6.4.1 Dataset

We tested our gesture recognition framework with a subset of standard aircraft handling signals defined in NATOPS [53]. Five pairs of similar gestures were selected that we believe well represent the intricacy of the entire gesture set (Figure 6-2). The pairs of gestures included:

- #2 (affirmative) and #3 (negative),
- #4 (spread wings) and #5 (fold wings),
- #10 (remove chocks) and #11 (insert chocks),
- #18 (engage nosegear steering) and #19 (hot brakes), and
- #20 (brakes on) and #21 (brakes off).

The reason for selecting pairs of similar gestures was to evaluate whether or not our gesture recognition framework well captures complex and subtle dependencies in each pair of gestures. For example,

- the gesture pair #10 and #11 both start with spreading arms about 45 degree front, then waving them inward and outward two times. In the collected dataset, the starting points and ending points of the arm movements varied from participants to participants, which made gesture recognition task even more challenging;
- the gesture pair #18 and #19 have the same left hand movements and similar right hand movements: in #18 the right hand points to the nose, while in #19 the right hand slightly moves up and down around the nose;
- the two pairs of gestures, #2 & #3 and #20 & #21, are also similar: the only way to distinguish from each other is to have the knowledge about hand poses.

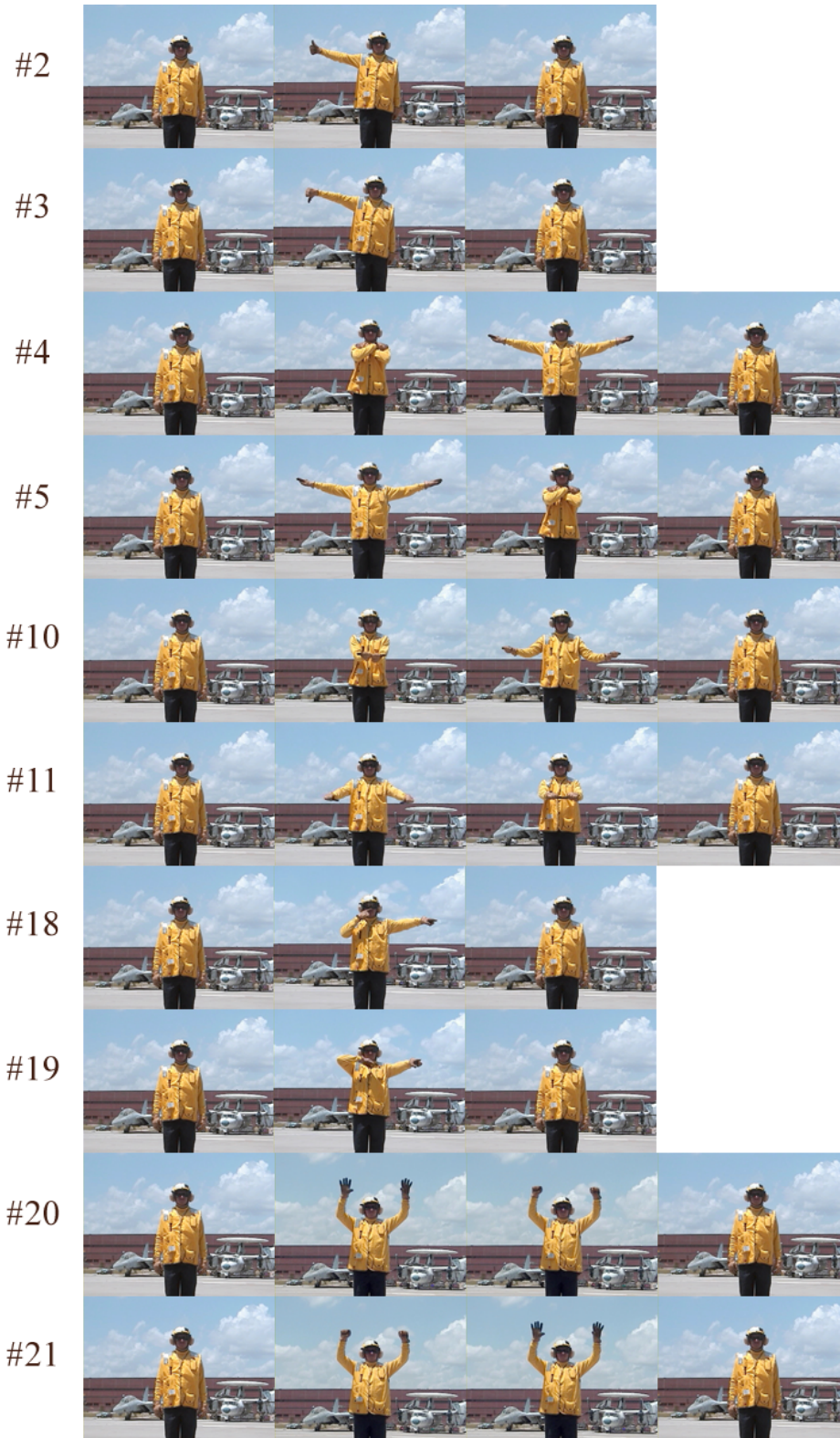


Figure 6-2: A set of 10 gestures used in this work. #2: Affirmative, #3: Negative, #4: Spread wings, #5: Fold wings, #10: Remove chocks, #11: Insert chocks, #18: Engage nosegear steering, #19: Hot brakes, #20: brakes on, #21: brakes off.

Using the body pose estimation and the hand pose classification, we extract various feature types: four body pose feature types and two hand pose feature types. The four body pose feature types include:

- joint angles (T),
- joint coordinates (P),
- derivatives of joint angles (dT), and
- derivative of joint coordinates (dP).

The two hand pose feature types include:

- a vector of probability estimates (S),
- a categorical value indicating which hand pose it is, obtained by selecting a hand pose class with the highest probability estimate (H)

Throughout this section, we will use combinations of these notations to refer to a dataset type, e.g, dTS for the combination of derivatives of body pose joint angles and hand pose probability estimate vector.

6.4.2 Does Combining Body Pose And Hand Pose Really Help?

Method

The first question was whether combining body and hand poses helps to improve gesture recognition for standard aircraft handling signals. To determine this, we compared recognition accuracy rates of two conditions: the first condition contained information on body pose only; the second condition contained information on body and hand poses combined.

Since there were two hand pose feature types (S and H), for the second condition we measured the recognition accuracy rate for each of them and took an average of both. Also, to make the experiment more accurate, the two conditions (with and without information on hand poses) were compared using each of the four body pose feature types (T, P, dT, and dP). In addition to differing input feature types, we also varied the number of hidden states, as it also affects the performance; for each test case we used 3 and 4 hidden states. In all test cases, the regularization factor was set to 1,000 (to prevent overfitting), chosen as the best performing value based on our preliminary experiments. All in all, there were in total 24 individual test cases (4 body pose feature conditions, 3 hand pose feature conditions, and 2 number of hidden states conditions).

We performed a 4-fold cross validation (4-CV) analysis for each of the 24 test cases, measuring precision and recall of each of the 10 gestures. The overall recognition accuracy rate for the 10 gesture set was obtained using the F1 score, a weighted average of the precision and recall ($F1 = 2 * \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$). Since a 4-CV analysis performs four repetitive tests, we also get variances of recognition accurate rates; we performed independent samples T-tests to see if the differences between two conditions (with and without hand pose) were statistically significant.

Result and Discussion

We first compared the two conditions (with and without hand pose) for each body pose features. Figure 6-3 shows a comparison of the two conditions' overall recognition accuracy rates averaged over the 10 gestures, and Table 6.1 shows the mean and standard deviation of each test condition, as well as the results from independent samples T-tests. In all our test cases, using body and hand poses resulted in higher overall recognition accuracy rates, where for two body pose features (dT and dP) the differences were statistically significant ($p=.001$).

Next, we compared the two conditions (with and without hand pose) for each gesture

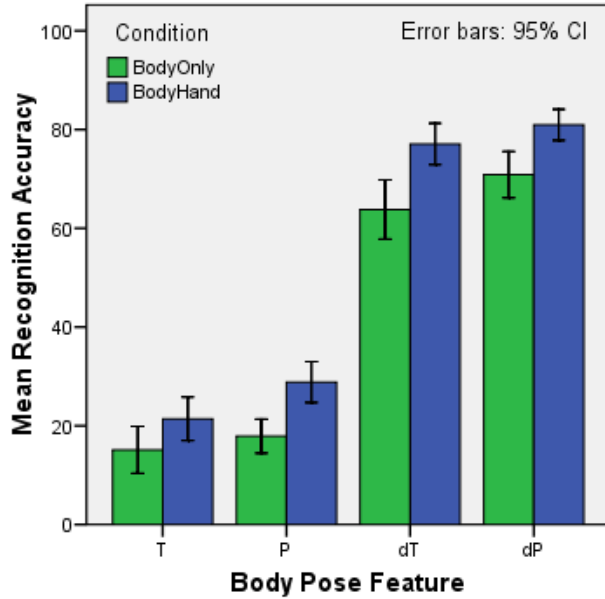


Figure 6-3: Recognition accuracy rates comparing two conditions, a) body pose only (BodyOnly) and b) body and hand pose combined (BodyHand), under the four different body pose features.

Body Feature Type	Condition	Mean	Std. Dev	Independent Samples T-test
T	BodyOnly	20.09	3.57	$t(22)=1.00, p=.326$
	BodyHand	27.02	3.83	
P	BodyOnly	23.26	11.07	$t(22)=1.21, p=.24$
	BodyHand	32.73	20.57	
dT	BodyOnly	62.47	7.21	$t(22)=4.06, p=.001$
	BodyHand	76.23	8.10	
dP	BodyOnly	70.94	6.73	$t(22)=3.82, p=.001$
	BodyHand	80.65	5.30	

Table 6.1: Statistics for the recognition accuracies comparing two conditions, body pose only (BodyOnly) and body and hand pose combined (BodyHand), under the four different body pose features.

class, because hand poses seemed important for four gestures (#2, #3, #20, and #21) only. Figure 6-4 shows comparisons of the recognition accuracy rates for each gesture class¹. The result shows that, for the two gesture pairs, #2 & #3 and #20 & #21, using information on body and hand poses together significantly improved the gesture recognition accuracy (on average 27.5%), while for other 6 gestures there were slight differences, but not significant ones.

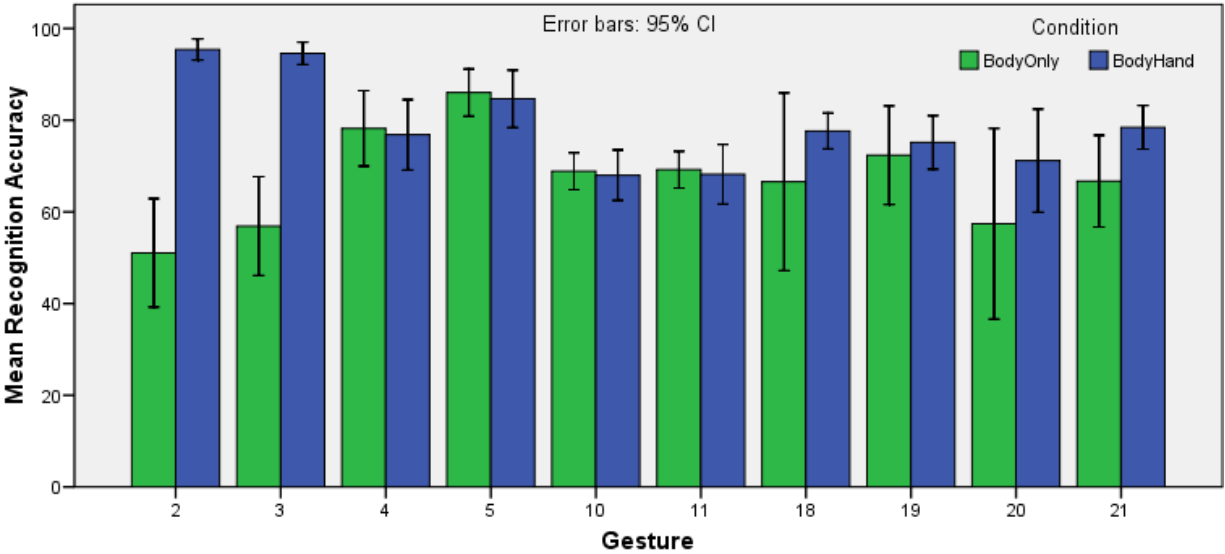


Figure 6-4: Recognition accuracy rates for each gesture class (for body pose features, only dT and dP were used).

¹Note that, for this graph, we excluded two body pose features, T and P, since they performed much worse than the other two features.

6.4.3 What Is The Best Feature To Use For Gesture Recognition?

Method

In this experiment, we show which combination of body and hand pose features allowed the highest gesture recognition accuracy rate. The previous experiment showed that two body features, T and P, resulted in inferior recognition performance compared to the other two features, dT and dP. Therefore, we omitted the two body pose features (T and P), and tested on the other two features (dT and dP) plus a combination of the two (dTdP). For more comprehensive analysis, we varied the number of hidden states from 3 to 5, which resulted in 18 individual test cases (3 body pose feature conditions, 2 hand pose feature conditions, and 3 number of hidden states conditions). Following the previous experiment, the regularization factor was fixed at 1,000 (to prevent overfitting). We performed 10-CV on each individual test cases, and the recognition accuracy rate was calculated in the same way as the previous experiment.

Result and Discussion

We first compared 6 different feature combinations (3 body feature types and 2 hand feature types), while averaging all of the 10 gesture classes and 3 numbers of hidden states (3, 4, and 5). Figure 6-3 shows the recognition accuracy rates for each feature combinations, and Table 6.2 shows the mean and standard deviations of the recognition accuracy rates.

The best performing hand pose feature type was S. In all of our test cases, S performed significantly better than H ($t(178)=2.24$, $p=.018$). Also, the best performing body pose feature type was dP; although there was no significance in the accuracy differences, dP always performed better than dT or dTdP.

Figure 6-6 shows more detailed comparison of the recognition accuracy rates, where we

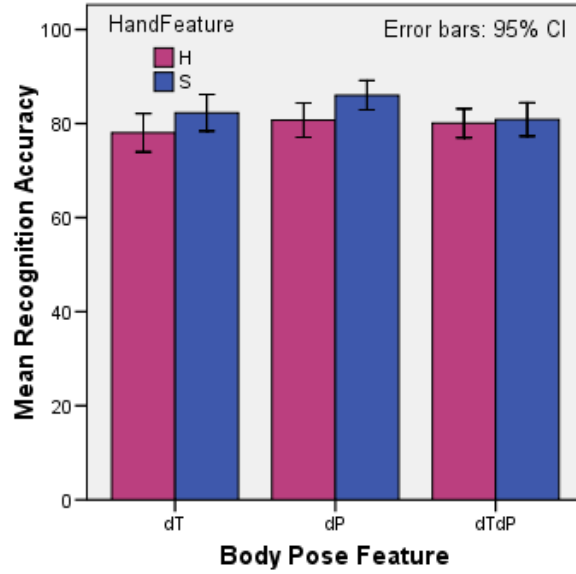


Figure 6-5: Gesture recognition accuracy graph of 6 feature combinations, averaging over all of the 10 gesture classes and 3 numbers of hidden states (3, 4, and 5). Regarding hand pose feature types, S performed significantly better than H in all of our test cases ($t(178)=2.24$, $p=.018$). Regarding body pose feature types, although there was no significance in the accuracy differences, dP performed the best.

Body Feature Type	Hand Feature Type	Mean	Std. Dev
dT	H	78.02	10.97
	S	82.27	10.42
dP	H	80.72	9.85
	S	86.02	8.32
dTdP	H	80.08	8.21
	S	80.86	9.51

Table 6.2: Mean and standard deviations of recognition accuracy rates for each feature combinations.

divided the results per each gesture class. Note that we excluded the categorical hand pose result (H), since it performed significantly worse than the other one (S). In this graph, we can assure that the body pose feature dP performed better than the other features for all gesture classes, except for #4 (spread wings). In all of the following experiments, therefore, we will use dPS –a combination of derivatives of joint coordinates (dP) and vectors of probability estimates (S)– as an input feature vector.

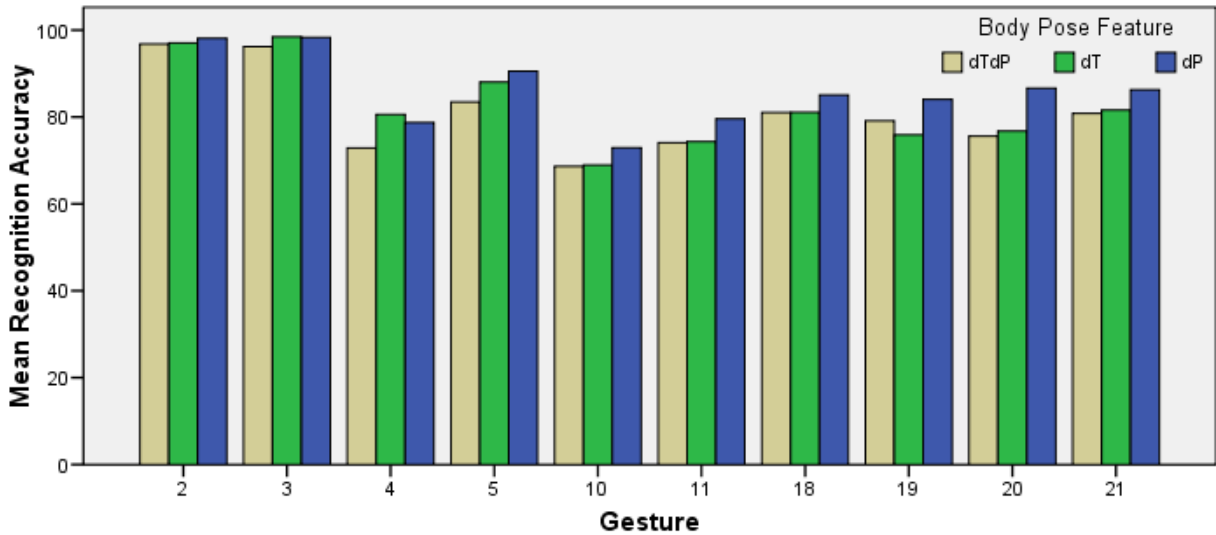


Figure 6-6: Recognition accuracy rates for each gesture class (for hand pose features, only S was used).

We also determined how the number of hidden states affected the recognition accuracy rates. In this experiment, we only compared the recognition accuracy rates of the dPS feature combination, varying the number of hidden states from three to six. As can be seen from Figure 6-7 and Table 6.3, the recognition accuracy rate increased sharply between the number of hidden states 3 and 4, but was stabilized as we added more hidden states. Tukey’s post-hoc comparisons of the four conditions indicated that the accuracy rate difference between the number of hidden states 3 ($M=82.64$, 95% CI[79.43 85.84]) and the rest were marginally significant ($p=.091$ for the number of hidden states 4 ($M=87.22$, 95% CI[84.45 89.99]), $p=.068$ for 5 ($M=87.44$, 95% CI[84.91 89.98]), and $p=.050$ for 6 ($M=87.67$, 95% CI[85.28 90.08])).

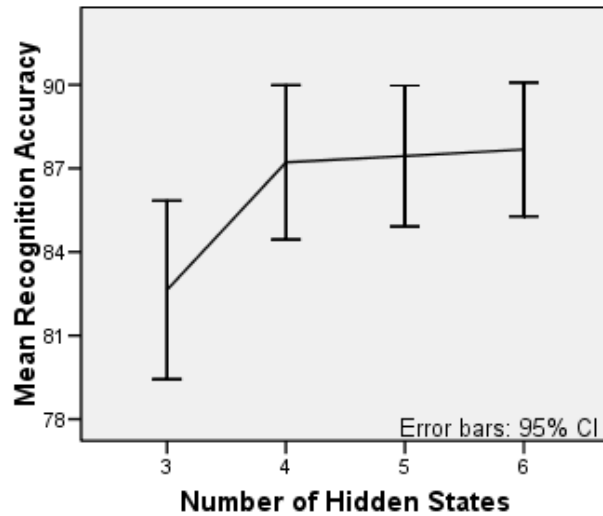


Figure 6-7: Recognition accuracy rates of the dPS feature combination –a combination of derivatives of joint coordinates (dP) and vectors of probability estimates (S)– for each number of hidden states (3, 4, 5, and 6).

Feature Type	Number of Hidden States	Mean	Std. Dev
dPS	3	82.64	16.17
	4	87.22	13.97
	5	87.44	12.75
	6	87.67	12.10

Table 6.3: Mean and standard deviations of recognition accuracy rates of the dPS feature combination (a combination of derivatives of joint coordinates (dP) and vectors of probability estimates (S)) for each number of hidden states (3, 4, 5, and 6).

Lastly, we show patterns of the dPS feature combination extracted from 20 participants, as this will be useful in understanding the result of the next experiment. Figures 6-8, 6-9, 6-10, 6-11, and 6-12 compare the patterns of body and hand pose signals for each pair of gestures using the dPS feature combination, averaging over all trials from 20 participants²³.

As expected, two gesture pairs, #2 & #3 and #20 & #21, showed very similar body pose signal patterns while showing two clearly different hand pose signal patterns. In gestures #2 and #3, there was one line in each hand pose graph indicating either “thumb up” or “thumb down”; in gestures #20 and #21, there were multiple lines in each hand pose graph indicating transitions of hand poses from “palm opened” to “palm closed”, or vice versa.

A pair of gestures #18 and #19 showed similar beginning of body pose signal patterns, although later on the gesture #19 seemed to capture the subtle up-and-down movement of the right arm (the right wrist joint oscillated a bit more than in #18). As expected, the hand pose signals showed no significant results (probability estimates remained mostly at zero).

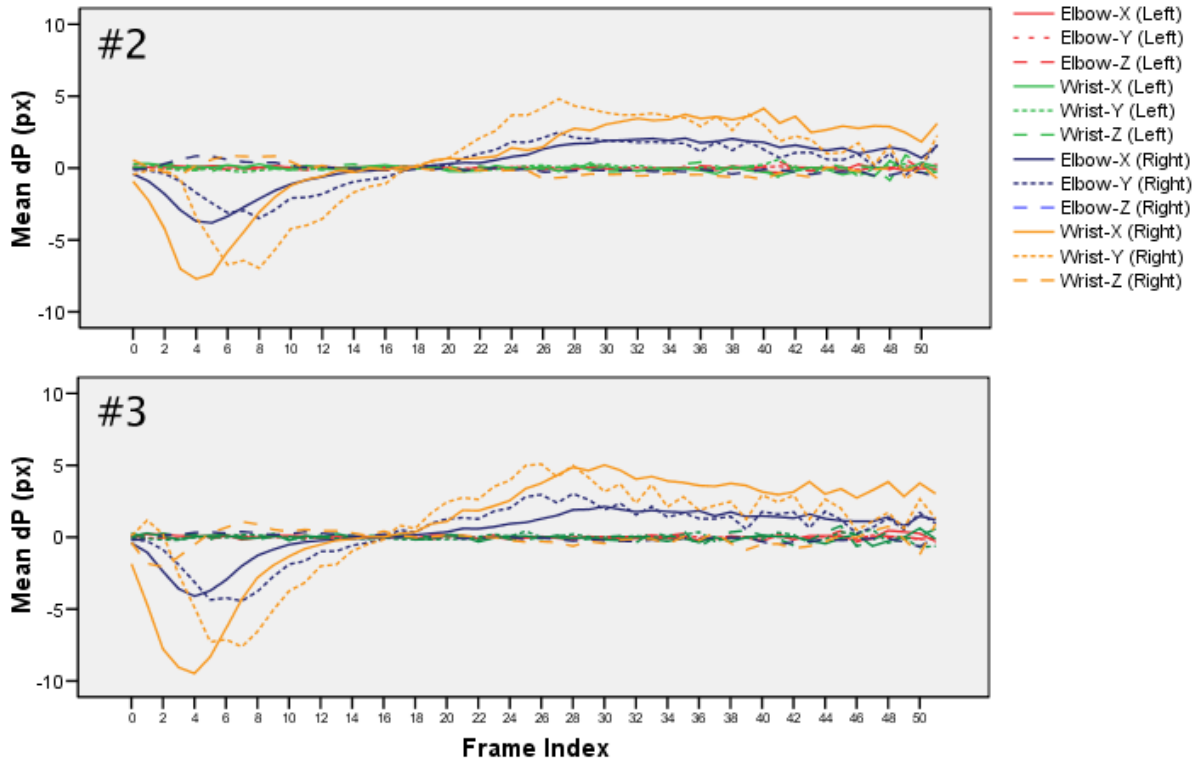
The remaining two gesture pairs, #4 & #5 and #10 & #11, showed reversed patterns of body pose signals, especially in the left and right wrist joint signals. These reversed patterns well represent the actual arm movements of the corresponding gestures, e.g., the velocity of a waving arm is faster for the direction of the thumb. In addition, although not significant, hand pose signals showed some patterns as well: the gestures #4, #5, and #10 showed a slight possibility of the right hand pose being “thumb up”; and #11 showed a slight possibility of the left hand pose being “thumb down”. These patterns indicate that using the probability estimates as the hand pose feature has a potential to capture subtle details of hand poses that might help recognizing gestures, as we discussed in Section 5.4.5.

²When seeing the graphs of body pose signals, we recommend readers look specifically at the patterns of the left and right wrist joints, envisioning the corresponding gestures as shown in 6-2.

³Three gesture pairs (#4 & #5, #10 & #11, and #18 & #19) did not include canonical hand poses, which resulted in low probability estimate values. To enhance the readability, we rescaled the Y-axis of the hand pose graphs in Figures 6-9, 6-10, and 6-11.

Gesture #2 (affirmative) and #3 (negative)

Body Pose: Derivatives of Joint Coordinates



Hand Pose: Probability Estimates

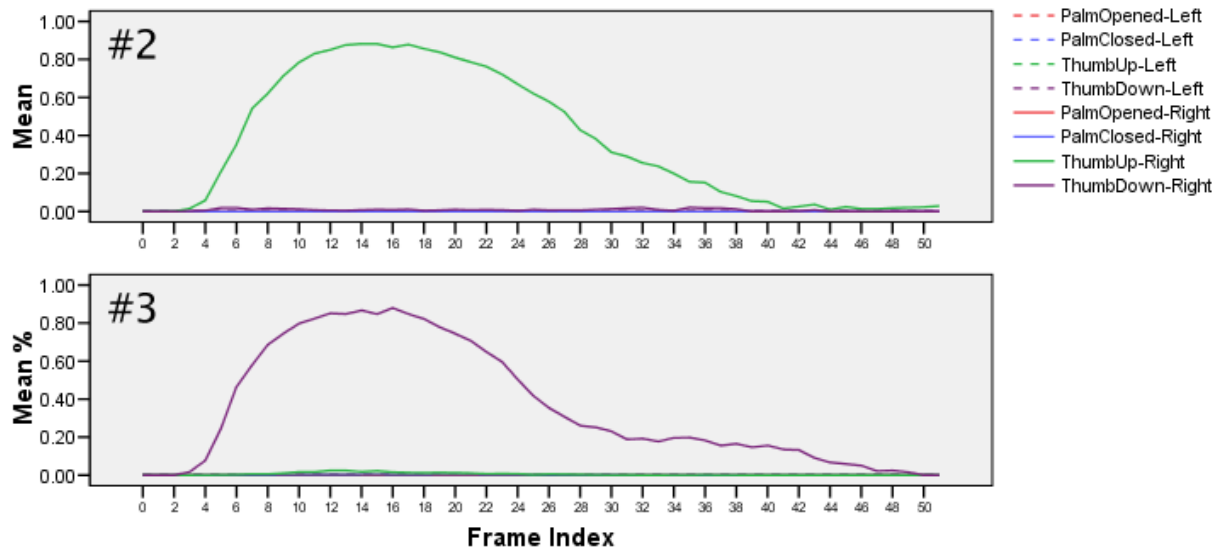
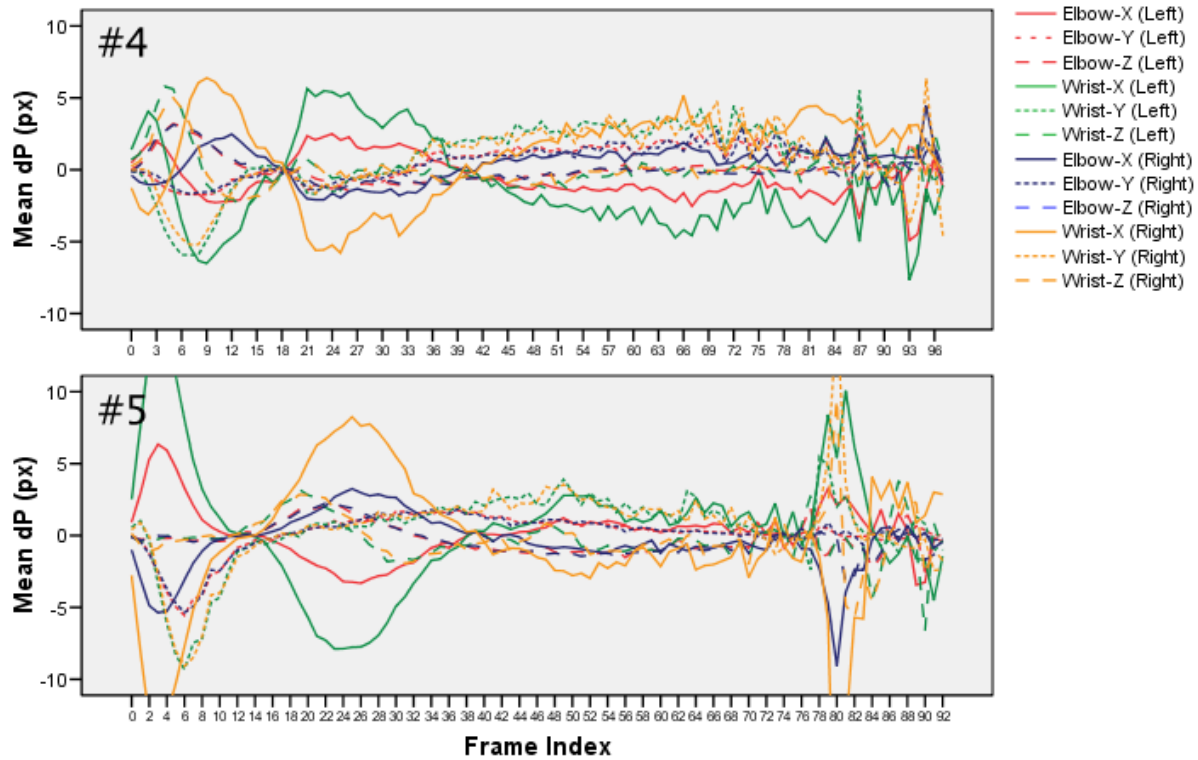


Figure 6-8: Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #2 (affirmative) and #3 (negative), averaged over all trials from 20 participants.

Gesture #4 (spread wings) and #5 (fold wings)

Body Pose: Derivatives of Joint Coordinates



Hand Pose: Probability Estimates

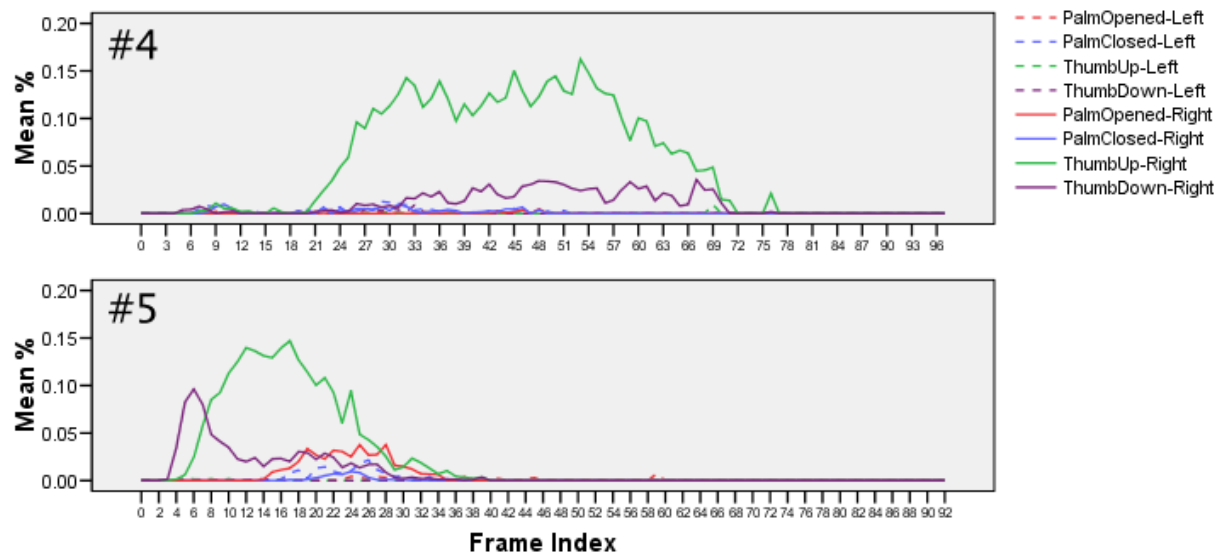
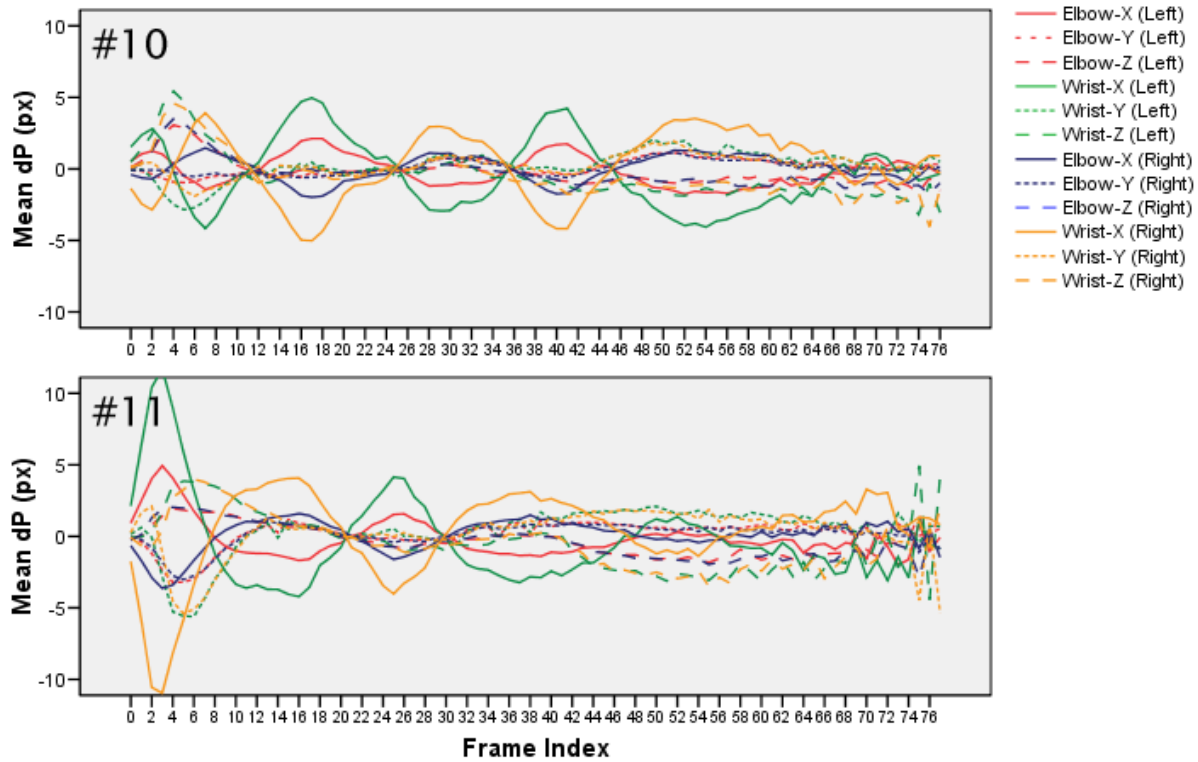


Figure 6-9: Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #4 (spread wings) and #5 (fold wings), averaging over all trials from 20 participants.

Gesture #10 (remove chocks) and #11 (insert chocks)

Body Pose: Derivatives of Joint Coordinates



Hand Pose: Probability Estimates

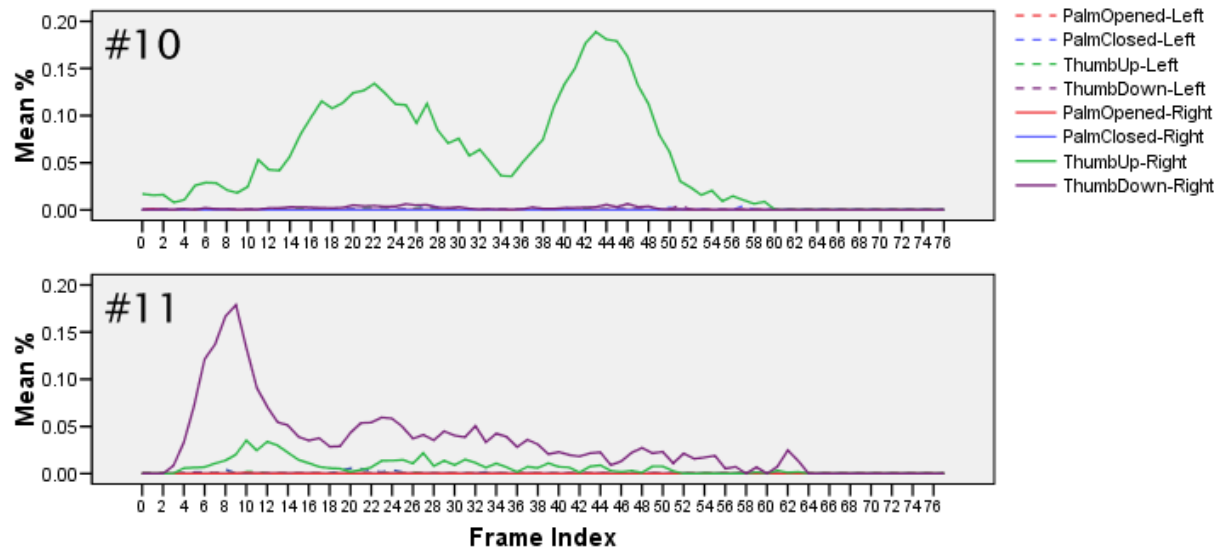
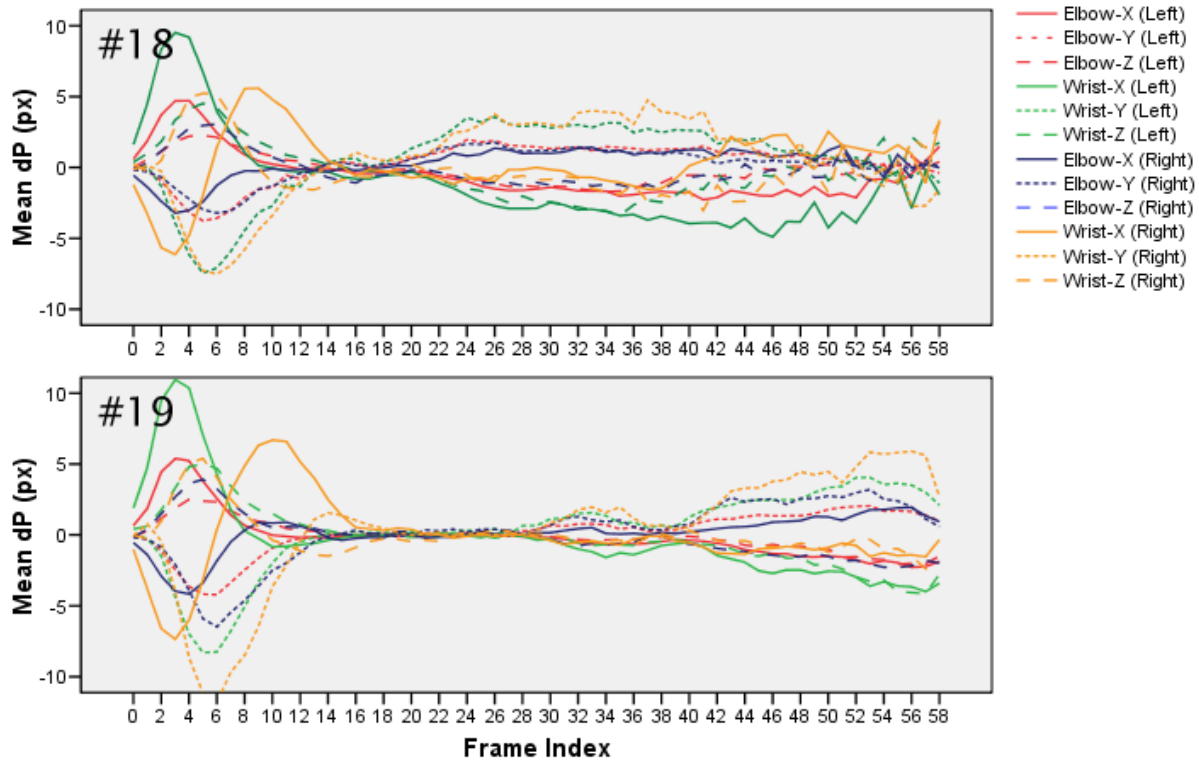


Figure 6-10: Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #10 (remove chocks) and #11 (insert chocks), averaging over all trials from 20 participants.

Gesture #18 (engage nosegear steering) and #19 (hot brakes)

Body Pose: Derivatives of Joint Coordinates



Hand Pose: Probability Estimates

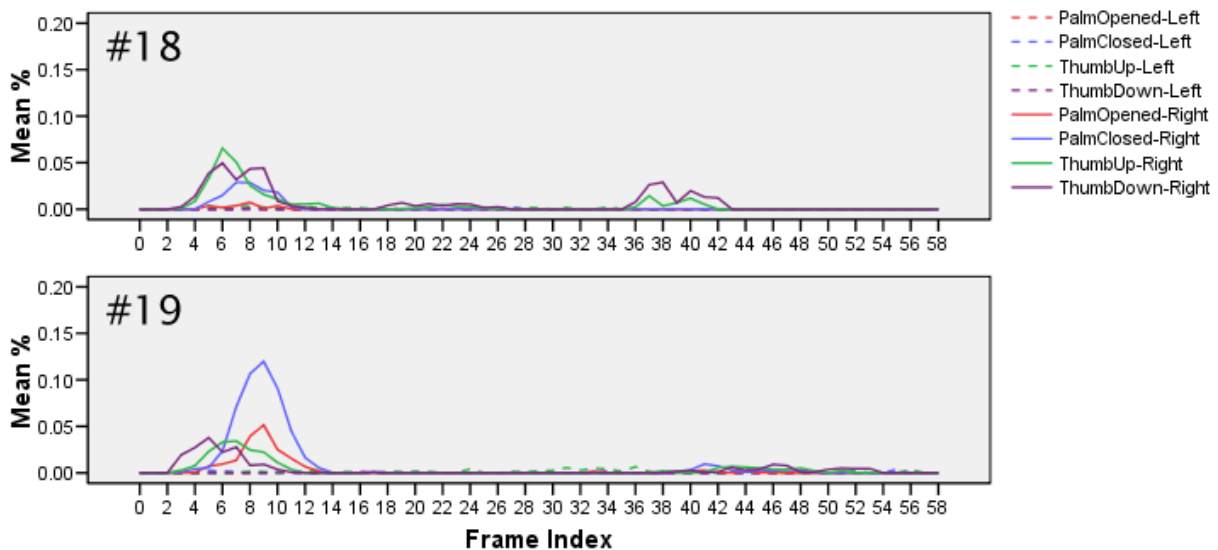
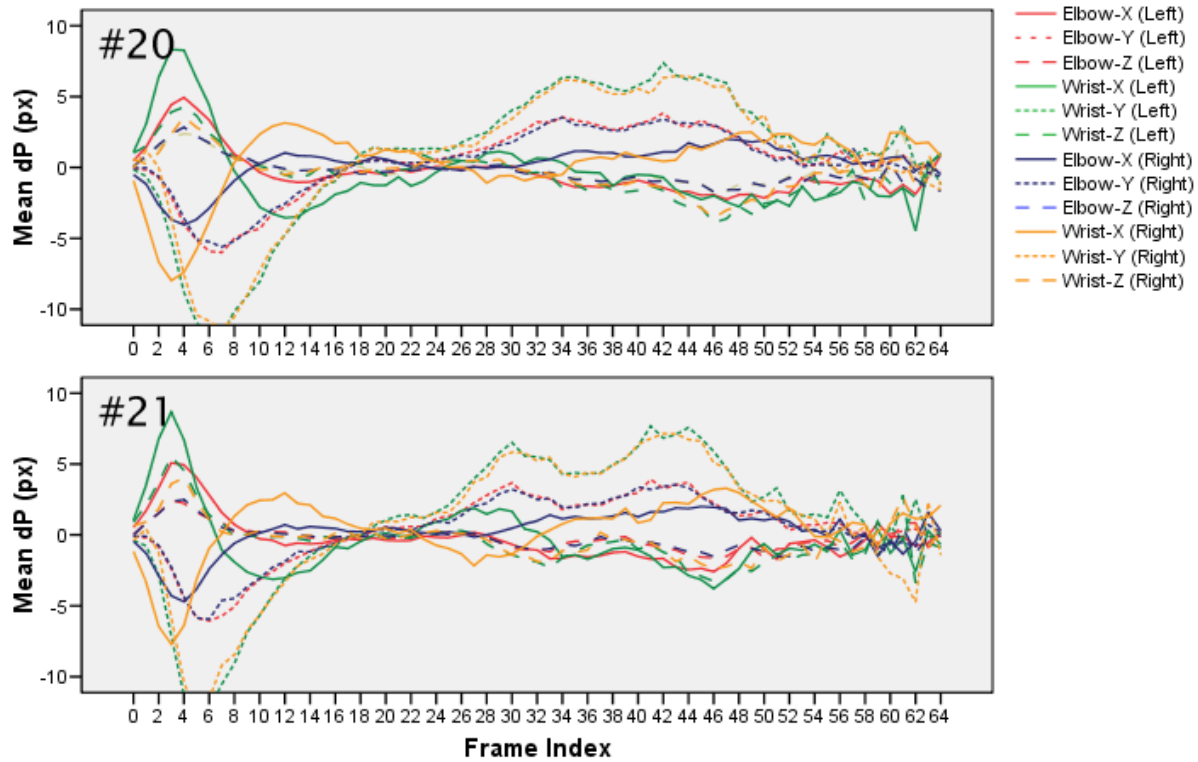


Figure 6-11: Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #18 (engage nosegear steering) and #19 (hot brakes), averaging over all trials from 20 participants.

Gesture #20 (brakes on) and #21 (brakes off)

Body Pose: Derivatives of Joint Coordinates



Hand Pose: Probability Estimates

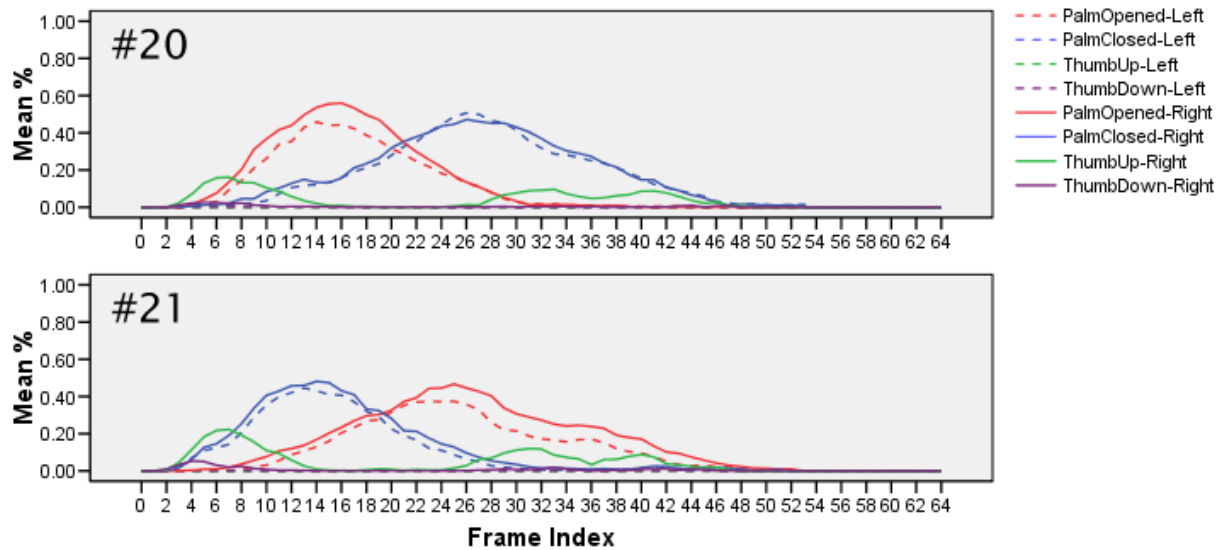


Figure 6-12: Comparisons of the patterns of body feature signals (derivatives of joint coordinates) and hand feature signals (probability estimates) for a gesture pair #20 (brakes on) and #21 (brakes off), averaging over all trials from 20 participants.

6.4.4 How does MIC-HCRF Perform Over HCRF?

Method

In our last experiment, we compared the recognition performance of MIC-HCRF to HCRF using the dPS feature combination (derivatives of joint coordinates (dP) and vectors of probability estimates (S)). Based on the previous experiments, HCRF was configured with 5 hidden states ($|\mathcal{H}|=5$); based on our preliminary experiments, MIC-HCRF was configured with 3 hidden states for the body pose channel and 2 hidden states for the hand pose channel ($|\mathcal{H}^1|=3$, $|\mathcal{H}^2|=2$). In both conditions, the regularization factor was set to 1,000 (to prevent overfitting) following the previous experiments. We performed 10-CV analyses for HCRF and MIC-HCRF, and the recognition accuracy rate was calculated in the same way as the previous experiments.

Result and Discussion

Figure 6-13 and Table 6.4 show the result of this experiment. Overall, MIC-HCRF performed slightly better (88.41%) than HCRF (87.08%). One important thing to note is that this performance is comparable to human pilots: subject matter experts indicated that the system performance was on par with the most US Navy pilots.

MIC-HCRF performed better on six gestures (#4, #5, #10, #11, #18, and #19), where the difference was 3.76% higher on average. For gestures #2 and #3, the performance of MIC-HCRF was on par (on average 0.26% lower than HCRF); for gestures #20 and #21, the performance of MIC-HCRF was less successful (on average 5.4% lower than HCRF).

We believe that the better performance of MIC-HCRF for six gestures (#4, #5, #10, #11, #18, and #19) came from its ability to separate multiple channels and capture the *intra*-dependencies, the dependencies within each separate channel. Figures 6-9, 6-10, and 6-11 show the patterns of body and hand pose signals for the six gestures. In these gestures, the hand pose signals were not significant (values of the probability estimates were low); it was

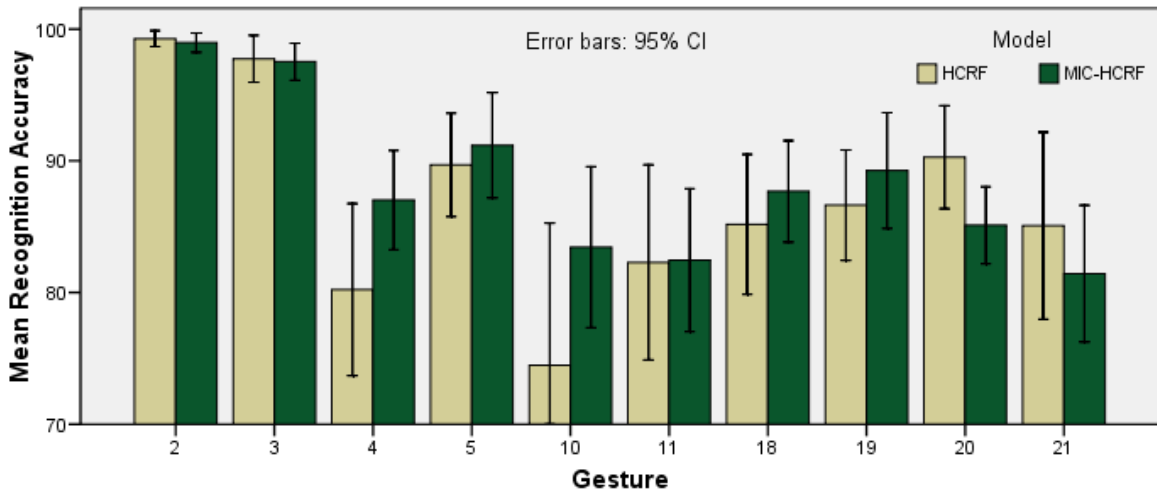


Figure 6-13: Recognition accuracy rates of HCRF ($|\mathcal{H}|=5$) and MIC-HCRF ($|\mathcal{H}^1|=3$, $|\mathcal{H}^2|=2$) when the dPS feature combination was used.

Gesture	HCRF ($ \mathcal{H} =5$)	MIC-HCRF ($ \mathcal{H}^1 =3$, $ \mathcal{H}^2 =2$)	Difference
#2	99.27 (1.05)	98.97 (1.28)	-0.30
#3	97.74 (3.08)	97.52 (2.44)	-0.22
#4	80.22 (11.31)	87.02 (6.54)	+6.80
#5	89.69 (6.79)	91.18 (6.92)	+1.49
#10	74.47 (18.71)	83.44 (10.59)	+8.97
#11	82.28 (12.84)	82.46 (9.38)	+0.18
#18	85.18 (9.19)	87.68 (6.67)	+2.50
#19	86.63 (7.25)	89.26 (7.61)	+2.63
#20	90.27 (6.77)	85.11 (5.06)	-5.16
#21	87.07 (12.30)	81.43 (8.99)	-5.64
Average	87.08 (12.19)	88.41 (8.95)	+1.33

Table 6.4: Recognition accuracy rates of HCRF ($|\mathcal{H}|=5$) and MIC-HCRF ($|\mathcal{H}^1|=3$, $|\mathcal{H}^2|=2$) when the dPS feature combination was used.

only the body pose information that could differentiate the gestures. In this case, we would expect that a mixture of body and hand signals might confuse a model learning patterns of these gestures, because the hand poses do not provide helpful information while making the dimensionality of the signal higher. Note that, the two signals are mixed in HCRF, but separated in MIC-HCRF. Therefore, we can see that the ability to separate multiple channels helped MIC-HCRF capture the subtle intra-dependencies within the body pose channel successfully.

Four other gestures (#2, #3, #20, and #21) are the ones that involve a mixture of body and hand poses; that is, hand pose played an important role defining the meanings of these gestures. In this case, we would expect that the model should capture the *inter*-dependencies (i.e., dependencies among multiple channels) in order to learn the patterns of these gestures. Therefore, the fact that MIC-HCRF performed less successfully for the two gestures (#20 and #21) might imply that MIC-HCRF is not good at capturing *inter*-dependencies (i.e., dependencies among multiple channels).

However, we claim that there might be another reason for this: the current configuration of MIC-HCRF was inappropriate. By comparing patterns of body and hand pose signals in Figures 6-8 and 6-12, we can see that for two pairs of gestures, #2 & #3 and #20 & #21, body pose information cannot help in differentiating between the gestures in the pair, and we need hand pose information in order to differentiate them. In the figures, there was only one line of hand pose signal in gesture #2 (ThumbUp-Right) and in gesture #3 (ThumbDown-Right): MIC-HCRF performed on par for these two gestures, which proves that the model learned the inter-dependencies between body and hand pose channels successfully.

On the other hand, there were multiple lines of hand pose signals in gestures #20 and #21, and MIC-HCRF performed less successfully for these two gestures. This might be simply because the configuration of MIC-HCRF in this experiment was sub-optimal, i.e., the number of hidden states or the optimization termination scheme. It is also possible that there were not enough examples in the training dataset to learn the optimal parameter

values describing the relationship between a gesture class label and hidden states. To understand this, remember that MIC-HCRF captures the dependencies between a class label and hidden states of two time-consecutive steps by creating combinatorial hidden state variables $\mathbf{h} \in (\mathcal{H}^{(1)} \times \mathcal{H}^{(2)})$ (see Eq. 6.9 in Section 6.3.2). This increases the required number of parameters to be optimized to the order of $\mathcal{O}(n^2)$, which in turn increases the required number of training samples for a successful recognition.

Note that, MIC-HCRF is still in its preliminary stage: we still have to explore different configurations of MIC-HCRF (i.e., the number of hidden states for each channel) or different construction of the underlying graphical model (i.e., different ways of capturing dependencies among channels). However, one of the advantages of MIC-HCRF is its ability to specify dependencies in the graphical model more precisely, which allows us to explore many more options. We anticipate this as an interesting direction for future work.

6.5 Related Work

Gesture recognition is a broad term that is increasingly used in natural human-computer interaction research. The meanings of gestures can range from dynamic human body motion [4][73] through pointing device gestures [11][42] to sign languages [66]. For more comprehensive review of gestures and gesture recognition, see survey papers [6][27][48].

In this work, we are primarily concerned with multi-signal gestures involving dynamic body movements and static hand pose configuration. Of particular interest is designing and implementing a statistical inference framework for multi-signal gesture recognition.

We review some of the recent research efforts aimed at exploring multi-signal gesture recognition.

In [7], Althoff *et al.* used trajectories of head and hand movement to recognize gestures for in-car control systems, where the trajectories were tracked using a near infrared camera.

They implemented two different versions of gesture recognizers, a rule-based and a HMM-based, and evaluated the two different approaches, showing similar recognition accuracy rates (90%) with 5 gestures (left, right, forward, backward, and wipe). However, their evaluation included recognition results obtained from each individual recognizer only, not testing how a combination of the features would perform for multi-signal gestures.

In [41], Li *et al.* designed and implemented a multi-signal gesture recognition system in a human-robot interaction scenario, using a combination of head orientation, body pose, and hand pose information. Three cameras were mounted on a mobile robot: a stereo camera tracked the orientation of the head, and a web camera and a time-of-flight camera tracked the body and hand. Head orientation was determined by tracking eye-gaze; body pose was estimated using a particle filter; and hand pose was classified using a multi-resolution image querying method. Their system was evaluated on the pointing gesture task, showing that their approach performed more reliably than other methods (i.e., a vector from head to hand). However, their scenario included static gestures only (i.e., tracking where a human is pointing at), a task too simple to explore the complex nature of multi-signal gestures.

In [15], Castellano *et al.* presented a multi-signal approach for human emotion recognition, using various features including facial expression, body movement, and speech. Two information fusion methods were compared: *feature-level fusion* method performed recognition based on one single feature vector combining all three features; *decision-level fusion* method performed recognition with each individual features, then the final decision was made based on the three recognition results (i.e., majority voting or best probability). They showed that the feature-level fusion (78.3%) performed better than the decision-level fusion (74.6%). However, their experiment did not include a condition for a statistical framework that considers multi-signal inputs, as was introduced in this work.

Chapter 7

Conclusion

In this thesis work, we designed and implemented a multi-signal gesture recognition system that uses a combination of information obtained from 3D body pose estimation and hand pose classification.

A stereo camera was used to capture pairs of time-synchronized images, calculating depth maps using existing stereo vision techniques. The images were also background subtracted using a combination of the codebook approach and depth information.

For 3D body pose estimation, we constructed a parametric model of the human upper body to generate 3D body poses. This model was then fitted to an input image by comparing several features extracted from the parametric model and the input image. Finally, pose estimation was performed using a particle filter.

For hand pose classification, we first defined a vocabulary of canonical hand poses that included opened palm, closed palm, thumb up, and thumb down. A multi-class SVM classifier was trained on a data set containing HOG features extracted from manually segmented images of hands. Hand pose classification was performed by searching for hands in the image around the wrist positions obtained from body pose estimation, then classifying them using the trained SVM classifier.

Finally, for gesture recognition, we developed the technique of MIC-HCRF to process multi-signal input data, as an extension from the previous work on HCRF, so that dependencies between body and hand poses can be modeled in a more principled manner.

The system was evaluated on a real-world scenario: we tested the performance of our gesture recognition system with a subset of the NATOPS aircraft handling signals [53], a challenging gesture vocabulary that involves both body and hand pose articulations. We showed that combining body and hand poses significantly improved the gesture recognition accuracy. We also showed what types of body and hand pose features performed the best: for the body pose, the derivatives of joint coordinates was the best feature; for the hand pose, a vector of probability estimates was the best feature. Lastly, we showed that MIC-HCRF captured complex dependencies within multiple channels successfully, achieving the overall recognition accuracy rate 88.41%, which is comparable to that of human pilots.

7.1 Summary of Contribution

There are three main contributions of this work:

- We designed and implemented a gesture recognition system that attends to both body and hand poses, and analyzed it, showing that using the combined information significantly improved the performance of gesture recognition.
- We extended the previous work on HCRF to deal with multi-signal input vectors, allowing the framework to decide how to handle complex dependencies among different input modalities, as opposed to assuming the input modalities are independent and forcing them to form a single feature vector.
- We applied the gesture recognition system to an interesting real-world problem, an aircraft carrier flight deck environment, and tested the performance of the system on a subset of aircraft handling gestures that are currently used in the domain.

7.2 Future Work

There are many unexplored topics in this work that might be interesting future work.

7.2.1 Refining Body Pose Estimation And Hand Pose Classification Using Results From One Another

One interesting topic would be using hand pose classification to improve body pose estimation, and vice versa.

Body pose estimation can benefit from hand pose classification. A hand pose classifier usually gives relatively more accurate wrist positions than a body pose estimator. Thus the result from body pose estimation can be refined after hand pose classification is performed by solving, for example, the inverse kinematics problem and optimizing parameter values.

Hand pose classification can also benefit from body pose estimation. In this work we showed one possible way of doing this: searching for hands in the image around estimated wrist positions only. Another possibility would be to using the direction of a lower arm (i.e, a vector from the elbow to the wrist) to constrain the possible orientation range of the hand, allowing the classifier to narrow down the search space. This could also help us to have a rotation-invariant hand pose classifier.

7.2.2 Real-Time Gesture Recognition System

Our system does not work in real-time, one of the main limitations of this work. A major portion of the processing time comes from body pose estimation, evaluating the likelihood function for each particle, and hand pose classification, computing HOG features from the image.

Some body poses that are easy to estimate (e.g., simple 2D body pose without self-

occlusion) require a small number of particles, while other cases might require more particles. Therefore, the processing time of body pose estimation could be optimized by maintaining the number of particles adaptively, that is, changing the particle set size based on some measure of a confidence score (e.g, the average of fitting errors from all particles).

It is also possible to reduce the processing time of hand pose classification. Computing HOG features from the image involves sliding a window and computing features repeatedly over the overlapping area, which is quite inefficient. Viola *et al.* introduced the integral image [71], an intermediate representation of the image with precomputed values that allows us to compute images features efficiently. We could extend the integral image to work for computing HOG features as well, thereby improving the speed performance of hand pose classification.

Another way to reduce the processing time would be to use the recent optimization techniques such as CUDA (Compute Unified Device Architecture) [54], a computing engine in graphics processing units (GPUs) specialized for parallel computation of concurrent threads, Intel Math Kernel Library [32], a library of optimized math routines, or OpenMP [55], a shared memory multiprocessing programming architectures.

7.2.3 Using Context Information

Our gesture recognition system assumes no background knowledge or context information; it performs the recognition solely based on the observation. However, humans make extensive use of these to perform many types of inference tasks, allowing them to make decisions rapidly or to come up with an answer that is impossible to infer solely based on the observation (e.g., see [45]). This intuition has led to a large body of research that concerns modeling and integrating human knowledge into the system (for high-level overview of the concept, see [44] and [46]).

Using this context information might improve the performance of our system in many ways.

For gesture recognition in the context of the flight deck environment, there is a certain sequence of customary gestures, and we can reason about what gestures would make sense based on the context. For example, it is less likely that an ABH will command "brakes off" to a pilot while taxiing an aircraft, because the brake should have been already off while moving. Therefore, we believe that a gesture recognition system that is able to consider the context information (i.e., reason about routine procedures on the flight deck) can improve its performance.

Similarly, in each gesture there are a certain set of body and hand poses we can expect, so once we know which gesture is being performed or what gestures we can expect next, we can narrow down the search space, which can improve the performance of a body pose estimator and a hand pose classifier.

One interesting question would be, how could we incorporate the context information into the current framework? To do so, we need an approach to describing the knowledge in a concise and consistent manner. Also, there needs to be an easy way to add or modify the knowledge manually, or even a way to reason about new things automatically based on the existing knowledge. It is important that the whole process should not be ad-hoc, i.e., it should be generalizable to many types of context information.

Designing and implementing a statistical estimation and inference framework that incorporates the context information is also an interesting future direction of this work. Most of the current approaches concentrate on learning from observations. Although we started to see some early approaches (e.g., [56]), this has yet to be fully explored.

7.2.4 Providing Feedback To Human

As we mentioned briefly in Chapter 2, allowing two-way communication between humans and systems is of particular interest in this work. In order to provide natural gesture-based interaction, it is important for a system to be able to recognize human gestures. At the

same time, it is also necessary for the system to have an appropriate feedback mechanism, that is, the system have to be able to gesture back, just as a human would do in the same situation.

In the aircraft carrier flight deck environment, deck personnel and pilots are required to have eye contact when communicating; pilots are in fact not permitted to take any action without having eye contact with one of the deck personnel confirming the action. This prevents misinterpretation of a command and aids in correcting mistakes immediately, keeping the environment safe. This makes feedback in communication indispensable to this safety- and mission-critical domain.

There are still many questions to be answered. What does it mean for a system to gesture? How can we define a natural feedback mechanism, or how natural a system's feedback should be? We leave all these questions to be answered for the future work.

Appendix A

NATOPS Gesture Dataset

The Naval Air Training and Operating Procedures Standardization (NATOPS) manual standardizes general flight and operating procedures for the US naval aircraft. There are a large number of publications that belong to NATOPS; among the many, an aircraft signals manual (NAVAIR 00-80T-113 [53]) contains information about all aircraft systems including general aircraft and carrier flight deck handling signals.

For this work, we selected 24 general aircraft handling signals that we believe well represent the intricacy of the entire gesture set. A Bumblebee2 stereo camera from Point Grey Research Inc. was used to collect a dataset, producing 320 x 240 pixel resolution images at 20 FPS¹. Twenty participants performed the 24 gestures, repeating each gesture 20 times.

In this appendix we provide information on the 24 aircraft handling signals as described in [53].

¹The camera used in this work is BB2-03S2C-38; up to 640x480 resolution, two 3.8mm color lenses with 12cm baseline, 48 FPS, 65-degree HFOV.





SIGNAL	DAY	NIGHT	REMARKS
<p>#1</p>  <p>ACR-F01a</p> <p>I HAVE COMMAND</p>	<p>Hold one hand open, motionless and high above head, with palm forward.</p>	<p>Same as day except with wand.</p>	
<p>#2</p>  <p>ACR-F01a</p> <p>AFFIRMATIVE (ALL CLEAR)</p>	<p>Hand raised, thumb up.</p>	<p>Same as day signal with addition of wands.</p>	<p>Conforms to ICAO signal.</p>
<p>#3</p>  <p>ACR-F01b</p> <p>NEGATIVE (NOT CLEAR)</p>	<p>Arm held out, hand below waist level, thumb turned downwards.</p>	<p>Same as day signal with addition of wands.</p>	
<p>#4</p>  <p>ACR-F01b</p> <p>SPREAD WINGS/ HELICOPTER BLADES</p>	<p>Arms hugged around shoulders, then swept straight out to the sides.</p>	<p>Same as day signal with addition of wands.</p>	

Figure A-1: General Aircraft Handling Signals (Sheet 1 of 6).





SIGNAL	DAY	NIGHT	REMARKS
<p>#5</p>  <p>ACR-F01aa</p> <p>FOLD WINGS/ HELICOPTER BLADES</p>	<p>Arms straight out at sides, then swept forward and hugged around shoulders.</p>	<p>Same as signal with addition of wands.</p>	
<p>#6</p>  <p>ACR-F01ac</p> <p>LOCK WINGS/ HELICOPTER BLADES</p>	<p>Hit right elbow with palm of left hand.</p>	<p>Same as day signal with addition of wands.</p>	
<p>#7</p>  <p>ACR-F01an</p> <p>UP HOOK</p>	<p>Right fist, thumb extended upward, raised suddenly to meet horizontal palm of left hand.</p>	<p>Same as day signal with addition of wands.</p>	
<p>#8</p>  <p>ACR-F01al</p> <p>DOWN HOOK</p>	<p>Right fist, thumb extended downward, lowered suddenly to meet horizontal palm of left hand.</p>	<p>Same as day signal with addition of wands.</p>	

Figure A-2: General Aircraft Handling Signals (Sheet 2 of 6).





SIGNAL	DAY	NIGHT	REMARKS
<p>#9</p>  <p>ACR-F010q</p> <p>REMOVE TIEDOWNS (director)</p>	<p>To tiedown crew: Makes wiping motion down left arm with right hand.</p>	<p>Same as day except with wands.</p>	
<p>#10</p>  <p>ACR-F01q</p> <p>REMOVE CHOCKS</p>	<p>Arms down, fists closed, thumbs extended outwards, swing arms outwards.</p>	<p>Same as day signal with addition of wands.</p>	<p>Conforms to ICAO signal.</p>
<p>#11</p>  <p>ACR-F01p</p> <p>INSERT CHOCKS</p>	<p>Arms down, fists closed, thumbs extended inwards, swing arms from extended position inwards.</p>	<p>Same as for day signal with addition of wands.</p>	<p>Conforms to ICAO signal.</p>
<p>#12</p>  <p>ACR-F01h</p> <p>MOVE AHEAD</p>	<p>Arms extended from body and held horizontal to shoulders with hands up-raised and above eye level, palms facing backwards. Execute beckoning arm motion angled backward. Rapidity indicates speed desired of aircraft.</p>	<p>Same as day signal with addition of wands.</p>	

Figure A-3: General Aircraft Handling Signals (Sheet 3 of 6).




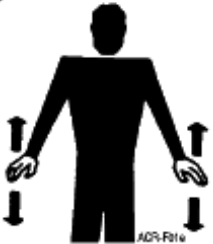
SIGNAL	DAY	NIGHT	REMARKS
<p>#13</p>  <p>ACR-F011</p> <p>TURN TO LEFT</p>	<p>Extend right arm horizontally, left arm is repeatedly moved upward. Speed of arm movement indicating rate of turn.</p>	<p>Same as day signal with addition of wands.</p>	<ol style="list-style-type: none"> 1. Clench first (day), or down-turned wand (night), means for pilot to lock indicated brake. 2. Also used for spot turns airborne aircraft. Conform to ICAO Signal.
<p>#14</p>  <p>ACR-F01g</p> <p>TURN TO RIGHT</p>	<p>Extend left arm horizontally, left arm is repeatedly moved upward. Speed of arm movement indicating rate of turn.</p>	<p>Same as day signal with addition of wands.</p>	<ol style="list-style-type: none"> 1. Clench fist (day), or down-turned wand (night), means for pilot to lock indicated brake. 2. Also used for sport turns airborne aircraft. Conform to ICAO Signal.
<p>#15</p>  <p>ACR-F01c</p> <p>PROCEED TO NEXT MARSHALER</p>	<p>Right or left arm Down, other arm moved across the body and extended to indicate direction to next marshal.</p>	<p>Same as day signal with addition of wands.</p>	<p>Conforms to ICAO signals.</p>
<p>#16</p>  <p>ACR-F01e</p> <p>SLOW DOWN</p>	<p>Arms down with palms towards ground, then moved up and down several times.</p>	<p>Same as day signal with addition of wands.</p>	<p>Conforms to ICAO signal.</p>

Figure A-4: General Aircraft Handling Signals (Sheet 4 of 6).





SIGNAL	DAY	NIGHT	REMARKS
<p>#17</p>  <p>ACR-F011</p> <p>STOP</p>	<p>Arms crossed above the head, palms facing forward.</p>	<p>Same as day signal with addition of wands.</p>	
<p>#18</p>  <p>ACR-F016h</p> <p>ENGAGE NOSEGEAR STEERING</p>	<p>Point to nose with index finger while indicating direction of turn with other index finger.</p>	<p>Same as day signal with addition of wands.</p>	
<p>#19</p>  <p>ACR-F016u</p> <p>HOT BRAKES</p>	<p>Makes rapid fanning motion with one hand in front of face and points to wheel with other hand.</p>	<p>Same as day except with wands.</p>	
<p>#20, #21</p>  <p>ACR-F017</p> <p>BRAKES</p>	<p>ON — Arms above head, open palms and fingers raised with palms toward aircraft, then fist closed.</p> <p>OFF — Reverse of above.</p>	<p>ON — Arms above head, then wands crossed.</p> <p>OFF — Crossed wands, then uncrossed.</p>	

Figure A-5: General Aircraft Handling Signals (Sheet 5 of 6).


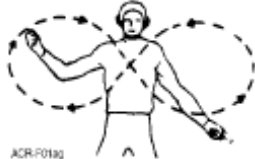

SIGNAL	DAY	NIGHT	REMARKS
<p>#22</p>  <p>ACR-F010r</p> <p>INSTALL TIEDOWNS (director)</p>	<p>To tiedown crew: Rotates hands in a circle perpendicular to and in front of his body.</p>	<p>Same as day except with wands.</p>	
<p>#23</p>  <p>ACR-F013g</p> <p>FIRE</p>	<p>Describes large figure eight with one hand and points to the fire area with the other hand.</p>	<p>Same except with wands.</p>	<p>Signal is meant for information only. Pilot should be given a cut engine or continuous turnup signal, as appropriate.</p>
<p>#24</p>  <p>ACR-F01e</p> <p>CUT ENGINE(S)</p>	<p>Either arm and hand level with shoulder, hand moving across the throat, palm down. Hand is moved sideways, arm remaining bent. Other arm pointing to engine.</p>	<p>Same as day signal with addition of wands.</p>	

Figure A-6: General Aircraft Handling Signals (Sheet 6 of 6).

Bibliography

- [1] *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, 16-22 June 2003, Madison, WI, USA. IEEE Computer Society, 2003.
- [2] *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, 17-22 June 2006, New York, NY, USA. IEEE Computer Society, 2006.
- [3] *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, 20-25 June 2009, Miami, Florida, USA. IEEE, 2009.
- [4] Yang Wang 0003 and Greg Mori. Max-margin hidden conditional random fields for human action recognition. In *CVPR* [3], pages 872–879.
- [5] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(1):44–58, 2006.
- [6] Jake K. Aggarwal and Quin Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.
- [7] F. Althoff, R. Lindl, and L. Walchshaeusl. Robust multimodal hand- and head gesture recognition for controlling automotive infotainment systems. In *VDI-Tagung: Der Fahrer im 21. Jahrhundert*, Braunschweig, Germany, November 22-23 2005.
- [8] Alan H. Barr. Superquadrics and angle-preserving transformations. *IEEE Comput. Graph. Appl.*, 1(1):11–23, 1981.

- [9] Aaron F. Bobick and James W. Davis. *Motion-Based Recognition - Computational Imaging and Vision*. Kluwer Academic Publishers, 1997.
- [10] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. Wiley, 2001.
- [11] Andrew Bragdon, Robert C. Zeleznik, Brian Williamson, Timothy Miller, and Joseph J. LaViola Jr. Gesturebar: improving the approachability of gesture-based interfaces. In Dan R. Olsen Jr., Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott E. Hudson, and Saul Greenberg, editors, *CHI*, pages 2269–2278. ACM, 2009.
- [12] Matthew Brand. Shadow puppetry. In *ICCV*, pages 1237–1244, 1999.
- [13] Patrick Buehler, Andrew Zisserman, and Mark Everingham. Learning sign language by watching tv (using weakly aligned subtitles). In *CVPR* [3], pages 2961–2968.
- [14] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov. 1986.
- [15] Ginevra Castellano, Loïc Kessous, and George Caridakis. Emotion recognition through multiple modalities: Face, body gesture, speech. In Christian Peter and Russell Beale, editors, *Affect and Emotion in Human-Computer Interaction*, volume 4868 of *Lecture Notes in Computer Science*, pages 92–103. Springer, 2008.
- [16] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.
- [17] Filiberto Chiabrando, Roberto Chiabrando, Dario Piatti, and Fulvio Rinaudo. Sensors for 3d imaging: Metric evaluation and calibration of a ccd/cmos time-of-flight camera. *Sensors*, 9(12):10080–10096, 2009.
- [18] Andrea Colombari, Andrea Fusiello, and Vittorio Murino. Video objects segmentation by robust background modeling. In Rita Cucchiara, editor, *ICIAP*, pages 155–164. IEEE Computer Society, 2007.

- [19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR (1)*, pages 886–893. IEEE Computer Society, 2005.
- [20] David Demirdjian and Trevor Darrell. 3-D articulated pose tracking for untethered diectic reference. In *ICMI*, pages 267–272. IEEE Computer Society, 2002.
- [21] Jaques Denavit and Richard S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans ASME J. Appl. Mech*, 23:215–221, 1955.
- [22] Jonathan Deutscher, Andrew Blake, and Ian D. Reid. Articulated body motion capture by annealed particle filtering. In *CVPR*, pages 2126–2133. IEEE Computer Society, 2000.
- [23] Joseph F. Dyro. *Clinical Engineering Handbook*. Materials & Mechanical. Academic Press, 2004.
- [24] AleErkan Engin. On the biomechanics of the shoulder complex. *Journal of Biomechanics*, 13(7):575–581, 583–590, 1980.
- [25] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, 2007.
- [26] Xuemei Feng, Jingzhou Yang, and Karim Abdel-Malek. Survey of biomechanical models for the human shoulder complex. Technical report, SAE International, 2008.
- [27] Darius Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.
- [28] Salih Burak Göktürk and Carlo Tomasi. 3d head tracking based on recognition and interpolation using a time-of-flight depth sensor. In *CVPR (2)*, pages 211–217, 2004.
- [29] Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell. A bayesian approach to image-based visual hull reconstruction. In *CVPR (1) [1]*, pages 187–194.

- [30] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, mar 2002.
- [31] Point Grey Research Inc. *Bumblebee 2 Stereo Vision Camera*.
- [32] Intel. *Intel Math Kernel Library*.
- [33] Michael Isard and Andrew Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.
- [34] Kyungnam Kim, Thanarat H. Chalidabhongse, David Harwood, and Larry S. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185, 2005.
- [35] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neuro-computing: Algorithms, Architectures and Applications*. Springer-Verlag, 1990.
- [36] Mathias Kölsch and Matthew Turk. Robust hand detection. In *FGR*, pages 614–619, 2004.
- [37] Nils Krahnstoeber, Mohammed Yeasin, and Rajeev Sharma. Automatic acquisition and initialization of articulated models. *Mach. Vis. Appl.*, 14(4):218–228, 2003.
- [38] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *ICML*, pages 282–289. Morgan Kaufmann, 2001.
- [39] R. Lange and P. Seitz. Solid-state time-of-flight range camera. *Quantum Electronics, IEEE Journal of*, 37(3):390–397, mar. 2001.
- [40] Mun Wai Lee and Isaac Cohen. A model-based approach for estimating human 3D poses in static images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(6):905–916, 2006.

- [41] Zhi Li and R. Jarvis. A multi-modal gesture recognition system in a human-robot interaction scenario. pages 41 –46, nov. 2009.
- [42] Chunyuan Liao, François Guimbretière, Ken Hinckley, and James D. Hollan. Papiercraft: A gesture-based command system for interactive paper. *ACM Trans. Comput.-Hum. Interact.*, 14(4), 2008.
- [43] John MacCormick and Michael Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In David Vernon, editor, *ECCV (2)*, volume 1843 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2000.
- [44] John McCarthy. Deep issues: phenomenal data mining. *Commun. ACM*, 43(8):75–79, 2000.
- [45] John McCarthy. Challenges to machine learning: Relations between reality and appearance. In Stephen Muggleton, Ramón P. Otero, and Alireza Tamaddon-Nezhad, editors, *ILP*, volume 4455 of *Lecture Notes in Computer Science*, pages 2–9. Springer, 2006.
- [46] Marvin Minsky. Deep issues: commonsense-based interfaces. *Commun. ACM*, 43(8):66–73, 2000.
- [47] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [48] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS - PART C*, 37(3):311–324, 2007.
- [49] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.
- [50] Thomas B. Moeslund, Adrian Hilton, and Volker Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, 2006.

- [51] Greg Mori and Jitendra Malik. Recovering 3d human body configurations using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(7):1052–1062, 2006.
- [52] NASA. *Man-Systems Integration Standards: Volume 1. Section 3. Anthropometry And Biomechanics*, 1995.
- [53] U.S. Navy. *Aircraft Signals NATOPS Manual, NAVAIR 00-80T-113*. Washington, DC, 1997.
- [54] NVIDIA. *CUDA*.
- [55] OpenMP. *The OpenMP API Specification For Paraller Programming*.
- [56] Tom Ouyang and Randall Davis. Learning from neighboring strokes: Combining appearance and context for multi-domain sketch recognition. In *Advances in Neural Information Processing Systems 22*, pages 1401–1409, 2009.
- [57] J. Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, pages 133–136, Pittsburgh, PA, 1982.
- [58] Ronald Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1-2):4–18, 2007.
- [59] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *NIPS*, 2004.
- [60] Lawrence R. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.
- [61] A. Rochas, M. Gosch, A. Serov, P.A. Besse, R.S. Popovic, T. Lasser, and R. Rigler. First fully integrated 2-d array of single-photon detectors in standard cmos technology. *Photonics Technology Letters, IEEE*, 15(7):963–965, jul. 2003.

- [62] Gregory Shakhnarovich, Paul A. Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750–759. IEEE Computer Society, 2003.
- [63] R. Sharma, M. Zeller, V.I. Pavlovic, T.S. Huang, Z. Lo, S. Chu, Y. Zhao, J.C. Phillips, and K. Schulten. Speech/gesture interface to a visual-computing environment. *Computer Graphics and Applications, IEEE*, 20(2):29–37, mar. 2000.
- [64] Cristian Sminchisescu and Bill Triggs. Kinematic jump processes for monocular 3D human tracking. In *CVPR (1)* [1], pages 69–76.
- [65] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. Robot modeling and control. John Wiley and Sons, Inc., 2005. Chap3. Forward and Inverse Kinematics.
- [66] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1371–1375, 1998.
- [67] Leonid Taycher, David Demirdjian, Trevor Darrell, and Gregory Shakhnarovich. Conditional random people: Tracking humans with crfs and grid filters. In *CVPR (1)* [2], pages 222–229.
- [68] Alvin R. Tilley and Henry Dreyfuss Associates. *The Measure of Man and Woman: Human Factors in Design*. O’Reilly Media, 2008.
- [69] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.
- [70] Vicon. *Vicon Motion Capture System*.
- [71] Paul A. Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
- [72] S. Wachter and H.-H. Nagel. Tracking persons in monocular image sequences. *Comput. Vis. Image Underst.*, 74(3):174–192, 1999.

- [73] Sy Bor Wang, Ariadna Quattoni, Louis-Philippe Morency, David Demirdjian, and Trevor Darrell. Hidden conditional random fields for gesture recognition. In *CVPR (2)* [2], pages 1521–1527.