UNIVERSAL MOTION-BASED CONTROL AND MOTION RECOGNITION

A Thesis
Presented to
The Academic Faculty

by

Mingyu Chen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2013

UNIVERSAL MOTION-BASED CONTROL AND MOTION RECOGNITION

Approved by:

Professor Biing-Hwang (Fred) Juang
and Professor Ghassan AlRegib,
Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Linda Wills
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Edward Coyle
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Nagi Gebraeel
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Professor Ayanna Howard
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Date Approved: August 23rd, 2013

To my family, friends, and loved ones.

ACKNOWLEDGEMENTS

The research journey is the most challenging and valuable experience I have ever had in my life. This dissertation is the milestone that marks all my efforts and achievements along the research journey. The friendships I have made during these years at Georgia Tech are equally important as well.

I would like to express my sincerest respect and gratitude towards my advisors, Professor Biing-Hwang Juang and Professor Ghassan AlRegib. I am grateful for their insights and advices to guide me through research problems. I also appreciate their patience to allow me to search and work on research topics that I am interested in. Their teaching of how to write and present in a scholarly manner is very beneficial. I am lucky to receive double the amount of training under the expertise of Professor Juang and Professor AlRegib.

My special thanks go out to the professors who share their precious time to be on my dissertation committee: Professor Edward Coyle, Professor Nagi Gebraeel, Professor Ayanna Howard, and Professor Linda Wills. I do appreciate their valuable feedback on my research. I also would like to thank those with the ECE department who are always kind to offer help: Pat Dixon, Diana Fouts, Jennifer Lunsford, Christopher Malbrue, Tammy Scott, Stacie Speight, Daniela Staiculescu, and Tasha Torrence.

Dring my stay at Georgia Tech, I have been fortunate to have the company of a group of great friends. It has been a pleasure to have Mohammed Aabed, Umair Altaf, Solh Mashhour, Sunghwan Shin, Fred Stakem, Dogancan Temel, Chao Weng, Jason Wung, Wenhui Xu, Yong Zhao, and Wang Zhen as friends. I am also grateful to all my Taiwanese friends that are too many to name for the joyful moments we

have shared together. Particularly, I owe someone special my deepest apology for my immaturity and appreciation for their precious time and love.

Finally but not least, I would like to thank my parents Su-Miao and Ping-Hsiang for their endless love and support since the day I was born. I also want to thank my sister Yi-Jen and my brother Hungli for their encouragement and company. My family makes me who I am today. Without their support, this work would not have been completed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this dissertation, we propose a universal motion-based control framework that supports general functionalities on 2D and 3D user interfaces with a single integrated design. The user interacts with the system mainly through control motions freely rendered in the air. Our design of motion-based control allows natural interactions that are conceptually similar to the user's real-world experience. We develop a hybrid framework of optical and inertial sensing technologies to track 6-DOF (degrees of freedom) motion of a handheld device, which includes the explicit 6-DOF (position and orientation in the global coordinates) and the implicit 6-DOF (acceleration and angular speed in the device-wise coordinates). On a 2D user interface, the handheld device functions as a virtual mouse in the air for interactions such as pointing-and-clicking, dragging, and scrolling. On a 3D user interface, the framework supports directly 3D object manipulation (translation and rotation) and 3D scene navigation and browsing. Motion recognition is another key function of the universal motion-based control and contains two parts: motion gesture recognition and air-handwriting recognition. The interaction technique of each task is carefully designed to follow a consistent mental model and ensure the usability. The universal motion-based control achieves seamless integration of 2D and 3D interactions, motion gestures, and air-handwriting.

In addition to the aspect of user interface design, motion recognition by itself is a challenging problem. For motion gesture recognition, two approaches are proposed: a statistical feature-based linear classifier and a hidden Markov model (HMM)-based classifier. We also propose a normalization procedure to effectively address the large in-class motion variations among users. The main contribution is the investigation of

the relative effectiveness of various feature dimensions (of tracking signals) for motion gesture recognition in both user-dependent and user-independent cases.

For air-handwriting recognition, we first develop a strategy to model air-handwriting with basic elements of characters and ligatures. Then, we build word-based and letter-based decoding word networks for air-handwriting recognition. The word-based network fully utilizes the vocabulary and is robust to character (sub-word) errors. However, word-based recognition cannot handle out-of-vocabulary words. In contrary, the letter-based network allows an arbitrary decoding letter sequence with the trade-off of decrease in recognition accuracy. Thus, we propose to utilize a restrictive bigram language model and n-best decoding to further improve the letter-based recognition performance. Moreover, we investigate the detection and recognition of air-fingerwriting as an extension to air-handwriting. A window-based approach is proposed to automatically detect writing events and segment the writing part from the tracking signals. We evaluate the recognition performance of the detected segments and compare with the recognition result of the ground-truth segments. To complete the evaluation of air-handwriting, we conduct usability study to support that air-handwriting is suitable for text input on a motion-based user interface.

CHAPTER I

INTRODUCTION

1.1 Motivations

The coming of the digital age has resulted in a revolution in the human consumption of information. As a result, digital convergence is taking place in many shapes and forms. Today, many devices and systems in the marketplace offer an array of digital information handling capabilities to allow a user to enjoy different digital services and consume information with ease. For example, an HD monitor can be used as a television or as a display for a personal computer. A smartphone allows the user to make phone calls, watch video clips, browse webpages, play games, and use all sorts of applications. The mode of digital information consumption is elevated to a new level with the improvement of new technologies, such as the touchscreen and motion sensors. As the mode of access and use of digital information becomes more diverse, the need of an effective user interface has become even more critical. An effective user interface design together with a proper control device can reduce the complexity of these usage modes. The combination of the widget-based user interface and the touchscreen is a good example.

Before the era of digital convergence, a device is usually task or service specific, and its user interface is designed to match the functionality the device intends to support. For example, a remote control typically has an array of buttons, each of which is associated with a command or action for the target system to perform. A keyboard allows the user to send alphanumerical symbols to the system for text input or commands. It is obviously tied with the conventional use of a computer,

particularly to satisfy the need in word processing. A mouse supports the point-and-click operation, which is extensively used in the graphic user interface (GUI). Another prevalent user interface design is the telephone keypad, which continues its tradition from a plain old telephone (POT) to the cellular handset. As data gets into a mobile phone, the need of a keyboard starts to change the form factor of a cellular device. A keypad with ten digits and a few special characters is no longer sufficient. A keyboard, whether physical or virtual, becomes commonly equipped in a smartphone. The touch-based pointing device is widely used nowadays and tightly bound to the GUI. The multi-touch technology further broadens the gesture dictionary to enable more control actions and functions. The touchscreen technology is a perfect example of combining the display and single or multi-touch tracking to build a very efficient control interface. The strongly connected input and output, control, and feedback make it very intuitive to use, particularly for novice users. Touchscreen works very well on handheld devices or display panels within arms reach. When the display is fairly large or far from the user, the touch control becomes infeasible, and a new type of control interface is needed.

Thus, one particular unfulfilled demand for an effective user interface arises when the aforementioned digital convergence takes place in the living room where a user switches arbitrarily among various digital services, such as watching TV programs or movies, web browsing, teleconferencing, and gaming. To perform these tasks properly with the same system, one faces the challenge of an integrated user interface that supports the switch and button control as a remote, the point-and-click operation as a mouse, text input as a keyboard, and even more. These control functionalities, currently rendered through separate devices, have to be seamlessly integrated and effectively supported by a single device in the hand of the user. Moreover, the integrated user interface has to be intuitive to use.

1.2 Objectives and Contributions

The objective of this dissertation is to develop an integrated motion-based control framework that supports all general interface functionalities in a single design. Instead of pointing devices that are confined to planar movements, we propose a new motion-based control scheme that incorporates 3D motion tracking with minimum, untethered user-worn components. The platform tracks the position and orientation of the control device and operates on 6-DOF (degrees of freedom) motion data. Control motions can be rendered freely in the air and allows more intuitive interactions. The full spatial information enables the system to support natural interactions with both the conventional 2D user interface and the emerging 3D user interface. The proposed system also supports motion recognition for gestures and handwriting in the air, which provides a complementary modality for motion-based control.

We investigate an interface design methodology, formulate technical solutions to the realization of a universal motion control framework, and develop a working prototype. The integrated motion control is designed to be universal and handy to all digital services. Specifically, we focus on the following main topics in this dissertation.

1. System design of the universal motion-based control framework.
    - Accurate and untethered motion tracking.
    - Integration of motion-based interactions for general control functionalities.
2. Robust and accurate motion recognition.
    - Motion gesture recognition.
    - Air-handwriting detection and recognition.

1.2.1 Universal Motion-based Control

The system design of the universal motion-based control framework involves two major aspects: motion tracking and motion-based interaction. In Figure 1, we show

an overview of the universal motion-based control framework. A motion tracking system is essentially the input device for a motion-based user interface and also affects the affordable interaction techniques. To precisely track the translation and rotation of the control motion, a hybrid tracking framework of optical and inertial sensing is proposed. The tracking results contain both explicit 6-DOF (position and orientation in the global coordinates) and implicit 6-DOF (acceleration and angular speed in the device-wise coordinates). The proposed 6-DOF motion tracking system achieves one-to-one motion mapping of the handheld device, which is crucial for the design of motion-based control.

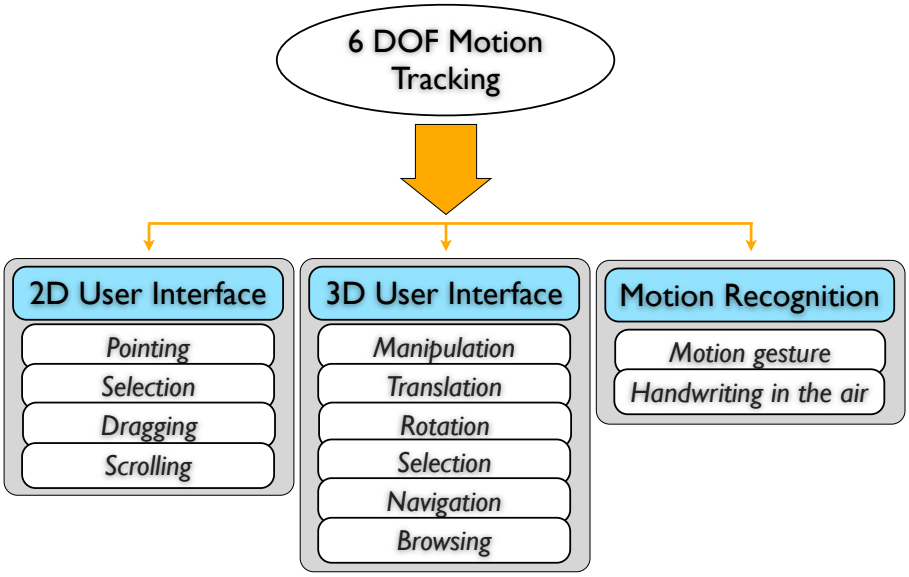


Figure 1: The overview of the universal motion-based control framework.

The supported motion-based control is categorized into three groups: *a*) 2D user interface, *b*) 3D user interface, and *c*) motion recognition as shown in Figure 1. Each group is organized to support task-specific interactions, and the linking of control motion to interactions is designed correspondingly. In this work, we are not going to invent new interaction metaphors since most of the design space for 3D interaction

techniques has been covered [10]. Taking into account all the general interactions required by the "universal user interface", we customize and design an integrated framework based on these well-known and widely-used techniques. The challenge is to integrate 2D/3D interactions and motion recognition seamlessly in the motion-based control framework. Our key contribution is to maximize the consistency over different interaction techniques while minimizing any potential degradation of performance. We also build a working prototype of universal motion-based control to demonstrate its usability.

1.2.2 Motion Recognition

In this dissertation, motion recognition contains two levels of sophistication: motion gesture recognition and air-handwriting recognition. Motion gestures can be convenient to trigger certain tasks or commands. Air-handwriting recognition allows the user to input text on a motion-based user interface and is especially useful when typing on a keyboard or writing on a touchpad is not available.

The goal of the proposed motion recognition is to produce a real-time recognition kernel that performs accurately and robustly across different users. With our 6-DOF motion tracking system, we represent a motion with a trajectory of 3D position, orientation, acceleration, and angular speed, which makes the recognition problem unconventional. Although most of gestures and all handwritings are defined by their 2D spatial trajectories, we successfully show that motion signals beyond a 2D trajectory are imformative to improve the accuracy and robustness of motion recognition. Compared to conventional 2D pointing devices, motions rendered freely in the air incur larger variations among individuals because people perform gestures or air-handwriting quite differently in scale, speed, and style. The large variations in motions make the recognition problem challenging. To address this problem, we

propose a normalization process to reduce the large in-class variations in motion signals and prove its effectiveness in improving the recognition performance, especially for the user independent case. Our main contribution is to investigate the relative effectiveness of various feature dimensions for motion recognition in both user dependent and user independent cases. To our best knowledge, we are the first to evaluate gesture and air-handwriting recognition with 6-DOF motions. The proposed motion recognition kernel is adaptable to different motion data from other motion tracking systems, and this study gives an insight into the attainable recognition rate.

Similar to motion gestures, air-handwriting involves no physical plane to write on and has no pen-up/pen-down information. In other words, air-handwriting is unistroke and different from ordinary handwriting. Therefore, conventional handwriting recognition techniques cannot be applied directly. We accomplish air-handwriting recognition by modeling a motion word with a composition of characters and ligature motions. In addition to the motion information, we also utilize the vocabulary and its language model to further improve the recognition performance.

Segmentation of the gesture or handwriting part from the continuous motion tracking data is another key issue for motion recognition. In our universal motion-based control, we adopt push-to-gesture and push-to-write for explicit delimitation of motion gestures and air-handwriting. The push-to-gesture/push-to-write scheme is straightforward when the tracking system requires a controller. In a controller-free system, delimiters in other forms can replace a button but makes the interaction less intuitive. It is preferable if the system can automatically detect the user's motion of a gesture or handwriting. Detection of a motion gesture is extremely difficult even for a human when the definition of the gesture set overlaps with common control motions, e.g., ambiguity in a swiping motion for a gesture or a cursor translation. On the other hand, detection of air-handwriting is feasible because the writing motion differs from

common control motions in general. As an extension work, we propose a novel approach for automatic detection of air-fingerwriting captured by a controller-free hand tracking device.

1.2.3 Summary of Results

The key results of this dissertation can be summarized as follows.

- A hybrid framework of optical and inertial tracking is presented to achieve 6-DOF motion tracking [18]. The proposed universal motion-based control supports general 2D/3D interactions and motion recognition and is demonstrated through two applications, **Lego Bricks** and **Ogre Scene Navigation**.

- A normalization process is presented to make the motion gesture recognizer scale and speed invariant [14, 15]. Normalization is proven to be effective to improve the recognition accuracy, especially for the user independent case. Depending on the features, the absolute increase of the recognition rate can range from 2% to 20%.

- An evaluation of motion gesture recognition with different dimensions of tracking signals is presented [14, 15]. Motion data beyond a 2D spatial trajectory is informative to distinguish gestures. With 6-DOF motion data, the recognition rate is 99.7% for the user-dependent case and 96.8% for the user-independent case.

- The normalization process is modified to address the offset issues arising for air-handwriting recognition [17, 16]. Isolated motion characters are recognized in a manner similar to motion gestures, and the lowest character error rate is 1.1%. Two approaches are proposed for motion word recognition. Word-based motion word recognition achieves a very low word error rate (WER) but has

little flexibility. Letter-based word recognition allows arbitrary letter sequences at the price of a higher WER.

- A window-based method of handwriting event detection is presented [16]. In terms of writing event detection, nearly all the writing segments (2699 of 2700) are detected. The quality and precision of the detected writing segment directly affects the recognition performance. The overall WER is 1.2% for word-based recognition and 9.8% for letter-based recognition.

- A usability study of using air-handwriting for text input in the universal motion-based control is conducted [17]. The words-per-minute of air-handwriting is 5.43, which may not be fast enough for general-purpose text input. The usability study shows that air-handwriting is intuitive and preferable for short text input. The result supports that air-handwriting is a suitable alternative for infrequent and short text input on a motion-based user interface.

1.3 Outlines

The rest of this dissertation is organized as follows.

In Chapter 2, we present the background knowledge related to motion-based control. First, we review the existing motion tracking technologies in details and explain the pros and cons of each technology. We also go over the evolution of user interface design. Then, we present the background knowledge related to motion recognition. Segmentation of the intended part from the motion tracking data is essentially the first problem before recognition. We discuss the common approaches for segmentation. Relevant motion recognition techniques are discussed in the perspective of gesture recognition and handwriting recognition.

In Chapter 3, we present our hybrid framework for 6-DOF motion tracking with the study of tracking performance in static and dynamic cases. We then explain the design of the universal motion-based control framework in details, including 2D user

interface, 3D user interface, and motion recognition. We also show the implementation of a working prototype of the universal motion-based control framework.

In Chapter 4, we focus on the problem of motion gesture recognition. A gesture database of comprehensive 6-DOF motion data is presented first. We propose two approaches, statistical feature-based and hidden Markov model (HMM)-based, for motion gesture recognition. The recognition performance is evaluated with different dimensions of tracking signals for both user-dependent and user-independent cases.

Chapter 5 is dedicated for air-handwriting. We first explain how air-handwriting differs from the conventional handwriting. We propose a HMM-based approach to modeling air-handwriting with the basic elements of characters and ligatures. We solve the recognition problem with word-based and letter-based word recognition. In addition to the push-to-write scheme for air-handwriting with our 6-DOF tracking system, we also propose a window-based method to detect and segment air-fingerwriting, which is tracked with a different controller-free device. To use air-handwriting as an alternative method for text input, we conduct usability study to investigate the input efficiency, motion footprint, arm fatigue, and other subjective dimensions.

Finally in Chapter 6, we conclude this dissertation with the overall summary of our research, our main contributions, and future research topics.

CHAPTER II

PROBLEM BACKGROUND

Human-computer interaction can be represented as a framework in Figure 2[21]. The user interface, i.e., the input and output devices, bridges between the user and the system. The user articulates his or her intension through the input device, which can be a mouse, keyboard, remote, or touchscreen. The system interprets the input, executes the task, and presents the result on the output device, which can be a display, a speaker, or a haptic device. The user then evaluates the feedback and continues another loop of interaction as in Figure 2. To apply the general human-computer interaction framework to a motion-based user interface, there are two key components we need to consider: motion tracking and motion-based control.

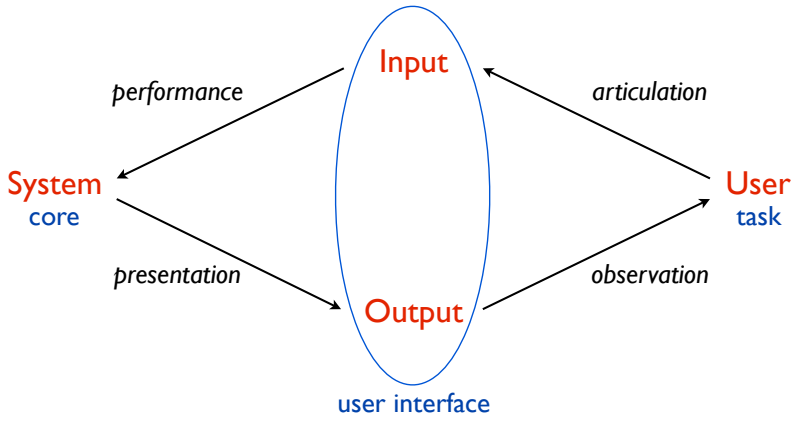


Figure 2: The general human-computer interaction framework.

In this chapter, we will elaborate the origin and history of the problem in three aspects: *1*) motion tracking; *2*) evolution of user interfaces; *3*) relevant motion recognition techniques.

2.1 Motion Tracking

A motion tracking system is the input device for a motion-based user interface. Common technologies for 3D motion tracking include optical-sensing, inertial-sensing, and magnetic-sensing [65]. These sensing technologies have their individual characteristics in terms of the sampling rate, latency, spatial resolution, and spatio-temporal accuracy in implementation. Both the speed and the precision of motion tracking are critical for interactive applications. These technologies are analyzed as follows.

- Magnetic Tracking — Typical products that use this line of technology include MotionStar by Ascensions and PATRIOT by Polhemus. Magnetic tracking systems are subject to error in their tracking results largely due to distortion of the magnetic field caused by the presence of metal or electromagnetic sources in the environment. It requires stringent calibration to take into account the map of the magnetic field and re-calibration once the magnetic field changes. For control functions of indoor digital systems, the technology is either insufficient in performance, too expensive, limited in the range of operation, or hard to deploy in practical applications.

- Inertial Tracking — Inertial tracking makes use of accelerometers and gyroscopes for getting physical information about acceleration and angular speeds in the device-wise coordinates of the inertial measurement unit (IMU). The orientation, as a measurement, is actually accumulated from the angular speeds and automatically calibrated by the direction of gravity measured by the accelerometer and the magnetic north (if magnetic sensors are equipped on the tracking device). It is possible to reconstruct the explicit motion trajectory by integrating the accelerations along the corresponding orientation, but the result is not reliable due to drifting and error propagation over time. The accuracy of inertial sensing has been studied in static, quasi-static, and dynamic cases in

[23]. One critical issue with this technology is that its measurements are jittery and noisy in general. To perform precise tracking of subtle control motion, extra post-processing/filtering is required. The advance of micro-electro-mechanical system (MEMS) technology makes inertial sensors cost-effective and compact in size. Although inertial tracking cannot track spatial trajectory precisely, it provides acceleration and orientation data.

- Optical Tracking — Optical tracking is the most competitive among all tracking technologies for use in motion-based user interfaces. There are two types of optical tracking: vision-based and traker-based. Vision-based optical tracking uses computer vision techniques to recognize the tracking target from the scene, which can be the user's head, hand, or body. Because vision-based optical tracking can be controller-free, it provides more natural and unencumbered interaction. The most successful vision-based optical tracking system is Kinect by Microsoft. Combining a simple RGB camera and a depth camera, Kinect is able to track human body in 3D, but it also has limitations on resolution and precision. For example, Kinect cannot track subtle hand movements such as wrist twisting, and is not suitable for interactions involved fine control. The accuracy and the robustness of vision-based systems are affected by many factors, including illumination and lighting, color of existing objects, and occlusion instances. The sensing technology for Kinect may be suitable for multi-user video games, but it poses as a serious system design problem when it comes to execution of a single user command-and-control protocol, especially for the authorization of control.

  The performance of optical tracking can be improved if it is dedicated to hand tracking in near depth range. Glove-based optical tracking has been proposed to ease and speed up the problem of hand tracking [64]. However, putting on

a glove may be cumbersome and uncomfortable for a long session of interaction. Controller-free motion tracking is believed to bring the most natural user experience. In [63], two consumer-grade webcams are used to achieve bimanual 6-DOF pose estimation at interactive rates for reliable pose detection, such as pinching and pointing. This technology [1] can track the user's hands to finger-precision and is further improved to millimeter-level accuracy with 3D cameras such as Kinect. In May 2012, Leap Motion publicly announced the Leap [2], a small USB peripheral device which is designed to track fingers (or stick-like objects such as a pen or chopstick) precisely in a desktop environment. The smaller tracking volume and higher resolution of Leap differentiates itself from Kinect, which is designed for body and face tracking in a livingroom-like environment.

The tracker-based optical tracking, e.g., OptiTrack, WorldViz, tracks either active or reflective markers with multiple cameras and provides accurate motion tracking results at a relatively high speed. A primary constraint of all optical systems is that there must be a clear line of sight between the tracking targets and the optical sensors, and at least two pairs of the target-sensor relationship are needed for valid triangulation to determine the position of one target. In our intended scenario, the user is usually facing the screen while performing the control action. Therefore, the occlusion problem should rarely happen if the optical sensors are properly mounted on the screen facing the normal direction toward the user. The tracker-based optical tracking provides motion trajectory data associated with a tracker, which results in better performance in terms of spatial resolution, precision, and responsiveness. A more thorough study of the characteristics of the signals acquired by optical tracking has been documented in [61, 13].

- Hybrid Tracking — Hybrid tracking combines different types of motion tracking technology and fuses different sensor data to enhance and improve the tracking results. For example, PlayStation Move together with PlayStation Eye form a hybrid framework of optical and inertial sensing, which claims to achieve 6-DOF motion tracking in position and orientation. It actually computes the depth by the blob size of Move measured by the single camera Eye. The position tracking of Move is not truly in 3D, and errors in the depth estimation are in general larger. Therefore, the hybrid tracking of PlayStation Move and Eye may be sufficient for gaming, but the precision and accuracy is not suitable for more advanced or precise motion control.

2.2 Evolution of User Interfaces

The evolution of user interfaces starts from the command-line interface (CLI) to graphical user interface (GUI), and to the emerging natural user interface (NUI). In CLI, a user has to learn to use the keyboard and a series of codified commands that can be issued through keystrokes. The syntax and responses of those commands are strict, making CLI unfriendly to novice users. The GUI is enabled by a pointing device, e.g., a mouse, trackpad, or touchpad. In GUI, all actionable choices are laid in front of the user, who uses the pointing device to issue the chosen action. The GUI relies on metaphors for interacting with on-screen contents or objects, and translates these metaphors back into the strictly codified language for the system to process.

Traditional input devices of a user interface include a keyboard, a mouse, and other possible alternative pointing devices, such as a trackpad. Users can type the letter input with a keyboard or use a mouse to control the cursor for point-and-click commands in the GUI paradigm. WIMP, standing for "windows, icon, menu, pointing device", is so intuitive and user-friendly that it is widely adopted for the graphic user interface in our daily life. The widget-based interface for touchscreen can be

considered as a variation of WIMP, which seamlessly integrates the pointing device and the display together. The idea of "touchscreen" first emerged in the 1940s, and its hardware and software system has sufficiently matured and been perfected over the last three decades. Fingertips are regarded as the most direct and natural pointing device, even though a stylus can better achieve detailed control with an unambiguous contact point on the screen. The limitation has been tackled with techniques such as Zoom-Pointing and Take-Off [3]. The emerging technology of "multi-touch" lifts the capability of touch-based control to another level. The touch-based pointing device leads to very useful applications, such as gesture recognition and hand-written character recognition.

The technologies introduced above are based on planar tracking and mainly designed for planar control. Most often, 2D interfaces suffice because most of the content resides in planar forms, e.g., text, tables, and pictures. However, an interaction related to reality, either virtual or augmented, is three-dimensional in nature. The WIMP GUI may still be irreplaceable for tasks that are two-dimensional, but a 3D user interface will be superior when the interaction is taking place within a 3D spatial context. Therefore, a universal user interface should be a mixture of 2D and 3D, and its input device should suffice the needs in both regards.

Traditional keyboard and mouse input devices work well for the 2D user interface in their own context. Degree separation extends the control capability to 3D operations. This paradigm prevails in 3D modeling and animation packages, CAD systems, video games, and many others, but these 3D interactions can be more intuitive, immersive, and efficient with the use of the 6-DOF input device.

Research in 3D user interface has a long history and received substantial boost in the mid-1990s, much due to the discovery of fundamental 3D interaction metaphors. Since then, many of the most basic techniques for 3D interaction tasks have been proposed and implemented. Researchers have often categorized universal 3D tasks

as navigation, selection, manipulation, and system control [11]. It is suggested that most of the design space for 3D interaction techniques has been covered [10]. Although knowledge exists to develop usable a 3D user interface, it is not being used to its full potential for application development, often because of the limitation in technology. Recently, the advance of motion tracking and 3D display technologies impacts the input and output devices of the general human-computer interaction framework. New motion sensing devices can provide a great tracking accuracy at a relatively low price, which make motion-based interactions affordable and popular for general use.

The NUI is meant to be *a*) effectively invisible to users with successive learned interactions; *b*) based on natural elements, i.e., physics. The word "natural" contrasts with the fact that most interfaces require artificial control devices whose operation has to be learned or memorized. Motion-based interactions directly link a user's real world motion with an intended control action in the system and thus will likely fulfill the design goal of the NUI.

2.3 Relevant Motion Recognition Techniques

In Section 2.3.1, we first discuss the segmentation issue i.e., how to determine the boundary of the intended motion for recognition. The relevant motion recognition techniques are presented in the perspective of gesture recognition in Section 2.3.2 and handwriting recognition in Section 2.3.3.

2.3.1 Motion Segmentation

After being activated, a tracking system continues to stream motion data from the user; the data stream contains both an intended control motion and other extraneous motions that do not correspond to any control motion. Thus, we have to extract the intended control motion of a gesture or a handwriting for recognition. There are two paradigms for motion segmentation: explicit delimitation and automatic detection.

Explicit delimitation can be easily accomplished with a push-to-gesture [15] or push-to-write [17] scheme, where the user holds a button to start and releases it to stop.

A controller-free system has no buttons and requires different forms of delimiters for push-to-gesture or push-to-write. A common alternative is to replace the button with a specific posture or gesture to signal the end points of the intended control motion, e.g., a pinch gesture or a waving palm. There are other approaches for explicit delimitation. For example, Kristensson et al. [40] proposed an input zone for gesture delimitation with Kinect. In [58], a user reaches out to write in the air, and the explicit delimitation is done by thresholding the depth information. In sum, the approaches that require delimiters in any forms to signal engagement are still considered as explicit delimitation.

Another paradigm for motion segmentation is through automatic detection (spotting) that requires no intentional delimitation. The system automatically detects the intended motion and segments the gesture or writing part correspondingly. If functioning perfectly, automatic detection can make the motion gesture/air-handwriting experience more convenient, especially for controller-free systems.

Detection of motion gestures can be very difficult when the defined gesture has a similar trajectory to other control motions. For example, even a human cannot distinguish between a swiping right gesture and a moving right control motion. In such a case, the motion itself may not contain enough information for automatic detection, and push-to-gesture is more robust and accurate for motion segmentation.

On the other hand, the writing motion is much different from other general control motions, which makes robust detection of air-handwriting possible. Amma et al. [5] proposed a spotting algorithm for air-handwriting based on the acceleration and angular speed from inertial sensors attached to a glove. They reported a recall of 99% and a low precision of 25% for handwriting spotting. Nonetheless, the recognition performance was evaluated on manually segmented writing instead of the detected

segments, and the author did not address how the detected segmentation affects the recognition performance.

In this work, we choose explicit delimitation for motion recognition with our 6-DOF motion tracking system and develop automatic detection for air-fingerwriting with the Leap [2].

2.3.2 Gesture Recognition

Motion gesture recognition has been an active research topic for years. Although there has been a significant amount of work on recognizing gestures with either explicit or implicit motion information, a thorough study and comparative evaluations of the full six dimensions are lacking.

Here, the definition of a gesture is temporarily relaxed as a finite characteristic motion made in 2D or 3D space using a suitable input device. Gesture recognition is also generalized as a spatio-temporal pattern recognition problem, which may include the sign language recognition. Unlike speech or handwriting, gestures lack a standardized "vocabulary", but there are still several widely accepted basic motions, e.g., swiping motions, circle, etc. High level linguistic contraints can help the recognition of concatenated characters in a word or sentence. In contrast, motion gestures usually don't concatenate and have little contextual information.

Major approaches for analyzing and representing spatial and temporal patterns include dynamic time warping (DTW)[22, 43], neural networks (NNs)[49], hidden Markov models (HMMs)[41, 47, 15], data-driven template matching [67, 38], and statistical feature-based classifiers [56, 27, 20]. In general, the reported recognition rates are above 90%. Since these results are obtained with different data sets and various experimental settings, a direct comparison of the performance achieved by these techniques is not meaningful.

When designing a recognizer, a trade-off is usually made between personalization

and generality. The two extreme cases are user-dependent and user-independent gesture recognition. Customized gestures are usually personal and are only considered in the user-dependent case, which has no generality issue. Even with a predefined gesture vocabulary, robust user-independent gesture recognition can be very challenging due to the large variations among different users.

The DTW is an effective algorithm based on dynamic programming to match a pair of time sequences that contain temporal variability (i.e., stretching or compressing in time). It is an important component in template-based pattern recognition and classification. The issue of general statistical variability in the observation is not explicitly addressed by the matching algorithm and a template-based system is usually used in user-dependent applications, e.g., personalized gesture recognition, where such a variability is limited. DTW can be useful for personalized gesture recognition, where a large set of training samples is hard to collect. When the range of variations increases, e.g., in the user-independent case, the need for explicit statistical modeling of the variability becomes crucial for the sake of performance and computational load.

The $1 recognizer [67] and its variants [38] are also based on template matching. Unlike DTW, which relies on dynamic programming, these algorithms process the trajectory with resampling, rotation, and scaling and then match the point-paths with the reference templates. These recognizers are simple to implement, computationally inexpensive, and require only a few training samples to function properly. However, for user-independent recognition, a significant amount of templates are needed to cover the range of variations and hence the performance of such approaches is often degraded.

The Rubine classifier [56] is a popular feature-based statistical classifier. It captures geometric or algebraic properties of a 2D gesture for recognition. The planar trajectory is converted into a fixed length feature set and recognized by a linear classifier. Hoffman [27] extends Rubine's feature set and works on implicit 6-DOF motion

data, i.e., the acceleration and angular speed. Note that the feature extraction actually treats a gesture more like a static path regardless of the temporal (ordering) information, which may cause confusion between mirroring gesture pairs. In Section 4.2, we propose an extension of the Rubine classifier to incorporate the temporal information, and use it as the baseline for performance comparison.

The HMM is efficient at modeling a time series with spatial and temporal variations, and has been successfully applied to gesture recognition [41, 47, 15] and sign language recognition [60, 52]. Depending on the tracking technology in use, the features (observations) for the HMMs vary, including the position, the moving direction, acceleration, etc. The raw sensor signals may need proper normalization or quantization to handle the variations of gestures, especially in the user-independent case. We will elaborate the feature selection and normalization procedure for HMM-based motion gesture recognition in Section 4.3.

2.3.3 Handwriting Recognition

Research in handwriting recognition has a long history, too. Handwriting recognition is much more challenging than optical character recognition (OCR) of machine printed text. Plamondon and Srihari [53] conducted a comprehensive survey on online and offline handwriting recognition. Offline handwriting recognition treats the handwriting as a static representation, and the offline writing data is usually of the form of a scanned image. Online handwriting recognition addresses the problem from a spatio-temporal point of view, i.e., looking at the writing trajectory instead of the shape. The existing online handwriting data, such as UNIPEN [26], is mostly collected from pen-based or touch-based devices, which track the 2D trajectory with the engagement (pen-up/pen-down) information. Different types of tracking devices are needed when people write in the air, e.g., a vision-based hand tracking system [58] or a set of inertial sensors attached to a glove [4, 5].

Techniques for gesture recognition are applicable here with adaptation, such as DTW [31], NN [30], and HMM [46, 28, 17, 16]. A character or a symbol can be recognized in a manner similar to gesture recognition with different extracted features, such as Legendre coefficients in [25]. Techniques commonly used in automatic speech recognition (ASR) [55] can also be applied to perform word recognition on top of character recognition. Cursive handwriting contains successive letters that are connected without explicit pen-up moves. Sin and Kim [59] used ligature models to handle the inter-letter patterns for online cursive handwriting recognition.

For pen-based or touch-based writing, the ink information is directly included when writing. The pen-up/pen-down moves naturally delimit the strokes for print writing or segment the word boundaries for cursive writing. For air-handwriting, the motion is tracked with a continuous stream of sensor data, which means the writing is uni-stroke with no engagement information. In such a case, delimitation can be accomplished with explicit segmentation (push-to-write) or automatic detection (spotting) as described in Section 2.3.1.

One of the applications of handwriting recognition is text input. There are two main paradigms for text input: typing and writing [44]. The QWERTY keyboard is the primary text entry method for computers and smartphones. The keyboard-based approach usually requires the user to type either physical keys of a keyboard or soft keys on a touchscreen. When neither physical nor soft keyboards are available, we can still point and click at a virtual keyboard on the display to mimic typing, which requires precise pointing motions and extra efforts to type. The key selection on a virtual keyboard can also be done with four cursor keys (up, down, right, left) and an `Enter` key. The drawback of a soft or virtual keyboard is that an eyes-free entry is impossible, and "typing" on a virtual keyboard may not be as efficient as typing on a physical one.

The pen-base text input technique does not necessarily need a pen. The writing

can be done with any pointing device, including a mouse, a trackpad, a stylus, or even a fingertip. Traditional handwriting styles include cursive or hand-printed letters as written on a piece of paper. To make it easier for a machine to recognize and quicker for a user to write, letters are simplified into single-stroke styles. The Graffiti alphabet [9] best exemplifies the uni-stroke handwriting. Because the uni-stroke alphabet differs from conventional writing, practice is required for a novice user to attain fast entry.

There are other text input modalities in addition to typing and writing, such as speech-based text input. Although text entry with automatic speech recognition is effortless, it is not always reliable especially in a noisy surrounding. One alternative approach is a mixture of typing and writing. In Quickwriting [51, 29], the user swipes strokes on zones and sub-zones to input the associated characters. TwoStick [37] applies a similar concept to the two joysticks on a gamepad. Swype allows the user to enter words on a soft keyboard by sliding from the first letter of a word to its last letter and uses a language model to guess the intended word. Similar to typing on a virtual keyboard, swiping strokes also requires the user's attention to the visual feedback while inputing text and is not eyes-free.

In [34], the text entry speed was studied. Words-per-minute (WPM) is a common performance metric for text input efficiency. WPM is computed based on correctly input word units, where one word unit is five letters (keystrokes). The reported average corrected WPM are 32.5 and 13.6 for keyboard and speech transcription entry respectively. In general, the typing speed falls in the range of 20 to 40 WPM for hunt-and-peck typists and in the range of 40 to 60 WPM for skilled typists. Human hand-printing speeds are commonly in the range of 15 to 25 WPM regardless the recognition accuracy. Although handwriting is not the fastest, it is the most primitive method for text input. In [39], a longitudinal user study of text entry

performance was done by comparing typing on a soft QWERT keyboard and pen-based unconstrained handwriting on a tablet. Their result shows that handwriting leads to a performance almost identical to that of a soft keyboard. When the writing or typing motions are rendered on a virtual plane instead of a real surface, it brings out new usability issues such as arm fatigue, and we will study the usability issue of air-handwriting in Section 5.4.

CHAPTER III

UNIVERSAL MOTION-BASED CONTROL

3.1 Hybrid Framework for 6-DOF Motion Tracking

It is necessary to capture a user's motion before the system interprets his or her intent for any control. When choosing a suitable motion tracking technology for the motion-based user interface, desired features should include: *a*) minimal size; *b*) free of tether; *c*) accurate; *d*) responsive; *e*) full 6-DOF rigid body motion tracking, i.e., position and orientation. A 6-DOF tracking device allows a user to control position and orientation simultaneously, which is convenient in most of 3D interactions. Also, a free moving 6-DOF device is most suitable when speed and a short learning curve are of primary concern. For interactions that require less than 6 DOF, the spared dimensions can be used for other control functions as long as the mapping is logical without interfering with the main interaction.

As explained in Section 2.1, there is no single technology that encompasses all these desired features. In a pilot study [18], a hybrid of optical sensing and inertial sensing was found to be particularly suitable for the proposed paradigm. The former measures the position of an optical tracker, and the latter estimates the orientation of the tracking device. The prototype of the motion-based control device supports simultaneous control of both 3D position and orientation with a minimum number of buttons on it. The controller prototype of the hybrid 6-DOF motion tracking is shown in Figure 3.

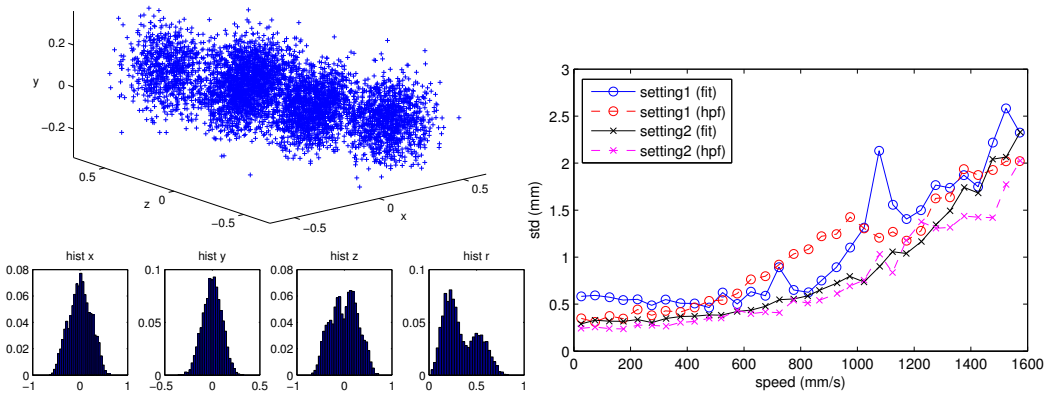
As for an initial implementation, WorldViz PPT-X4 is used as the optical tracking system, which contains four cameras mounted on the top four corners of the tracking volume (approximately 3.5 $\times$ 3.5 $\times$ 2m). PPT-X4 tracks infrared dots at 60 Hz

Figure 3: A prototype of the motion controller.

and transmits the results with Virtual Reality Peripheral Network (VRPN) through Ethernet. In [13], the characteristics of optical motion tracking signals were studied in both static and dynamic aspects. In the static case, the tracking target remains stationary during measurement, and the mean position is used as the ground truth to compute the spatial jitter and distortion. Without loss of generality, we measured the position of a stationary marker that is placed near the center of the tracking volume. The spatial jitter of the 3D signal and the histograms in x, y, z coordinates, and radius is shown in Figure 4a. The 3D point scatter doesn't follow a Gaussian distribution, but the standard deviations are less than 0.3mm, which means a sub-millimeter precision.

The dynamic precision is measured with the pendulum motion, which can be characterized by a second order ordinary differential equation $\theta'' = -\alpha_1 sin\theta - \alpha_2\theta'$, where α_1 relates to the length of the pendulum, α_2 is the damping factor, and θ is the angle between the pendulum and the vertical equilibrium line. An infrared LED is attached at the end of the compound pendulum and swings in two different orientations to see the effect of motion blur in different projective views. Two methods, fitting with the second order ODE and high-pass filtering, are used to extract the spatial jitter

from the pendulum motion. Both methods show that the spatial jitter grows as the motion speed increases, but the dynamic spatial jitter is still small compared to the control motion in scale. In Figure 4b, the relationship between motion speed and the standard deviation of jitter is plotted. In summary, the static and dynamic precision of the optical tracking system is of millimeter level.



(a) Static 3D spatial jitter and histograms (b) Standard deviation vs. motion speed

Figure 4: Performance evaluation of PPT-X4 in static and dynamic aspects.

After studying the capability of optical tracking, we also build our own optical tracking system that consists of two synchronized mono-color Firefly MV cameras mounted on a rig as show in Figure 5. Similar to the deployment of Kinect, the camera rig is placed in front of the display in a living room-like environment. The angle panning of the two cameras are adjusted to achieve the largest overlap of field of view around where the control activities occur. We use OpenCV to calibrate the intrinsic parameters, estimate the camera positions and orientations, and compute the camera matrices. We then extract the 2D blobs position of the infrared tracker on both views to triangulate the 3D position. We set the camera frame rate to 60 Hz and complete the tracking process in real-time. The optical tracking results are transmitted through VRPN, so we can easily swap the back-end optical tracking system without affecting the front-end applications. The precision of our own implement is slightly worse than the PPT-X4. However, there is no discernible difference in terms of user experience of our motion-based control.

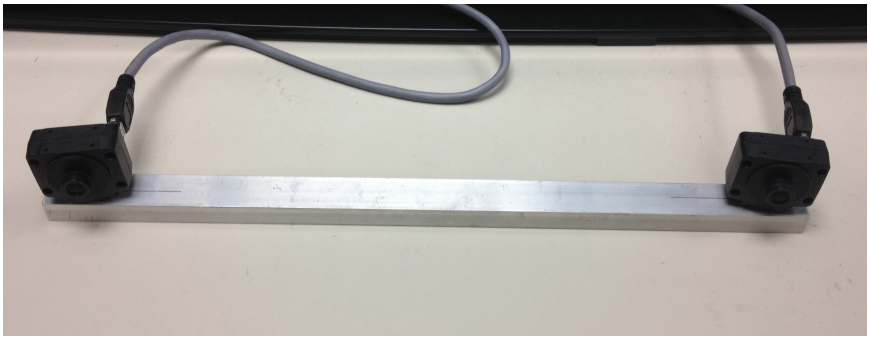
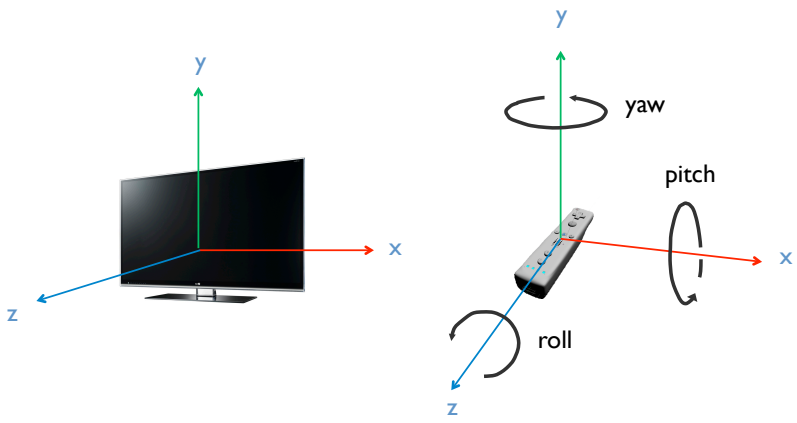


Figure 5: The camera rig of our own optical tracking system.

As for the inertial tracking, we use the accelerometers and gyroscope embedded in Wii Remote Plus (Wiimote), which samples and reports at about 100 Hz. It also facilitates the prototype implementation to use Wiimote as a module that integrates buttons, sensors, and Bluetooth for wireless control. The orientation of the tracking device is then computed from the inertial measurement. The Kalman filter has become the basis of many orientation filter algorithms, but it demands a large computational load and high sampling rates, typically between 512 Hz and 30 kHz. Hence, we implement the Madgwick's method [45] for orientation estimation, which is reported to perform similar to and slightly better than the Kalman filter at a relatively low sampling rate but with much lower computation. The acceleration, i.e., the indication of gravity, is used to calibrate the orientation in pitch and roll. Because Wiimote is not equipped with a magnetometer, automatic calibration in yaw cannot be done. The drifting issue in yaw is solved by manually aligning the controller to the global coordinates and resetting the orientation to identity quaternion periodically. The experimental results [18, 19] show that the orientation estimation is stable enough for the duration of several consecutive control motions.

The hybrid tracking framework updates the tracking results from PPT-X4 and Wiimote at 60 Hz, and gives us the essential 6-DOF motion data for the universal motion-based control. In addition to the explicit position and orientation, the tracking framework also provides the acceleration and angular speed. The implicit motion

data can also infer the kinematic properties of the motion gesture. Throughout this dissertation, we use a right-handed Cartesian coordinate system for both the global and the device coordinates as shown in Figure 6.



(a) the global coordinates (b) the device coordinates

Figure 6: The right-handed coordinate system for our 6-DOF motion tracking

3.2 Universal Motion-based Control Framework

In the area of human-computer interaction, there are several principles that need to be considered for a universal design [21]. Interaction techniques that follow a consistent metaphor are easier to learn and will avoid errors, but no interaction technique can be effective in every manipulation situation. Because the proposed motion-based control is meant for general use across a variety of interactions, trade-offs are made among generality, performance, and usability, in establishing the mapping of the interaction technique to the device. The user's past experiences with computers, mobile devices, or video games can be useful to duce a logical mapping of causes onto effects of motion-based control. A detailed overview of the universal motion-based control framework is shown in Figure 7.

Even though a 6-DOF device allows simultaneous control of position and orientation, the ergonomics needs to be considered. It is against human nature to achieve mutual independence of 6-DOF hand motion, especially when the hand is held in the

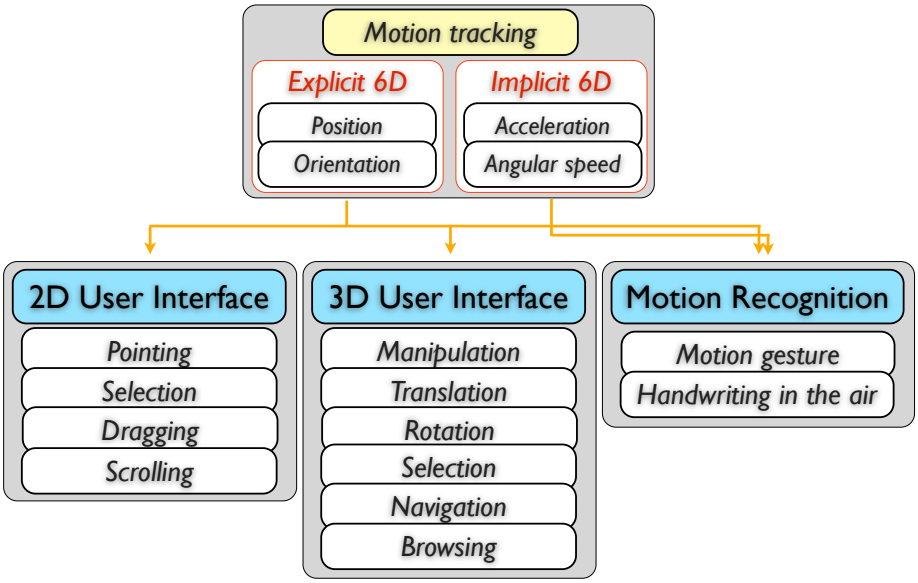


Figure 7: A detailed overview of the universal motion-based control.

air. For example, it would demand an extra effort for the user to make translation without rotation and vice versa. Therefore, the ergonomic issue needs to be addressed in the design of motion-based control. In the following sections, we will elaborate the design details of 2D/3D user interfaces and motion recognition.

3.2.1 2D User Interface

An effective motion-based control should support common functions of a 2D graphic user interface, such as pointing, selection, and scrolling. There are two metaphors for the techniques of motion-based remote pointing: *ray-pointing* [50, 32] as used in Nintendo Wiimote, and *virtual mouse in the air* [12] as used in PlayStation Move and XBox 360 Kinect. Both of them are intuitive, slightly worse than a mouse, and better than a joystick-type controller in terms of performance. *Ray-pointing* follows the metaphor of a laser pen in real world and easily covers a large area with minimal hand motion. The user has to carefully maintain the pointing posture, which makes the selection task especially hard on a fine scale. On the other hand, *virtual mouse in the air* allows direct spatial mapping without any posture constraint, but the motion footprint is relatively large. The supported selection precision of the virtual mouse

metaphor depends on the precision of the motion tracking technology in use.

Virtual mouse in the air is chosen because it is consistent with the metaphor used for 3D manipulation. The motion on a virtual plane parallel to the display controls the cursor correspondingly. The 2D cursor position on the screen is calculated by projecting, scaling, shifting, and clipping the 3D coordinates of the controller. The scaling factor is determined based on the screen resolution, the operating arm range, and the precision of the optical tracking. In the current setting, a motion of 1 cm in real world corresponds to a translation of 20 pixels on the screen. The operating range for the full HD resolution (1920×1080) is roughly 100×60 cm. The Button A and B on the Wiimote are analogous to the left and right clicks of the mouse. Therefore, the user can easily extend his or her experience of the mouse to the motion-based control for 2D user interfaces. Although the concept is the same as a conventional mouse, *virtual mouse in the air* translates the cursor based on the absolute position and hence doesn't have the mouse acceleration, i.e., increase in the speed of the cursor based on the speed of the control movement. We use Button Down to reset the current controller position to be mapped to the center of the screen, so the user can easily adjust the center of the control motion.

Pointing and selection are the two fundamental functions in 2D GUI. The dragging function comes as an natural extension of the implementation of pointing and selection. Another common function of a 2D user interface is scrolling, which is conventionally supported by the scrolling wheel on a mouse. Scrolling can also be done by the swiping motion with multi-touch on a trackpad or a touchscreen. In the implementation of *virtual mouse in the air*, the depth information is unused for the cursor control and can be bound to auxiliary functions such as scrolling or zooming. The scrolling and zooming functions are integrated together because they follow the same mental model that drags the content and move, e.g., drag down to scroll up, or drag toward the body to zoom in. To avoid interference with the ordinary cursor

control, the scrolling and zooming functions are triggered by holding Button B (the secondary button). In sum, the control motion in 3D space is logically translated to general interactions on a 2D user interface.

3.2.2 3D User Interface

In the 3D user interface design, the supported interactions are categorized into 3D manipulation and 3D navigation. The 3D manipulation supports selection and general manipulation (translation and rotation) in 3D space. For 3D navigation, two techniques are designed to address different applications.

3D Manipulation Among the well-known techniques for selection in a virtual environment [11], the two most intuitive metaphors are *ray-casting*[42] and *hand-extension*[48]. Both of them work very well even in a dense and occluded condition with certain modifications [62].

With *ray-casting*, the user points at an object with a virtual ray that defines the pointing direction. The virtual ray can be attached directly to a 6-DOF controller or a 3D widget that can be controlled by a mouse. In terms of target selection, *ray-casting* has a smaller motion footprint and a relatively large coverage in the virtual environment (the virtual ray can shoot to infinity). It suffers when selecting distant and small objects due to the angular accuracy. The hand tremor when holding in the air also affects the pointing precision of *ray-casting*. On the other hand, the *hand-extension* metaphor directly maps the user's hand to a 3D virtual cursor in a one-to-one correspondence. It is extremely intuitive, but the possible reaching range is limited. As for object manipulation, *hand-extension* is more natural and efficient than *ray-casting*.

In our implementation for 3D manipulation, the *hand-extension* metaphor is chosen because it provides direct 6-DOF manipulation, seamless integration with the 3D browsing technique, and consistency to the *virtual mouse in the air* for 2D user

interfaces. To extend the virtual reaching range, a non-linear mapping is derived in Equation 1 based on the go-go technique [54]:

$$R_v = \begin{cases} sR_r & if R_r < D \\ sR_r(1 + k(R_r/D - 1)^2) & otherwise, \end{cases} \quad (1)$$

where R_r and R_v are calculated from the user-defined origin, s and k are scalars, and D is the radius of the linear mapping sphere. The user-defined origin in the real-world coordinates can be set to any location where the user feels comfortable to start with, and then the position of the controller is offset and scaled to calculate the position of the virtual hand. Since the user can easily recognize the device's orientation, a one-to-one mapping for 3D orientation is used. Other specific or complicated manipulation tasks, such as resizing, 3D cloning, multiple selections, etc, can be done with the help of 3D widgets, which are similar to the menu brought up by a secondary click in window-based 2D user interfaces.

3D Navigation 3D navigation allows the user to translate and rotate the viewpoint, i.e., the first-person view, in the virtual world. Translation can be one dimensional (moving forward and backward), two dimensional (add moving left and right), or three dimensional (add moving up and down). Rotation can also be one dimensional (yaw), two dimensional (yaw and pitch), or three dimensional (yaw, pitch, and roll). The total dimensions needed for 3D navigation depend on the degrees of freedom allowed or required by the application. For instance, most virtual reality applications rotate the viewpoint with two DOF because the rotation in roll is restricted. Terrain-following is another common constraint that reduces the DOF of viewpoint translation to two.

Two techniques are designed: *steering* for the walking/flying-type navigation, and *grabbing the air* for the browsing-type navigation. Both of them support full 6-DOF navigation, and the degrees of freedom can be further reduced based on the

application, e.g., 2 DOF for terrain following. 3D navigation is a common interaction in a virtual environment, but it is rarely standalone. 3D navigation has to work together with selection and manipulation to make the user interface complete. In the implementation, the user has to hold Button B (secondary button) to control the navigation and release to stop to avoid possible interference with other interactions, such as 2D cursor control or 3D manipulation.

- *Steering* — It is the most common metaphor for navigation. Our design turns the motion controller into a 6-DOF analog stick for walking/flying-type navigation. The position while Button B is pressed down serves as the origin of a polar coordinate system. The controller position is then converted in the corresponding coordinates as r, θ, ϕ, where θ and ϕ control the heading direction, and r controls the velocity of the viewpoint translation. r is non-linearly mapped to the velocity with a dead zone to suppress unintentional hand movement. The controller's angles in yaw, pitch, and roll control the respective angular speeds to rotate the viewpoint. Similarly, angles below certain thresholds are ignored for rotation. Upper bounds are set for translation and rotation speed when r or the rotation angles reach beyond thresholds.

  Steering is suitable for applications that require 3D walk-through in a virtual scene and 2D interactions, e.g., control on the head-up display (HUD). In our design, we combine *steering* for 3D walk-through and *virtual mouse in the air* for 2D user interfaces together.

- *Grabbing the air* — This navigation technique for 3D browsing is the counter part of the scrolling and zooming on a 2D user interface. The world is viewed as an object to be manipulated while the viewpoint remains stationary. Dragging the world backward has the same effect as moving the viewpoint forward. The viewpoint rotation is done reversely to the hand rotation while grabbing the air,

so the user keeps the same mental model. Note that it is difficult and requires extra caution to move the hand without rotation, so the rotation mapping is separated from translation by default to avoid causing disorientation. The user can specify his or her intention of viewpoint rotation by gripping the controller with a point-up heading when he initiates the 3D browsing interaction. After the intention of rotation is signaled, the hand motion is considered as pure rotation if the displacement in position is small or reverted to 6-DOF control if the translation exceeds certain threshold. Comparing to pointing forward, the gripping posture of pointing upward is also more ergonomic to have a larger range of wrist rotation in all directions.

Since *grabbing the air* follows the hand-extension metaphor, it is suitable in situations where both navigation and object manipulation tasks are frequent and interspersed. *Grabbing the air* is the default navigation technique that works with 3D manipulation in the integrated framework. This combination of interaction techniques is appealing for applications like the 3D widget wall, which can be floating in the air and virtually surround the user.

3.2.3 Motion Recognition

Motion recognition is an important feature in the proposed universal motion-based control. Motion recognition provide a natural and complementary modality in human-computer interactions. As shown in Figure 7, motion recognition contains two levels of sophistication: motion gesture recognition and air-handwriting recognition. The user can render a gesture to trigger a certain task or command and use air-handwriting to input text. Motion gestures can be viewed as an array of virtual buttons, and the function mapping is no longer restricted by the limited number of physical buttons, which simplifies the layout design of the control device. The design space of motion gestures is covered in [6, 57]. A good motion gesture design should meet the following

criteria: *a*) easy to remember and perform, *b*) reliably activate the desired function with few false positives, and *c*) a logical mapping onto functions to ensure the usability.

In our design, motion recognition is initiated with the push-to-gesture/push-to-write scheme by holding Button A (primary button). In addition to the explicit delimitation of the continuous motion tracking signals, holding a button has the advantage of clear intension for gesture/handwriting recognition. As a result, motion recognition can be integrated into the control framework without interfering the interactions on 2D/3D user interfaces. Motion recognition can also be used as a stand-alone operating mode if no cursor or any pointing functions are needed, e.g., the control interface of a digital video player or a smart TV. Automatic detection of air-handwriting is an extension work and is not part of the current universal motion-based control.

The recognition of gestures and handwriting by itself is a challenging topic and beyond the scope of universal user interface design. The motion gesture recognition is elaborated in Chapter 4, and the air-handwriting recognition is covered in Chapter 5.

3.2.4 Demo Applications

The visual part of our universal motion-based control is built with Object-Oriented Graphics Rendering Engine (OGRE). On top of it, we create two applications, **Lego Bricks** and **Ogre Scene Navigation**, to demonstrate the usability of the universal proposed motion-based control. We also create a high level 2D GUI to hold the two demo applications, the motion recording and viewing programs. Motion gestures are used to confirm (`Vshape`) or cancel (`Xshape`) a dialog box, pause (`SwipeUp`), resume (`SwipeDown`), and switch between applications (`SwipeLeft`, `SwipteRight`).

Lego Bricks best exemplifies the frequent switching and close integration of 3D manipulation and 3D navigation (browsing). The user has to assemble lego bricks that are scattered randomly on the ground. These bricks can only be assembled when they are aligned with small errors in position and orientation, just like assembling

real Lego bricks. A red overlay is shown whenever an object is "grabbable" by the virtual hand (a Wiimote model). Otherwise, the grabbing target is the air. When the controller is pointing upward, a semi-transparent green sphere is shown at the head of the Wiimote model to inform the user that 6-DOF browsing can be triggered. We also show a red overlay on objects when they are engaged but not correctly aligned. A green overlay on objects indicates correct alignment, and the user can press button A to assemble them. This application shows that the *hand-extension* metaphor is perfectly intuitive for both 3D manipulation and 3D browsing in this scenario.

Ogre Scene Navigation allows the user to either walk or fly in the virtual scene with an HUD. By default, this demo application operates in the 2D cursor mode, in which the user can choose to either walk or fly from a drop down menu. When holding button B, the interface switches to 3D navigation interaction. Note that the 3D walk and fly interactions are actually 4-DOF and 5-DOF because the viewpoint rotation in roll is restricted.

In Figure 8 and 9, we show the screenshots of **Lego Bricks** and **Ogre Scene Navigation** applications with picture-in-picture snapshots of the hand motion. Stereoscopy is disabled for illustrative purpose. The universal motion-based control is best demonstrated by videos[1]. Before we move on to the motion recognition part, we summarize the design of motion-based control for the 2D user interface, 3D user interface, and motion recognition in Table 1.

[1] Universal Motion Control: `http://youtu.be/KaoxMg5nNcs`
Motion Gesture Recognition and Control: `http://youtu.be/jYchogTdGp0`

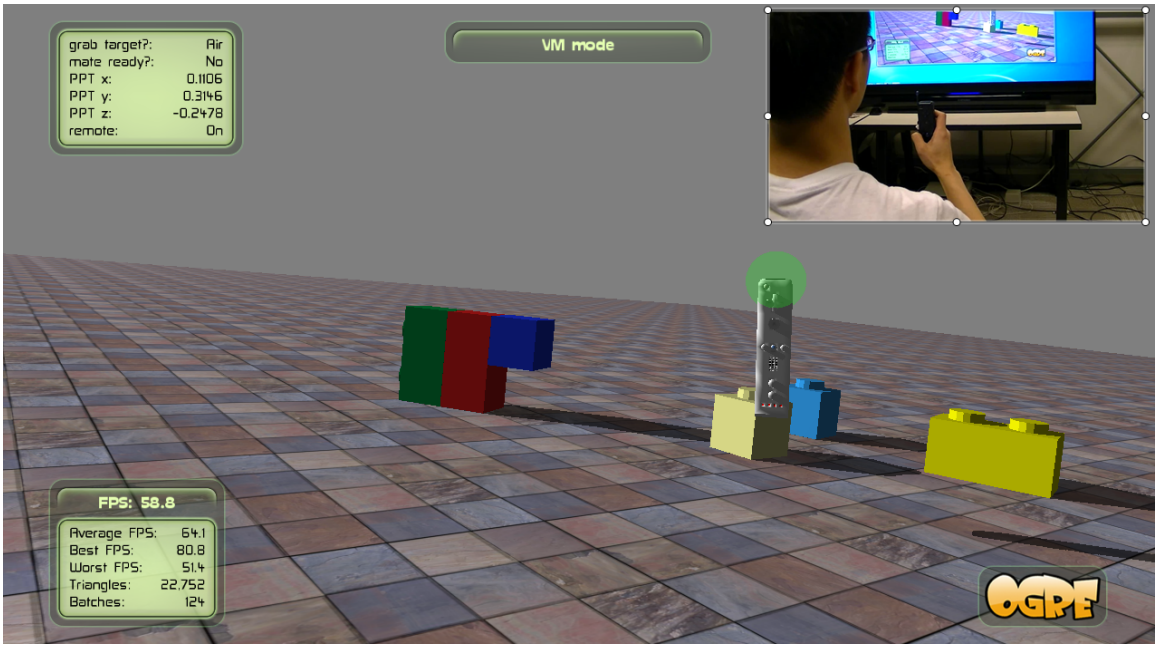


Figure 8: Lego Bricks application

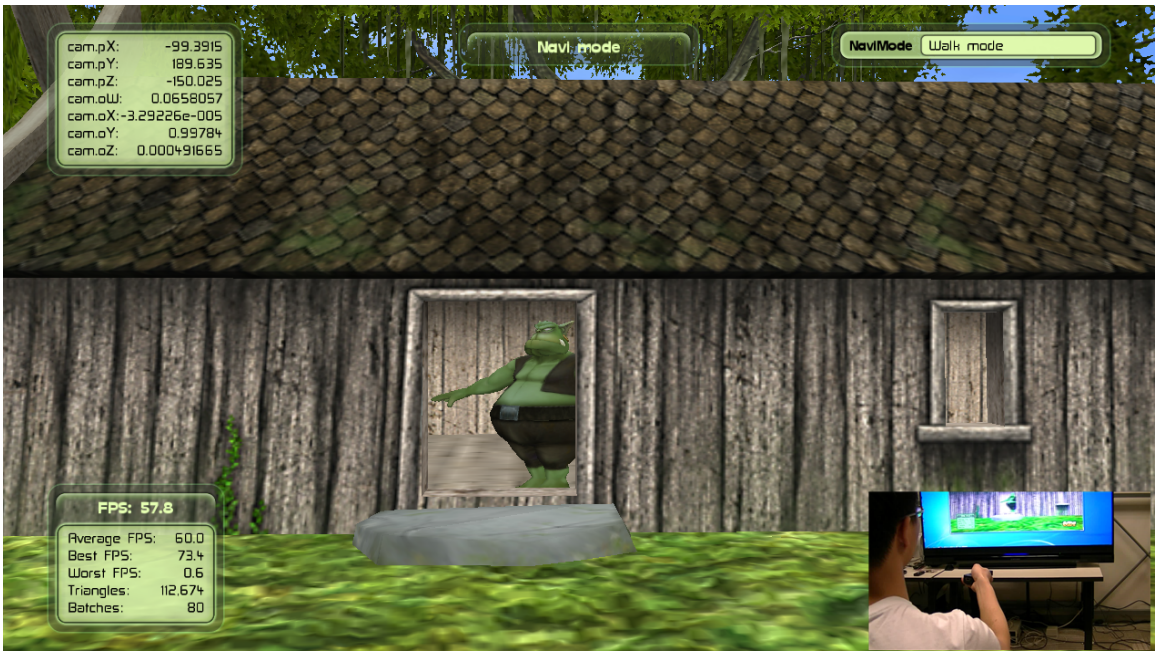


Figure 9: Ogre Scene Navigation application

Table 1: The design of motion-based control for the 2D user interface, 3D user interface, and motion recognition.

design	interaction technique	buttons	control motion	UI action
2D cursor[1]	*virtual mouse in the air*	`A`: left-click `B`: right-click	*xy*-plane motion	cursor movement
3D manipulation[2]	*hand extension*	hold `B` to grab an object	translation and rotation	1-to-1 motion mapping
3D browsing[2]	*grabbing the air*	hold `B` to grab the air	translation	reverse camera translation
		+ point up	+ rotation	+ reverse camera rotation
3D walk-through[1]	*steering*	hold `B` to activate	translation	2D stepping / 3D flying
			rotation	panning, tilting, (rolling)
gesture control[3]	-	hold `A` to activate	motion gestures	specified actions
text input[3]	-	hold `A` to activate	air-handwriting	recognized text

[1]The 2D cursor and 3D walk-through are combined to support tasks of 2D interactions and 3D scene navigation simultaneously.
[2]The 3D browsing is the 3D navigation method to be seamlessly integrated with 3D manipulation.
[3]Gesture control and/or text input can be activated from the aforementioned design depending on the need.

CHAPTER IV

MOTION GESTURE RECOGNITION

Motion gesture recognition can be challenging because users tend to perform the same gesture differently in terms of motion speed, scale, and style. Similar to the case of speech recognition, it is desirable that the gesture recognition system accommodates user-specific customization, but it is also very important to achieve robust user-independent recognition if such a recognition system is to be deployed to serve publicly, e.g., as in an information kiosk. Therefore, both user-dependent and user-independent recognition should be addressed. With our 6-DOF motion tracking system, we can investigate the relative effectiveness of various features derived from different tracking signals for motion gesture recognition. In Figure 10, a general system diagram for motion gesture recognition is shown, where the tracking technology, the corresponding motion data, and the recognition kernel vary upon implementation.

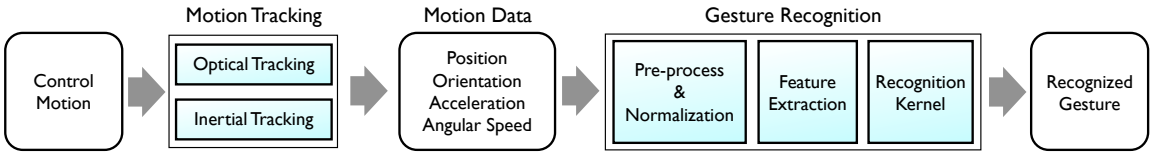


Figure 10: The system diagram for motion gesture recognition.

It is very important to understand what defines a motion gesture before recognition. In most cases, it is the spatial trajectory that matters. This basically holds true not only for our gesture set but also for other existing gestures with 3D spatial or gaming interactions. Exception exists when the spatial trajectory contains little or no deterministic information. For example, the wrist twisting gesture is better described by the change in orientation or angular speed than the position because the spatial trajectory of wrist twisting is small and varies upon the gripping posture.

Therefore, the features for gesture recognition should be primarily extracted from the spatial trajectory and also supplemented with orientation information.

In general, people recognize a gesture by the path spanned by the motion regardless of its speed and scale. Therefore, the recognizer should not be affected by the speed or scale unless fast/slow or big/small motions have different meanings in the gesture set. This is very unlikely to happen especially in user-independent systems because the definition of fast/slow or big/small motions can be vague and different among users.

We first introduce the notation for the motion data. Let $A^o = [a_x, a_y, a_z]^\top$ denote the device-wise accelerations and $W^o = [w_y, w_p, w_r]^\top$ denote the angular speeds in yaw, pitch, and roll, respectively. From the position data, we can derive P^o and V^o, where $P^o = [p_x, p_y, p_z]^\top$ denotes the positions offset by the starting position, and $V^o = [\Delta p_x, \Delta p_y, \Delta p_z]^\top$ is the rate of change in position. In our 6-DOF motion tracking system, the orientation is represented in quaternion, $O^o = [q_w, q_x, q_y, q_z]^\top$. Although it is easier to interpret and visualize Euler angles, an Euler representation suffers from gimbal lock and discontinuity when the angle wraps around, and it is numerically less stable near a singularity. The notations above represent the time sequences of a gesture in corresponding coordinates, e.g., $A^o = [a_x(i), a_y(i), a_z(i)]^\top, i = 1, 2, ...N$, where N is the number of samples in a gesture.

In this dissertation, a motion gesture is recognized by treating it as a static pattern or by considering the time series nature of the motion. In the former approach, a corresponding fixed length feature set is extracted from various tracking signals. These features are either geometric or algebraic and barely contain any temporal or ordering information. Thus, a temporal extension to the feature set is proposed to include the temporal characteristics of a motion gesture. Benchmark recognition results are then obtained by applying a simple linear classifier on the extracted features. The second

approach represents the motion gesture as a sequence of feature vectors (observations) derived from various tracking signals and uses hidden Markov models (HMM) for recognition. The HMM structure is chosen with reasonable physical meanings. We also propose a feature normalization procedure and prove its importance and effectiveness in achieving "scale" invariance especially for the user-independent case.

We will elaborate the implementation details of these two approaches of gesture recognition followed by a performance evaluation. The statistical feature-based linear classifier [20] has the advantage of fast and easy implementation yet with reasonable performance, and it works as the baseline. The hidden Markov model-based classifier [14] is based on a more sophisticated statistical model framework and utilizes the time series nature of motion signals to achieve an improved performance.

4.1 6DMG: 6D Motion Gesture Database

There is no standard or consensus on how motion gestures should be defined, performed, and mapped onto commands invoked on the system. Wobbrock [66] elicited motions from 20 participants in response to a variety of tasks to find the commonalities in mental models for user-defined gestures. The design of motion gestures can be considered as manipulating established motion taxonomies while preserving the logical (and conventional) mapping of causes to effects [57]. Common motion gestures are mostly defined with 2D movements on a plane. It is natural that human motions are still in 3D even though people intend to perform planar motions. The additional information beyond a 2D trajectory, such as depth and orientation, may give more insight into the motion gesture and offer a possibility to improve the accuracy and robustness of recognition. Moreover, gestures are no longer limited to planar motions if full spatial tracking results are available. Any type of motion can be considered a gesture so long as it can be differentiated from others.

There exists no published gesture dataset that has both a sufficiently large size and

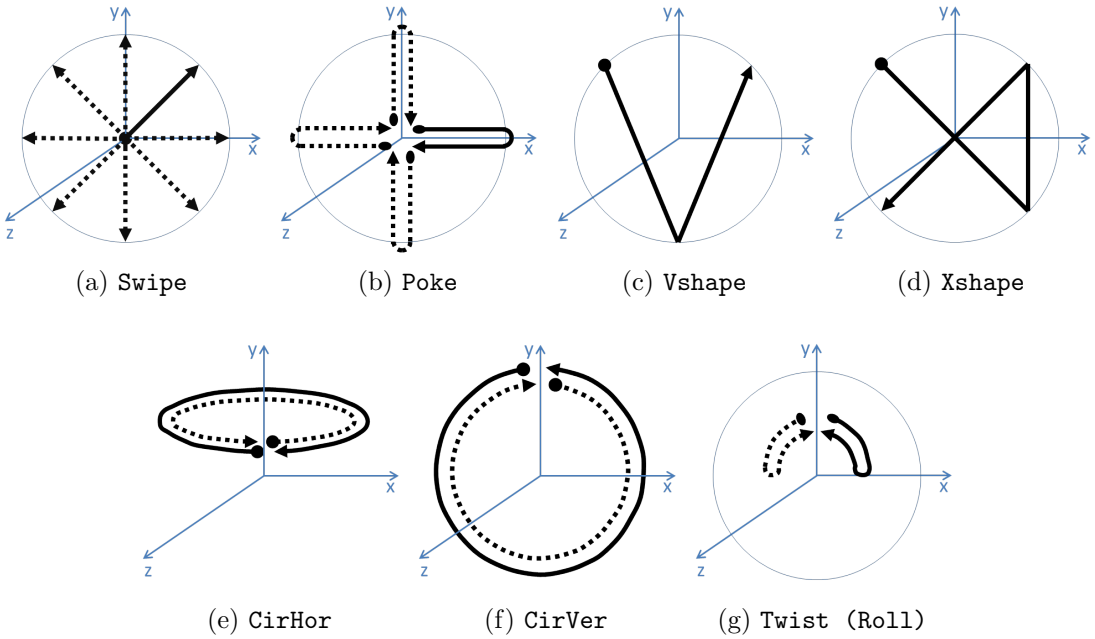
comprehensive motion information. In order to investigate the relative effectiveness of various tracking signals in motion gesture recognition, we build a 6-DOF motion gesture database (6DMG) that contains both explicit and implicit 6-DOF motion data from the hybrid tracking framework described in Section 3.1. It is interesting to understand which type of tracking signals and features help to describe the motion gesture. 6DMG makes it possible to compare the recognition performance over different tracking signals on a common ground.

In Figure 11, a set of 20 motion gestures is defined, including, swiping motions in eight directions (Figure 11a), swiping forth and back rapidly in four directions (Figure 11b), v-shape, x-shape, circle, and wrist twisting (Figure 11c-11g). These gestures are derived from 2D/3D spatial and gaming interactions, which are meant to be intuitive to perform and easy to memorize. The gesture set is actually not limited to what we have defined, but any distinguishable motions should work.

The recording apparatus is similar to using a remote in front of a TV as shown in Figure 12. While recording, the system always shows a virtual controller on the screen with a one-to-one motion mapping to provide real-time visual feedback. There is also a real-time playback function for the subject to review the recorded motion before it is stored into the database. The subject accepts or rejects the tentatively recorded gesture on the fly, which is more efficient than verifying the database offline at a later stage. Before recording, we first explain the basic functions of the controller to the subject, and let him or her play with the device. Once the subject is familiar with the control interface, we start the recording process described as follows:

1. The subject resets the origin to a location that is comfortable to start with;
2. The system selects a motion gesture from the set in Table 2 and briefly demonstrates it to the subject;
3. The subject presses and holds Button B during recording and releases the button upon termination of the trial;

4. After performing each trial, the subject reviews the playback and decides to save it or not;
5. After recording 10 trials, repeat Step 2 until all gestures in the set are recorded.



(a) `Swipe` (b) `Poke` (c) `Vshape` (d) `Xshape`

(e) `CirHor` (f) `CirVer` (g) `Twist (Roll)`

Figure 11: The illustration of 20 gestures in the database.

We recruited 28 participants (21 right-handed and seven left-handed, 22 male and six female, and ranging in age of 15 to 33) for recording, and each subject was asked to repeat each distinct gesture 10 times. There are in total 5600 gesture samples in 6DMG database. When recording, each subject was advised to perform the gesture in a consistent way, but no constraint was placed on his or her gripping posture, the gesture articulation style, range, and speed. Variations of the same gesture between individuals are expected, and recording motion gestures from different users ensures the in-class variability of 6DMG. The statistics on the duration of the 20 motion gestures in 6DMG are listed in Table 2, and we will explain the normalization ratio later.

Each recorded gesture contains comprehensive motion data, including position,

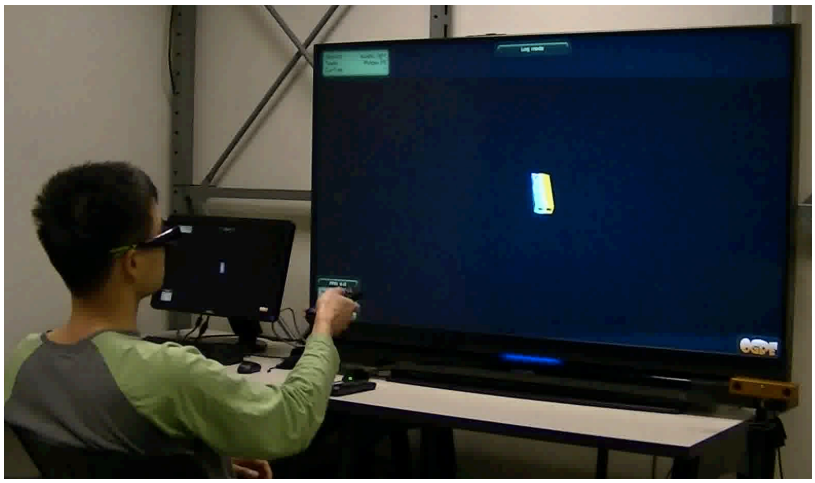


Figure 12: The gesture recording apparatus.

Table 2: The gesture list of 6DMG.

Figure	Name	Sample #		max/min norm. ratio				
		Avg.	Std.	P	O	V	W	A
11a	`SwipeRight`	51.9	20.7	8.7	5.9	14.5	12.4	24.1
	`SwipeLeft`	51.6	20.4	6.4	5.0	30.3	10.8	27.4
	`SwipeUp`	44.6	15.5	5.0	5.2	27.0	11.6	19.2
	`SwipeDown`	47.2	16.7	4.2	7.2	46.1	21.1	29.9
	`SwipeUpright`	45.2	16.9	5.4	5.6	17.7	16.6	31.7
	`SwipeUpleft`	44.9	17.5	5.5	3.9	14.6	17.0	36.2
	`SwipeDnright`	46.5	18.8	6.1	8.1	18.8	15.2	20.9
	`SwipeDnleft`	47.5	19.0	7.2	5.2	26.2	37.8	40.3
11b	`PokeRight`	70.9	23.0	4.2	4.5	15.4	11.2	16.5
	`PokeLeft`	74.5	25.1	4.4	4.7	16.0	18.4	19.3
	`PokeUp`	72.3	23.4	4.9	4.4	23.5	13.0	11.1
	`PokeDown`	71.0	24.9	5.0	5.5	18.1	17.4	20.4
11c	`Vshape`	71.6	23.7	4.4	4.9	9.7	16.0	11.4
11d	`Xshape`	99.3	28.0	4.0	4.2	9.1	11.7	13.8
11e	`CirHorClk`	104.3	27.0	3.5	4.2	9.0	7.3	9.1
	`CirHorCclk`	103.1	30.0	3.6	4.0	9.0	10.6	7.2
11f	`CirVerClk`	108.3	33.0	3.8	5.8	13.4	8.9	19.2
	`CirVerCclk`	102.4	32.0	3.8	6.5	8.4	8.6	13.6
11g	`TwistClk`	63.2	18.9	8.8	3.3	8.6	4.2	6.5
	`TwistCclk`	64.5	18.9	11.4	2.7	6.7	4.0	10.0

orientation (in quaternion), acceleration, and angular speed. The data can be useful to investigate motion gesture recognition with various dimensions of tracking signals. The 6DMG database can be a handy platform for researchers and developers to build their recognition algorithms and a common test bench for performance comparison. The 6DMG database and accompanying example programs, including the viewer, loader, and exporter, are available at `http://www.ece.gatech.edu/6DMG`.

4.2 Statistical Feature-based Linear Classifier

Rubine's feature set was originally designed for 2D trajectories using a mouse or a stylus [56]. Hoffman et al. [27] adapted Rubine's feature set to the implicit 6-DOF domain with an underlying assumption to treat the acceleration and angular speed as position in a 3D space. After a close look at the signals of the inertial sensors, the "trajectory" in the acceleration space is very jerky and far from the geometric concept that Rubine's feature set was originally designed for. Therefore, a running average with a span of five points was used to smooth the acceleration and angular speed before feature extraction. The tracking results of the position and orientation are much smoother, so filtering is unnecessary.

The feature set derived from the spatial trajectory is introduced first. For simplicity, we use the notation $[x, y, z]^\top$ for the spatial trajectory, which can be either the explicit $[p_x, p_y, p_z]^\top$ or the implicit $[a_x, a_y, a_z]^\top$, and x_i denotes the i_{th} sample in a gesture. The first feature f_1 is the gesture duration. The following features f_{2-13} are the maximum, minimum, mean, and median values of x, y, and z, respectively. f_{14} is the diagonal length of the bounding volume. The step distance and angles in

xy- and xz-planes are defined as follows:

$$\Delta x_i = x_i - x_{i-1} \quad \Delta y_i = y_i - y_{i-1} \quad \Delta z_i = z_i - z_{i-1}$$
$$d_i = \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}$$
$$\theta_i = \arctan(\frac{\Delta x_i \Delta y_{i+1} - \Delta x_{i+1} \Delta y_i}{\Delta x_i \Delta x_{i+1} + \Delta y_i \Delta y_{i+1}})$$
$$\gamma_i = \arctan(\frac{\Delta x_i \Delta z_{i+1} - \Delta x_{i+1} \Delta z_i}{\Delta x_i \Delta x_{i+1} + \Delta z_i \Delta z_{i+1}}).$$

The features relating the angles in xy- and xz-planes or the traveled distance can be derived as follows:

$$f_{15} = (x_3 - x_1)/\sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}$$
$$f_{16} = (y_3 - y_1)/\sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}$$
$$f_{17} = (z_3 - z_1)/\sqrt{(x_3 - x_1)^2 + (z_3 - z_1)^2}$$
$$f_{18} = (x_N - x_1)/\sqrt{(x_N - x_1)^2 + (y_N - y_1)^2}$$
$$f_{19} = (y_N - y_1)/\sqrt{(x_N - x_1)^2 + (y_N - y_1)^2}$$
$$f_{20} = (z_N - z_1)/\sqrt{(x_N - x_1)^2 + (z_N - z_1)^2}$$
$$f_{21} = \sum_{i=2}^{N-1} \theta_i \quad f_{22} = \sum_{i=2}^{N-1} |\theta_i| \quad f_{23} = \sum_{i=2}^{N-1} \theta_i^2$$
$$f_{24} = \sum_{i=2}^{N-1} \gamma_i \quad f_{25} = \sum_{i=2}^{N-1} |\gamma_i| \quad f_{26} = \sum_{i=2}^{N-1} \gamma_i^2$$
$$f_{27} = \sum_{i=2}^{N-1} d_i \quad f_{28} = \max d_i^2$$
$$f_{29} = \sqrt{(x_N - x_1)^2 + (y_N - y_1)^2 + (z_N - z_1)^2}.$$

The sine and cosine of the starting angle in the vertical (xy) plane are f_{15-16}, and f_{17} is the sine of the staring angle in the horizontal (xz) plane. The third sample is chosen empirically based on the average duration of the gesture data to derive the

starting angle. f_{18-19} are the sine and cosine of the angle from the first to last point in the xy-plane, and f_{20} is the sine of the angle from the first to last point in the xz-plane. After that, f_{21-26} are the total angle traversed, the absolute value and the squared value of that angle in the xy- and xz-planes respectively. The last three features f_{27-29} are the total traveled distance, the maximum squared step distance, and the Euclidean distance between the first and the last point.

As for the rotational trajectory, the features for W^o and O^o are slightly different. The angular speed introduces another 12 features, f_{30-41}: the maximum, minimum, mean, and median values of w_y, w_p, and w_r, respectively. For the orientation (quaternion), f_{30-45} represents the maximum, minimum, median, and mean values of q_w, q_x, q_y, and q_z.

These features barely contain any temporal information. After a few test runs, we discovered that Hoffman's feature set leads to confusion between some pairs of mirroring gestures like `PokeRight` and `PokeLeft`, `PokeUp` and `PokeDown`, and `CirHorClk` and `CirHorCclk`. Obviously, the time series nature of a motion gesture is crucial for distinction here. We introduce extra features to incorporate the temporal information: the mean values of the first half, the second half, and the center one third of $[a_x, a_y, a_z]^\top$, the mean values of the first half of $[w_x, w_y, w_z]^\top$, and the mean values of the first and the second half of $[p_x, p_y, p_z]^\top$. These features are designed to describe the motion in different time windows at a very coarse time scale. The window selection is empirical and also depends on how complicated the gesture is defined. In general, the time derivative physical quantity usually requires temporal features at a finer scale, which explains why more temporal features are used to describe A^o than P^o and W^o than O^o.

After converting a motion gesture g into a feature vector f, a linear classifier is used for the initial investigation. Associated with each gesture class is a linear

evaluation function defined as follows:

$$v_c = w_{c0} + \sum_{i=1}^{F} w_{ci} f_i, \qquad 0 \leq c < C, \tag{2}$$

where F is the number of features and C is the total number of classes. The classification of g is the class index c that maximizes v_c. Please refer to [56] for details on training the weights w_c. The feature-based statistical classifier is presented here as a baseline, which is proven to be simple yet effective in [20]. However, with a full consideration of the spatio-temporal nature of motion gestures, the recognition task can be significantly improved.

4.3 Hidden Markov Model-based Classifier

Hidden Markov models are especially known for their application in temporal pattern recognition such as speech, handwriting, and gestures. The hybrid motion tracking framework gives us a set of features (observations) with kinematic meanings for the HMMs, including the position, velocity, acceleration, orientation, and angular speed. With these features, a motion gesture can be represented as a spatio-temporal pattern. Each underlying state in the HMMs actually has a particular kinematic meaning and describes a subset of this pattern, i.e., a segment of the motion. Because the motion gesture is an order-constrained time-evolving signal, the left-to-right HMM topology is found to be suitable and widely used [60, 41, 47, 4]. Although the transition between hidden states can be estimated, the physical motion transition is often blurred. If carefully selecting the number of states, it is unlikely to skip a segment of the continuous motion when rendering a gesture. Thus, skip transition is not considered in the HMM topology.

To make the recognizer scale and speed invariant, proper feature normalization is very important. The upper case letters without superscript denote the normalized feature, and the corresponding normalization procedure is explained as follows. Normalization of P^o, V^o, and W^o is straightforwardly accomplished by uniform linear

scaling, i.e., $P = s_p P^o$, $V = s_v V^o$, and $W = s_w W^o$, where the scaling factors are computed as:

$$s_p = \frac{1}{\max[d_x, d_y, d_z]}, \begin{cases} d_x & = \max(p_x) - \min(p_x) \\ d_y & = \max(p_y) - \min(p_y) \\ d_z & = \max(p_z) - \min(p_z) \end{cases} \tag{3}$$

$$s_v = \frac{1}{\max(\|V^o(i)\|)}, \qquad i = 1, 2, ...N \tag{4}$$

$$s_w = \frac{1}{\max[\max(|w_y|), \max(|w_p|), \max(|w_r|)]}. \tag{5}$$

The scaling factor is determined according to the physical meaning of the normalization target.

The device-wise acceleration A^o is actually a mixture of the acceleration of the gravity and the motion, which provides a very rough estimate of the partial orientation, i.e., pitch and roll. A^o cannot be scaled directly because of the gravity. In [4], the gravitational acceleration is compensated by subtracting the mean of A^o based on the assumption that the sensor heading is constant over the time of one recording. This apparently does not work in our case because the heading of the control device keeps changing during gesture articulation. Extra information, i.e., the orientation, is needed to remove the gravitational acceleration. Given O^o, we first convert the device-wise acceleration to the global coordinates and subtract the constant gravity g as shown in Equation 6.

$$[0, A^g(i)] = \vec{q_i} * \vec{a}_i * \vec{q_i}^{-1} - \vec{g}, \tag{6}$$

where $\vec{q_i} = O^o(i)$, $\vec{a}_i = [0, A^g(i)]$, $\vec{g} = [0, 0, 1, 0]^\top$, and $*$ denotes quaternion multiplication. Then, A^g is linearly scaled to obtain the normalized $A = s_a A^g$, where

$$s_a = \frac{1}{\max(\|A^g(i)\|)}, \qquad i = 1, 2, ..N. \tag{7}$$

The normalization of O^o is the most tricky one because the quaternion cannot be "scaled" directly. First, O^o is offset (rotated) by the starting orientation so that the

first orientation becomes a unit quaternion, and then we convert the quaternion into the axis-angle representation as

$$[\cos\frac{\alpha_i}{2}, \vec{r}_i \sin\frac{\alpha_i}{2}] = O^o(i) * O^o(1)^{-1}, \tag{8}$$

where α_i is the angle rotated about the axis $\vec{r}_i$ by the right-handed rule. Even though the absolute orientation may provide extra information to distinguish gestures, it is only consistent and usable within a single user. For example, `SwipeRight` rendered by different users can be from the center to right with a rotation angle of 30 degrees or from left to right with an angle of 120 degrees. The concept here is to normalize the rotation angle α and keep the axis $\vec{r}$ untouched, i.e., scale the rotation amount without changing the rotation direction. However, there is one important limitation of the orientation representation: the rotation direction (or angle) is not unique. For example, rotating 0.5π and -1.5π around the same axis $\vec{r}$ have different rotation directions but result in an identical orientation. This is exactly the physical interpretation of $\vec{q}$ and $-\vec{q}$: they are different quaternions but represent the same orientation. Given an orientation, the true rotation amount and the direction to normalize are unknown. The ambiguity cannot be resolved unless we keep track of the evolving orientation.

In 6DMG, the orientation starts at identity quaternion when the tracker is pointing forward and facing upright. The orientation is updated with a delta rotation at every sampling instant to ensure its continuity. In other words, the evolving orientation is tracked, and the quaternion represents the true rotation direction and angle in the range from 0 to 2π. The ambiguity still arises when the angle exceeds 2π, but fortunately this never happens in 6DMG. We then scale the rotation angle and compute the normalized orientation as follows:

$$O(i) = [\cos\frac{s_\alpha \alpha_i}{2}, \vec{r}_i \sin\frac{s_\alpha \alpha_i}{2}] \tag{9}$$

$$s_\alpha = \frac{\alpha_{max}}{\max(\alpha_i)}, \qquad i = 1, 2, ...N. \tag{10}$$

α_{max} indicates the maximum rotation angle after normalization and is determined empirically. The distribution of the maximum rotation angles of all gestures in 6DMG is analyzed. The median, mean, and standard deviation are 0.47π, 0.48π, and 0.18π respectively. Thus, $\alpha_{max} = 0.5\pi$ is considered a reasonable choice here. The gesture recognition is tested with two values of α_{max}: 0.5π and π. The performance shows almost no difference, and $\alpha_{max} = \pi/2$ gives an insignificantly better accuracy ($<$ 0.1%).

Given recordings of the same gesture, the ratio of the maximum over the minimum scaling factor of the normalized features, i.e., the normalization ratio, can be a good indicator for "scale" variations. The normalization ratio of each gesture rendered by all subjects is listed in Table 2. In general, the normalization ratios of P and O are much smaller than those of the time-derivative features A, V, and W. The only exception is the normalization ratio of P for twisting gestures, in which the spatial trajectory is already expected to be nondeterministic. In the user-dependent case, the normalization ratios for all features basically fall under 3, which means limited variation. Therefore, the normalization process should be more helpful for the user-independent case due to its huge in-class variations as shown in Table 2. Note that feature normalization is no elixir. The concept of scale invariance reduces the in-class variations, but it may backfire if the variations between classes are reduced too much at the same time. For example, P^o of `TwistClk` and `TwistCclk` tends to be small in scale, which is an important clue to distinguish them. After normalization, their non-deterministic spatial trajectories are scaled up and may cause confusion with other gestures. In such case, P may be less discriminative than P^o.

4.4 Performance Evaluation

In this section, we first evaluate the performance of the HMM-based recognizer in both user-dependent and user-independent cases, including the evaluation of normalization, combined feature sets, and adaptation to stripped-down motion tracking. Leave-one-out cross validation [36] is used to further investigate the effect of handedness and HMM structures on motion gesture recognition. Lastly, the performance is compared with the baseline, the statistical feature-based linear classifier.

For the HMM-based recognizer, the Hidden Markov Model Toolkit (HTK)[1] is used for modeling, training, and testing. Although different topologies can be specified per gesture according to its complexity, the same topology is used for all gestures for generality. We experiment with HMMs of four, six, and eight states and one single Gaussian component per state (in the general mixture density form).

For the user-dependent recognition experiment, we form the training set with five samples randomly drawn from each gesture of a single user and use the remaining five samples for testing. The experiment is repeated 50 times for each of the 21 right-handed users for cross validation. For the user-independent case, the training set is formed from the gesture data of randomly selected five right-handed users. The testing sets are the gestures of the remaining 16 right-handed (UI.R) and seven left-handed users (UI.L) respectively. This is equivalent to training the recognizer in advance with only right-handers and having new right- or left-handed users simply come in and use the system. The experiment is repeated 200 times to calculate the average recognition rate. The same initial seed is used to randomize the combination of selected training samples so that the results are reproducible and comparable across different feature sets. The size of the training set is chosen intentionally to provide an estimate of achievable performance with very limited training data, which is motivated

[1]The Hidden Markov Model Toolkit (HTK) is available at `http://htk.eng.cam.ac.uk/`

Table 3: The recognition rates with and without normalization of user-dependent (UD) and user-independent (UI.R) cases.

	States	P^o	P	O^o	O	V^o	V	W^o	W	A^o	A
UD	4	95.88	96.23	97.45	97.51	97.84	97.88	96.78	97.26	97.58	97.03
UD	8	97.57	97.83	98.54	98.76	98.20	98.54	97.71	98.10	98.54	98.40
UI.R	4	85.13	87.38	64.42	85.32	73.64	87.65	63.75	75.40	62.15	80.33
UI.R	8	88.72	88.62	72.55	88.88	82.05	91.31	69.83	80.79	71.51	88.58

by the fact that it is time consuming to collect a lot of gesture data for general users. We separate the right- and left-handed testing sets to investigate the effect of handedness on gesture recognition.

4.4.1 Evaluation of Normalization

First, the effectiveness of the normalization of the five basic features (P , V, O, A, and W) is investigated in both user-dependent (UD) and user-independent (UI) cases. The average recognition rates from HMMs of four states and eight states are listed in Table 3. The UI.R in Table 3 shows the results of the right-handed testing sets. In the user-dependent case, the normalization only slightly improves the performance (and decreases in A), but it has significant impact in the user-independent case. This actually confirms the hypothesis that the normalization helps when large in-class variations exist. Note that in the normalization of A^o, O^o is needed to remove the gravity, which means the embedded partial rotation information is also removed. Thus, the recognition rate of A is slightly worse than A^o in the user-dependent case.

In Table 2, it is shown that the time-derivative features V, W, and A have higher normalization ratios than features P and O. Thus, the time-derivative features benefit more from the normalization process than P and O as shown in Table 3. The performance of P even slightly falls behind P^o in the eight-state HMM case, where the ambiguity caused by the normalized `TwistClk` and `TwistCclk` outweighs the gain of in-class variation reduction. Note that the ratio of O only takes into account the

"scaling" of rotation angles and doesn't include the variation of the absolute orientation, i.e., the staring orientation. Therefore, the large improvement of O (from 64.4% to 85.3%) is partially contributed by the orientation offset (+15.0%) and the rotation angle normalization (+5.9%) with four states one Gaussian mixture HMMs. Increasing the number of states also gradually improves the performance, but the gain is less prominent for the features that already achieve high accuracy.

Second, the discriminative power of these five basic feature sets is compared. In the user-dependent case, V achieves the highest accuracy (98.5%), and surprisingly P is the lowest (97.8%). In the user-independent case, V still performs the best (91.3%), but W becomes the worst (80.8%). Based on the results, even though the motion gestures are mostly defined by the spatial trajectory, each of the five basic feature sets is effective to a certain degree to distinguish the motion gesture.

4.4.2 Evaluation of the Combined Feature Sets

After showing the effectiveness of the normalization process, we evaluate the recognition performance with different combinations of these basic features, including the explicit spatial 3D (PV), implicit 6D (AW), explicit 6D (PVO), and complete 6D ($PVOW$ and $PVAOW$). The normalization of A^o actually requires the orientation O^o. Therefore, AWO can be considered the full feature set when only inertial sensors are available, which can be the case in a smartphone. These combinations of features correspond to the possible available tracking signals. The average recognition rates are plotted in Figure 13.

After putting together the position and orientation information, either the implicit or explicit 6D feature sets achieves over 99% accuracy in the user-dependent case. In the user-independent case, the performance of the implicit 6D feature set degrades more than the explicit 6D (PO and PVO), and the complete 6D feature sets still achieve the best accuracy. Adding the orientation to the implicit 6D leads to

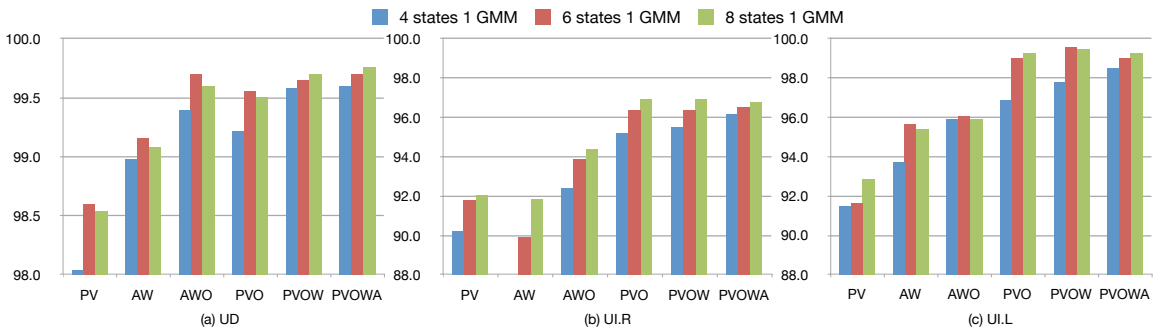


Figure 13: The recognition rates of combined feature sets. (a) UD: user-dependent case. (b) UI.R: user-independent case on right-handed users. (c) UI.L: user-independent case on left-handed users.

significant improvement, and AWO should be the best feature set for a pure inertial tracking system based on our findings.

When combining feature sets of different kinematic meanings, more "constraints" are tied to each HMM state and make it more discriminative. However, the improvement becomes marginal at certain level. In general, adding more HMM states can better captures the time seriers details of motion signals and helps to model the motion gesture, but it may suffer from the overfitting problem especially with the combined feature sets.

4.4.3 Adaptation to Stripped-down Motion Tracking

In addition to the combined feature sets above, we also investigate the recognition with limited motion information that reflects a special case of the tracking system in practice: 2D optical tracking when only one camera is used. Let $\hat{P}$ and $\hat{V}$ denote the 2D projection of P and V onto the image plane. For simplicity, the z (depth) component is truncated to form $\hat{P}$ and $\hat{V}$. This is very close to placing the camera in the front center of the user with negligible perspective projection. Three new feature sets are derived, $\hat{P}\hat{V}$, $\hat{P}\hat{V}OW$ and $\hat{P}\hat{V}OWA$. The experiment is repeated with these new feature sets, and the results are shown in Table 4. The results of AW and AWO are also listed for comparison. Note that only the results of the right-handed testing

sets are shown in the user-independent case (UI.R).

Table 4: The recognition rates of combined feature sets of user-dependent (UD) and user-independent (UI.R) cases.

	States	PV	$\hat{P}\hat{V}$	$PVOW$	$\hat{P}\hat{V}OW$	$PVOWA$	$\hat{P}\hat{V}OWA$	AW	AWO
UD	4	98.04	97.04	99.58	99.43	99.60	99.47	98.98	99.40
UD	8	98.54	98.19	99.70	99.73	99.76	99.76	99.08	99.60
UI.R	4	90.24	92.22	95.52	94.81	96.19	95.80	87.30	92.41
UI.R	8	92.05	94.46	96.96	96.17	96.80	96.69	91.86	94.38

In the gesture set, only `HorCirClk` and `HorCirCclk` are not defined on the vertical (xy) plane, but their 2D projections are still distinguishable from other gestures. In some cases, the z dimension is less meaningful and sustains large variations among different subjects. For example, the z increases monotonically if `SwipeRight` is rendered from center to right, but it may decrease then increase if rendered from left to right. Thus, losing the z dimension should not affect the recognition much in our gesture set. In Table 4, it is shown that $\hat{P}\hat{V}$-related feature sets performs slightly worse than their full spatial 3D counterparts in general. The only exception is that $\hat{P}\hat{V}$ outperforms PV in the user-independent case. When only the spatial trajectories are available, the variation in z can be misleading and degrades the performance. However, the z dimension is still essential if the gesture definition covers the whole 3D space, e.g., swiping forward or backward.

4.4.4 Leave-one-out Cross Validation

It is interesting that the accuracy of the left-handed testing set in Figure 13(c) is higher than that of the right-handed testing set in Figure 13(b), even though the recognizer is trained with right-handed data. This may have resulted from the unbalanced size of testing sets (16 right-handed versus seven left-handed). Based on the results, we assume that the proposed HMM-based recognition of motion gestures is handedness-independent. To verify this assumption, PV, AWO, and $PVOWA$ are chosen as the representative feature sets for leave-one-out cross validation on the

Table 5: The recognition rates of leave-one-out cross validation over different HMM structures.

states	GMM	total mixture #	PV	AWO	$PVOWA$
4	1	4	94.55	96.66	98.21
4	2	8	95.34	97.36	98.27
4	3	12	95.63	(93.50)	98.13
6	1	6	95.80	**97.38**	98.39
6	2	12	95.52	(93.90)	98.29
8	1	8	95.73	**97.38**	**98.48**
8	2	16	**96.05**	(97.09)	98.25

whole database.

It is shown that more HMM states can better model the motion gesture and improve the performance in previous experiments. With leave-one-out cross validation, the largest training set from 6DMG is attained, i.e., 270 samples per gesture (27 users×10 trials per gesture), which allows HTK to train more complicated HMMs. Therefore, we further investigate the modeling capability of multiple Gaussian mixtures per state upon the original experiment setting. The average recognition rates of leave-one-out cross validation are shown in Table 5. The best result of each column is boldfaced. The training processes of AWO with four states three GMM, six states two GMMs, and eight states two GMMs partially fail in HTK, so their results are marked with round brackets. To solve this problem, it requires more training data, which is not possible for the current setting. In Table 5, it is shown that the pure inertial feature set AWO outperforms the optical (spatial) only feature set PV, and the complete feature set $PVOWA$ achieves the best performance. Motion information beyond a spatial trajectory indeed provides additional insight to the motion gesture and improves the performance.

After dividing the leave-one-out testing set by handedness, the left-handed group still has higher average and smaller standard deviation of the recognition rate than the

right-handed group for all HMM settings. For example, the average (and standard deviation) of the 21 right-handers and seven left-handers are 97.76% (3.25%) and 99.57% (0.56%), respectively, with four states, one GMM per state, and $PVOWA$. This confirms the assumption that the HMM-based recognizer is handedness-independent, and the performance difference results from the intrinsic variations in the database.

The confusion matrix of the leave-one-out cross validation with eight states, one GMM per state, and $PVOWA$ is shown in Figure 14. The confusion between **g01** (`SwipeRight`), **g05** (`SwipeUpright`), and **g07** (`SwipeDnright`) reflects the fact that some users tend to render the diagonal gestures very close to `SwipeRight`. Similarly, confusion arises between **g02** (`SwipeLeft`), **g06** (`SwipeUpleft`), and **g08** (`SwipeDnleft`). With the feature set PV, confusion arises between **g19** (`TwistClk`) and **g20** (`TwistCclk`), e.g., 88.6% and 85.7% accuracy with eight states and one GMM per state, which can be solved by introducing the orientation-based features.

Compared with Table 4, more training data significantly improve the performance for the same HMM structure, i.e. four, six, eight states with single Gaussian mixture. In general, using more states in HMM still improves the recognition rate, but the gain becomes less prominent as can be seen in Table 5. On the other hand, using more Gaussian mixtures per state improves the performance when the HMM topology is very simple, i.e., four states with single Gaussian mixture per state. The time series nature of motion gestures is better captured by more states in HMM. Even single Gaussian mixture works well enough to model the probability distribution within each state. When considering the HMM structures of the same number of total Gaussian mixtures, using more states instead of more mixtures per state tends to be a better strategy.

The effect of HMM structures on motion gesture recognition has been shown. The optimal number of states and mixtures per state actually depends on the gesture set. Therefore, fine-tuning the optimal HMM structure should be done on a case-by-case

	Recognition Result																				
	g01	**g02**	**g03**	**g04**	**g05**	**g06**	**g07**	**g08**	**g09**	**g10**	**g11**	**g12**	**g13**	**g14**	**g15**	**g16**	**g17**	**g18**	**g19**	**g20**	**acc (%)**
g01	267				2		6		4				1								95.4
g02		251				13		14							2						89.6
g03			274		1	4					1										97.9
g04				271								3	4	2							96.8
g05					275		1								4						98.2
g06		21				247		2							10						88.2
g07	3			1			266						7	3							95.0
g08		2						278													99.3
g09	1								276						3						98.6
g10		1								276					1	2					98.6
g11											280										100.0
g12				2								277		1							98.9
g13													280								100.0
g14														280							100.0
g15															268	1	11				95.7
g16															1	275		4			98.2
g17															3		277				98.9
g18														1				279			99.6
g19																1			279		99.6
g20																				280	100.0

Figure 14: The confusion matrix of leave-one-out cross validation with eight states, one GMM, and $PVOWA$, where g01 to g20 are the gestures from top to bottom in Table 3.

basis and beyond the scope of this dissertation.

4.4.5 Comparison with the Baseline

The statistical feature-based classifier is used as the baseline for performance comparison. Due to the defined statistical features in Section 4.2, not all combinations of tracking signals are available. The recognition results are obtained with features extracted from either implicit or explicit 6D motion data. In the HMM case, the best corresponding feature sets derived from implicit and explicit 6D are AW and PVO. The comparison between the statistical feature-based linear classifier and the HMM-based recognizer with eight states and single Gaussian mixture per state is shown in Table 6. In the user-dependent (UD) case, the performance is almost the same. In the right-handed user-independent (UI.R) case, the HMM-based recognizer outperforms by 6.6% for implicit 6D and 3.4% for explicit 6D in the absolute recognition rate.

Table 6: The comparison between the statistical feature-based linear classifier and the HMM-based recognizer.

	Implicit 6D		Explicit 6D	
	Linear	HMM	Linear	HMM
UD	98.80	99.08	99.59	99.51
UI.R	85.24	91.86	93.51	96.93
UI.L	78.58	95.43	96.99	99.29

In the user-independent case with left-handed testing set (UI.L), the HMM-based recognizer still achieves better performance. In general, the left-handed testing set yields higher accuracy than the right-handed set, which probably results from the unbalanced size of testing sets. Note that the linear classifier with implicit 6-DOF features particularly has much worse performance on the left-handed set than the right-handed one. In such a case, we postulate that the handedness makes a difference to a certain level for the implicit statistical features.

Our study gives an insight into the attainable recognition rate with different tracking devices. Two approaches for motion gesture recognition are presented: the statistical feature-based linear classifier as a simple baseline and the HMM-based recognizer that takes account of the spatio-temporal nature of gesture signals. The effectiveness of various features derived from different tracking signals is also compared in both user-dependent and user-independent cases.

Overall, the statistical feature-based linear classifier can achieve 85.2% and 93.5% accuracy with implicit and explicit 6-DOF data. The HMM-based recognizer has higher recognition rates, 91.9% and 96.9% respectively. In addition to better performance, the HMM-based recognizer also works with more flexible feature combinations and in general keeps the accuracy above 96%, which means flexibility in choosing the tracking technologies. Based on the results, motion gesture recognition benefits from the complete 6D motion information. Robust motion gesture recognition is achievable even for the challenging user-independent case.

CHAPTER V

AIR-HANDWRITING RECOGNITION

Motion gestures are meant to be simple and limited in numbers, so a user can easily memorize and perform them. To ensure the usability of motion gestures, the expressive power of gestures for complicated operations is often curtailed. One way to expand the functionalities of motion-based control is through air-handwriting recognition that enables the user to input text by "writing". Air-handwriting is especially useful for user interfaces that do not allow the user to type on a keyboard or write on a trackpad/touchscreen.

Air-handwriting refers to the style of writing characters or words in the free space. Different from conventional pen-based handwriting, air-handwriting is rendered on a virtual plane without visual or haptic feedback. Similar to motion gestures, the user can write in the air regardless of eye sight. The main difference is that motion-based writing is continuous without the pen-up/pen-down information, i.e., all strokes and characters are connected. The amount of fine motor control for writing in the air is less than that for writing on a rigid surface. As a result, air-handwriting tends to be messy in shape, especially when there is no immediate visual feedback of the writing trajectory. Recognition of air-handwriting presents a new challenge due to the lack of engagement information and the large variability in motion signals compared to the conventional handwriting.

In this chapter, we first describe how air-handwriting is different from conventional handwriting and the challenging aspects of air-handwriting recognition.

We start from air-handwriting rendered with a handheld device and elevate to controller-free air-fingerwriting rendered with a finger tip. The writing motion is

captured by different tracking technologies (controller versus hand-free), which also affect the affordable methods for writing segmentation. In Section 5.2, we elaborate the modeling and recognition of air-handwriting with the 6-DOF motion tracking system. In Section 5.3, we propose an method for automatic detection of writing segments, and build a complete air-fingerwriting system with detection and recognition stages. Lastly, we conduct usability study of air-handwriting as a text input method in Section 5.4.

5.1 A Unique Writing Style

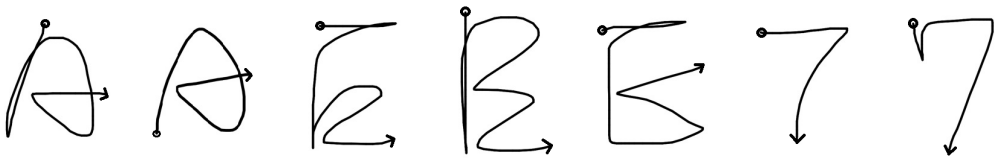
A handwriting is define by its 2D spatial trajectory. To be more specific, it is the shape that defines a character, and the temporal information is not necessary for character recognition, e.g., optical character recognition (OCR) of scanned images of handwritten or printed text. However, the temporal information, i.e., the stroke or ink information, is proven to be substantially helpful to handwriting recognition. Similar to motion gestures, air-handwriting is tracked with a continuous stream of sensor data, which means the writing rendered in the air is uni-stroke without any pen-up and pen-down information. The user forms a visionary model to write on an imaginary plane without haptic feedback. If no visual feedback is provided, the user can still write in the air. Air-handwriting consists of two levels: motion characters and motion words. Although it is feasible, text input of sentence-level with air-handwriting is not considered or recommended due to the usability issue as discussed in Section 5.4.

5.1.1 Motion Characters

Motion characters are isolated alphanumeric letters and written in one continuous stroke. We can view a uni-stroke motion characters as a gesture with a "standard" definition. In our design, we do not consider the modified uni-stroke styles of alphabet, such as Graffiti [9], because it requires extra learning for a novice user. Instead, we

simply treat a naturally written character or letter as a uni-stroke pattern without pen-up and pen-down motions since they are connected when written in the air.

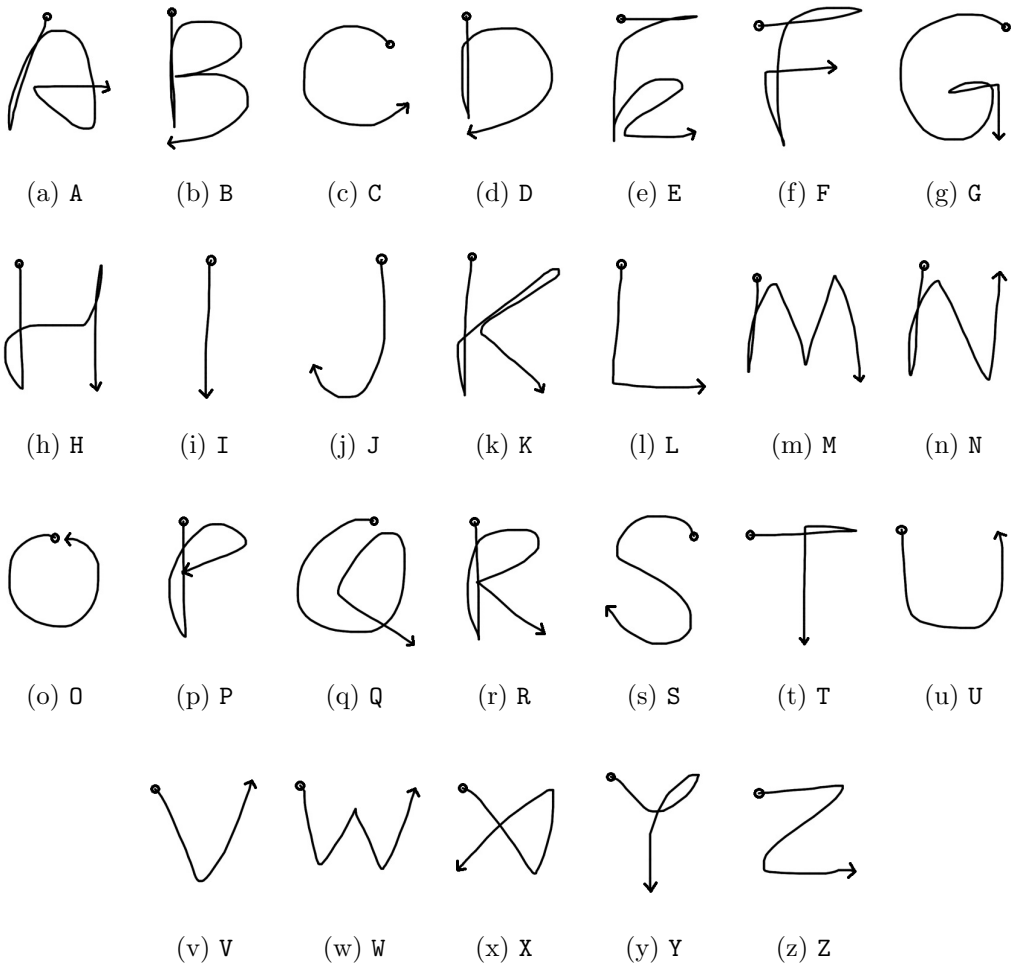
In the perspective of spatio-temporal signals, different stroke orders and allographs would result in completely different patterns for the same motion character (see Figure 15). Therefore, we should model the possible stroke orders and allographs separately and assign the same class label to all the associated models. In order to do so, sufficient motion data from different subjects are needed to have a broad coverage of possible allographs and variations in stroke orders. The recording process is very time consuming, but the data itself can be of great merit for the research community and a great addition to the existing 6DMG database.



(a) `A` (b) `A` (c) `E` (d) `E` (e) `E` (f) `7` (g) `7`

Figure 15: The spatial trajectories of motion characters: (a) and (b) two stroke orders for uppercase `A`; (c) to (e) three stroke orders for uppercase `E`; (f) and (g) two allographs for number 7.

To solve the air-handwriting recognition problem without loss of generality, we only consider uppercase letters A to Z in this dissertation. To simplify the recording process, we instructed the subject to follow the suggested "stroke order" for each character to avoid allographs or different stroke orders. Hence, we will have 26 models for motion characters. In Figure 16, we illustrate the uni-stroke writing trajectory of uppercase letter A to Z. Once the proof-of-concept air-handwriting recognition is done, we can add lowercase letters, allographs, and variations in stroke orders by collecting more data.



(a) A (b) B (c) C (d) D (e) E (f) F (g) G

(h) H (i) I (j) J (k) K (l) L (m) M (n) N

(o) O (p) P (q) Q (r) R (s) S (t) T (u) U

(v) V (w) W (x) X (y) Y (z) Z

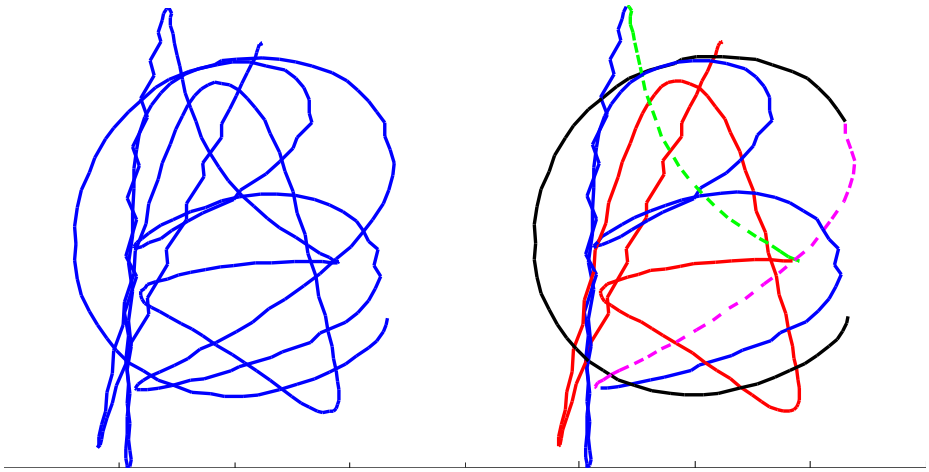
Figure 16: Illustration of the uni-stroke writing of isolated uppercase letters. The circle symbol indicates the starting point, and the arrow means the writing direction.

5.1.2 Motion Words

Performing recognition with models at the word level is impractical unless the vocabulary is restricted to a small size. Instead, a motion word is formed by connecting motion characters or letters orthographically with ligature motion in-between. The modeling of ligature is critical for motion word recognition. Given the huge variations of motion rendered in space, several models for ligatures are needed. We will elaborate the character and ligature modeling in Section 5.2.3.

When there is no haptic or visual feedback, the ordinary left-to-right writing style can be hard to keep without overlap or shape distortion. In our preliminary experiment, we discovered that users tend to shrink and overlap the last few characters of a word when running out of "writing space" due to the limited arm reach. Therefore, we consider a box writing style which is more suitable for air-handwriting: the user overlays characters in a virtual box. The projected trajectory of an air-handwriting of `ABC` is shown in Figure 17a. It is difficult even for a human to recognize the overlapped handwriting. To better illustrate the example, we manually segment the motion word and show the results in Figure 17b. The dash lines are the ligature motions that connect characters `A`, `B`, and `C`. When the user writes freely in the air without visual feedback of the writing trajectory and the virtual box, it is hard to keep air-handwriting neat, which makes the recognition even more challenging.

Air-handwriting is quite different from conventional handwriting due to the box-writing style and the lack of stroke information. Moreover, we track air-handwriting with 6-DOF motion data, which is also different from the conventional 2D spatial trajectory of pen-based writing. In our case, features derived for traditional handwriting recognition cannot be applied directly. Among related works of air-handwriting, Amma et al. [5] had a similar box-writing style that is tracked with only inertial sensors. To our best knowledge, we are the first to evaluate air-handwriting recognition with 6-DOF motions.



(a) ABC (b) segmented ABC

Figure 17: 2D projected trajectory of a motion word ABC.

5.2 Air-handwriting Recognition

We accomplish air-handwriting recognition [17] with comprehensive motion data using the hybrid motion tracking system of optical and inertial sensors described in Section 3.1. A system diagram of air-handwriting recognition is shown in Figure 18. The control motion is tracked with 6-DOF motion data and explicitly segmented by a manual push-to-write operation. The writing segment is then passed to the recognition stage and outputs the recognized handwriting. The recognition stage includes feature extraction, normalization, and the HMM-based recognition kernel.

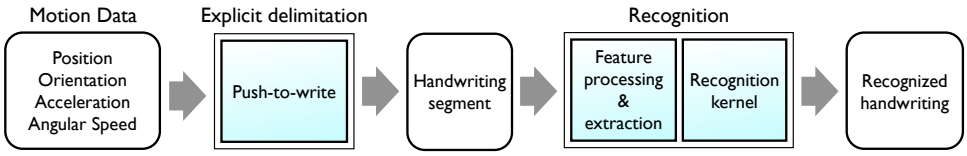


Figure 18: A system diagram of air-handwriting recognition.

5.2.1 Data Recording with 6-DOF Motion Tracking

In our design, a user holds the controller to write in the air with the 6-DOF motion tracking system introduced in Section 3.1. It is convenient to utilize the buttons on the controller to segment handwriting motions. The use of push-to-write explicitly

provides the starting and ending points of a uni-stroke sequence, regardless of its content, and we focus on the recognition problem.

The recording apparatus is identical to the motion gesture recording in 4.1. We first explain the box-writing style and specific stroke orders to the subject and start recording after the subject is familiar with air-handwriting through practice. Each subject first records isolated motion characters and then motion words. The user holds Button B to start writing and releases it when a character or a word is finished. The resulting motion character (see Figure 16) or word (see Figure 17) itself is in a single continuous stroke, and we don't have the boundary of letters within a motion word. The subject has to review the playback of the recording right away, and the recording is saved only if the specific writing style and stroke order are followed.

We recruited 22 participants (all right-handed, 17 male and 5 female) to record air-handwriting data. Each subject was advised to write in a consistent manner, but we did not constrain his or her gripping posture of the controller, the writing scale or speed. The only rules are the box-writing style, the stroke order for each letter, and the push-to-write scheme.

Each isolated motion character (A to Z) was recorded 10 times by every subject. For motion words, we select 40 words from common television channels, e.g., `ABC`, `CNN`, `FOX`, and common digital/Internet services, such as `TV`, `MUSIC`, and `GOOGLE`. The shortest word has two characters, and the longest one is `DISCOVERY`. See Appendix A for the whole list of the 40-word vocabulary. This vocabulary covers all 26 characters, and the average number of characters per word is four. Every subject records each word five times. The whole recording process (26 characters and 10 words) takes about one hour. In addition to the 40-word vocabulary, we create another $1k$-word vocabulary, which includes the most frequent 1000 two-letter and three-letter words and three-letter prefixes from the Google Web 1T 5-gram data set. There is no intersection between the two vocabularies. The 1000-word set was recorded without

Table 7: The statistics of durations of motion characters (in number of samples)

	avg	std		avg	std		avg	std
A	159.5	37.4	J	60.5	12.1	S	92.7	17.8
B	156.7	37.1	K	136.5	26.6	T	88.6	16.4
C	77.3	19.8	L	64.8	15.1	U	73.5	14.1
D	118.7	24.9	M	146.2	29.6	V	67.2	11.5
E	190.8	48.6	N	115.7	21.1	W	110.4	19.3
F	132.6	27.4	O	85.1	17.4	X	91.3	16.1
G	149.7	35.1	P	107.6	20.3	Y	105.7	20.5
H	137.5	29.9	Q	119.4	26.4	Z	94.1	18.6
I	42.8	10.3	R	134.9	24.5			

repetition only by subject M1, which took about three hours. Our air-handwriting data set has 5720 motion characters and 5400 motion words in total[1].

5.2.2 Feature Processing

From the 6-DOF motion data, we derive the same features (observations) as in motion gesture recognition: position P and velocity V from optical tracking, orientation O, acceleration A, and angular speed W from inertial tracking. The notation is also identical to that in Section 4.

The writing style and speed vary among users. In Table 7, we list the statistics of the writing duration (in number of samples) of motion characters from 22 subjects. We see the large variation in writing speed among different users and the difference in writing time among different characters. Similar to motion gesture recognition, the feature normalization is also critical in making air-handwriting recognition scale and speed invariant. We use uppercase letters with superscript o and without superscript to denote the raw and the normalized features, respectively.

In Section 4.3, we have explained the feature normalization for 6-DOF motion gestures, which requires to offset P^o and O^o by the starting position and orientation of a motion gesture. We cannot apply the same normalization procedure to isolated

[1]The air-handwriting data, vocabulary, viewer, loader and exporter are available at http://www.ece.gatech.edu/6DMG

motion characters because the resulting characters will have the starting point at the origin and the starting orientation at the unit quaternion. In other words, the resulting character model will not be centered in the same "virtual box" and will cause problems when we concatenate character models to form a word model. Therefore, we modify the normalization process for P^o and O^o to address the offset issue. First, P^o and O^o are offset to account for variations in the position and orientation of the "virtual box".

$$\begin{aligned} \tilde{P}(i) &= P^o(i) - \vec{p}_{cen} \\ \tilde{O}(i) &= O^o(i) * \vec{q}_{cen}^{-1}, \end{aligned} \tag{11}$$

where $\vec{p}_{cen}$ is the center point of the bounding volume of P^o, $\vec{q}_{cen}$ is the normalized quaternion of $[\bar{q}_w, \bar{q}_x, \bar{q}_y, \bar{q}_z]$, i.e., the arithmetic mean of O^o, and $*$ denotes quaternion multiplication. The arithmetic mean of quaternions servers as a reasonable approximation for the "center" of the range of orientation. Normalization of $\tilde{P}$ is straightforward with uniform linear scaling, i.e., $P = s_p\tilde{P}$, where s_p scales the longest edge of the bounding volume of $\tilde{P}$ to unit length as in Equation 3.

To normalize $\tilde{O}$, we need to convert the quaternion into an axis-angle representation expressed as,

$$[\cos\frac{\alpha_i}{2}, \vec{r}_i \sin\frac{\alpha_i}{2}] = \tilde{O}(i), i = 1, 2, ..N, \tag{12}$$

where α_i is the angle rotated about the axis $\vec{r}_i$ by the right-hand rule. We would like to normalize the rotation angle and keep the rotation axis intact. Our system tracks the evolving orientation by updating the orientation with a delta rotation at every sampling instant. Thus, Equation 12 defines the true rotation direction and the angle in the range from 0 to 2π. We can scale the rotation angle and compute the normalized orientation as in Equation 9 and 10. In Equation 10, α_{max} indicates the maximum rotation angle after normalization and is set to 0.2π empirically for air-handwriting. Features V, W, and A do not suffer from the offset issue; we use the same normalization process as in motion gesture recognition reported in Section 4.3.

5.2.3 Air-handwriting Modeling

A writing motion can be represented as a spatio-temporal pattern, which is suitably modeled by an HMM. Each underlying state in an HMM describes a subset of the writing motion with a particular kinematic meaning. A left-to-right HMM is suitable for order-constrained time-evolving signals. Skip transitions are not considered because it is unlikely to skip a segment of the continuous motion in air-handwriting.

One advantage of HMM is the scalability to assemble the model of a complicated pattern from the models of several unit blocks. In the air-handwriting case, the unit blocks are the models of motion characters and ligatures. Here we define the ligature as the motion from the end point of the preceding character to the start point of the following character. We will address air-handwriting modeling on two levels: motion characters and motion words.

HMMs of motion characters are trained directly from the isolated A-to-Z recording. Because allographs and different stroke orders are already excluded, we create one model for each character. As shown in Table 7, the duration of each letter varies substantially, and we assign the number of states for each character correspondingly. For example, the most complicated letter `E` has 18 states, and short letters `I` and `J` have only eight states. We choose single Gaussian mixture per state for all motion characters and empirically determine the number of states for each character based on its average duration (see Table 8).

The HMM for a motion word is formed by connecting the models of motion characters with ligature models. Ligatures are simple and short in duration, so they are modeled with only three states and a single Gaussian mixture per state. The ligature models are context dependent, and 26 upper case letters would result in 676 (26×26) models. It is nonetheless difficult to obtain enough data to cover all combinations to train the ligature models. Therefore, we have to cluster similar ligatures into fewer groups to make the modeling process feasible.

Table 8: The number of states of each motion character.

	state #		state #		state #
A	14	J	8	S	10
B	16	K	12	T	10
C	10	L	10	U	10
D	14	M	12	V	10
E	18	N	12	W	14
F	14	O	10	X	10
G	12	P	12	Y	10
H	12	Q	12	Z	10
I	8	R	16		

Table 9: Manual clusters for start and end points of characters

(a) Start point

S1	BDEFHKLMNPRTUVWXYZ
S2	AIJOQ
S3	CGS

(b) End point

E1	BDSX
E2	ITY
E3	CEGHKLMQRZ
E4	JP
E5	AF
E6	O
E7	NUVW

Based on P of the first and last states of the character HMMs, we heuristically cluster the start and end points of upper case A to Z into three and seven groups, respectively, in Table 9. The clustering reduces the number of ligature models to 21 (7×3). We label every ligature given the preceding and the following characters and build the model for a motion word correspondingly. For example, the HMM for ABC can be constructed as A·lig_E5S1·B·lig_E1S3·C, where "·" denotes concatenation. Note that the occurrence of each ligature in our data set is not equally distributed.

Although the hard clustering in Table 9 is reasonable, clustering ligatures with a data-driven decision tree can be more precise. By asking questions about the connecting previous or next letter of each ligature, the decision tree attempts to find those contexts which make the largest difference to distinguish clusters. Each question splits the current pool of training data into two sets and increases the log likelihood

with the use of two sets rather than one. We branch the decision tree by selecting the question that maximizes the increase of log likelihood, and repeat the process until the increase of log likelihood achieved by any question at any node is less than the threshold.

In Table 9, we manually clusters characters according to the position of the start and end points. Based on the hard clustering, we generate several general questions, e.g., *"does the previous letter belong to* `E1`*"* and *"does the next letter belong to* `S2`*?"* To take into account both the end-point position and the stroke direction, we further divide the hard clustering to create more detailed questions, e.g., *"is the previous letter* `B` *or* `D` *(ends at bottom left with a right-to-left stroke)"* and *"is the next letter* `I` *or* `J` *(starts at top center with a top-to-bottom stroke)?"* We list the complete question set for the data-driven decision tree in Table 10.

Table 10: Question set for the data-driven decision tree.

	Is the previous letter?
Q1	BDSX
Q2	CEGHKLMQRZ
Q3	NUVW
Q4	AF
Q5	BD
Q6	C
Q7	ELZ
Q8	GHM
Q9	ITY
Q10	JS
Q11	KQR
Q12	NU
Q13	O
Q14	P
Q15	VW
Q16	X

	Is the next letter?
Q17	BDEFHKLMNPRTUVWXYZ
Q18	AIJOQ
Q19	A
Q20	BDHKLMNPRU
Q21	CGS
Q22	EFTZ
Q23	IJ
Q24	OQ
Q25	VWXY

Because the decision tree is data-driven, the questions asked and the resulting clusters vary upon the training data. Besides, not all questions in Table 10 will be asked during the branch process. With the help of the decision tree, we are able to

synthesize unseen ligatures in the training data and generate models for all possible ligatures. For example, the HMM for `ABC` becomes `A·lig_AB·B·lig_BC·C`.

From the recordings of isolated motion characters, we train the HMMs of motion character and use them to initialize the character part in a motion word model. We do not have or need the letter-level segmentation in a motion word. With the composite word HMM, segmentation (alignment) of characters and ligatures can be simultaneously accomplished with recognition. Because the motions of characters and ligatures usually blend together, the ground truth of segmentation can be vague. To have a better understanding of ligature motions, we manually segmented all the motion words in the 40-word vocabulary recorded by subject `M1`. The segmented ligatures are manually clustered as in Table 9 and used to train the isolated ligature HMMs. To re-estimate the embedded ligature models, using the isolated ligature HMMs for initialization is proven to work better than initializing with zero means and global variances.

Given the initial HMMs of isolated characters and ligatures, we synthesize the HMM for each word in our vocabulary and perform embedded Baum-Welch re-estimation [7, 33] on all recordings in the training set. We take the re-estimated results as the initial values for the next iteration of re-estimation. Iterative re-estimation allows the trained models to converge to the training data. However, repeated re-estimation may lead to over-training if the models become too closely matched to the training data and fail to generalize well on unseen test data. In practice, around two to five iterations of embedded re-estimation are usually sufficient for training. We use the improvement of the overall log likelihood per frame of the training data as a termination condition for repeated re-estimation and stop after the third iteration of embedded re-estimation.

To this point, we have the refined HMMs of characters and 21 hard clustered ligatures, which are used to generate the decision tree. Because the decision tree is

data driven, the resulting ligature clusters may vary in different training sets. After clustering with the decision tree, two more iterations of re-estimation are performed to obtain the final HMMs of characters and decision-tree-clustered ligatures. The re-estimated character and ligature HMMs are the building blocks of the decoding word network for motion word recognition.

For HMM modeling, training, and testing, we still use the Hidden Markov Model Toolkit (HTK). The air-handwriting recognition is studied in the user-independent case with leave-one-out cross validation. We choose one subject as the testing set and train the models with the remaining 21 subjects. This procedure is repeated for every subject to obtain the average results of air-handwriting recognition.

5.2.4 Motion Character Recognition

We evaluate motion character recognition with the five basic features (P,V,O,A,W) and different combinations of them (PV, AWO, $PVOWA$). These combinations of features actually correspond to different motion tracking devices. PV is the feature set derived purely from optical tracking, and AWO can be considered the full feature set from inertial measurements. $PVOWA$ uses all the available data from the hybrid 6-DOF motion tracking system. A motion gesture can be defined in a 3D space, but handwriting is actually defined on a 2D surface regardless the true writing motions. Therefore, we also investigate the feature $\hat{P}$ and $\hat{V}$, which are the 2D projection of P and V onto the vertical (xy) plane. We consider $\hat{P}$ and $\hat{V}$ as the tracking results from a single camera placed right in front of the air-handwriting plane, i.e., similar to the stripped down optical tracking in Section 4.4.3.

The HMMs of motion characters are trained and tested with isolated characters, and we show the character error rate (CER) of leave-one-out cross validation with different features in Table 11. First, we compare the discriminative power of the basic features. The explicit 6D features (P, V, and O) outperform the implicit 6D features

Table 11: The character error rate (CER) of motion character recognition

features	CER (%) average	std
P	3.72	(3.60)
V	6.12	(2.88)
O	3.81	(5.05)
A	7.97	(7.38)
W	7.92	(3.34)
PV	1.61	(2.16)
AWO	1.84	(2.37)
$PVOWA$	1.05	(1.23)
$\hat{P}$	3.88	(3.55)
$\hat{V}$	6.15	(2.69)
$\hat{P}\hat{V}$	1.61	(2.06)
$\hat{P}\hat{V}OWA$	1.05	(1.33)

(A and W). The projected 2D features ($\hat{P}$ and $\hat{V}$) have slightly higher error rates than the 3D ones. Combining features of different kinematic meanings makes each HMM state more discriminative and improves the recognition rate. The improvement of combining features becomes less prominent at certain level.

Although a character is only defined by its 2D spatial trajectory, features of different kinematic meanings prove to be informative to distinguish motion characters written in the air. We can achieve robust motion character recognition even with AWO. The CER of pure inertial tracking is slightly higher than the CER of pure optical tracking, and $PVOWA$ achieves the lowest CER. The performance of motion character recognition confirms with our study on motion gesture recognition in Section 4.4.

5.2.5 Motion Word Recognition

We perform motion word recognition with two approaches: word-based and letter-based recognition. First, we explain the pros and cons of these two approaches. We then evaluate the recognition results and propose other techniques to further improve the performance.

For word-based word recognition, we synthesize the HMM for every word in the vocabulary and construct the decoding word network as shown in Figure 19. Given a vocabulary of N words, word-based recognition is formulated as a one-out-of-N problem, i.e., the recognition result is the path with the highest score among the N possible paths. The recognition is done at the unit of one word and hence more robust to individual letter errors within a word. However, word-based recognition requires the user to finish the whole word before recognition and cannot handle words that are out of vocabulary (OOV).

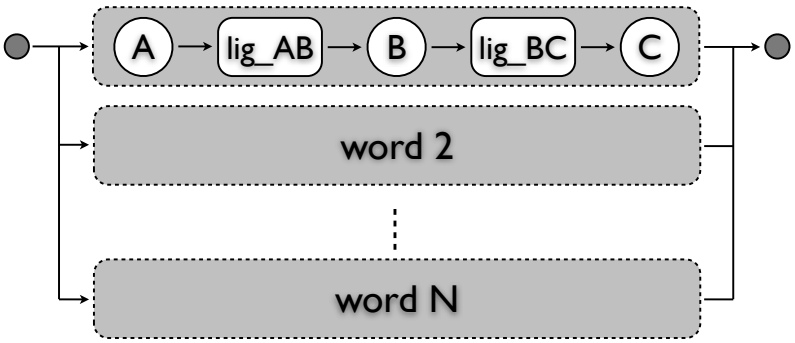


Figure 19: Decoding word network for word-based word recognition.

On the other hand, word recognition can be done on a letter basis. In Figure 20, we illustrate a simplified example of a letter-based decoding network that is built with letter A and B and the corresponding ligature models. The letter-based decoding network allows arbitrary decoded letter sequences and can handle out-of-vocabulary words. The recognition result is the arbitrary path that is legitimate in the letter-based network and achieves the highest score. Another advantage of letter-based word recognition is to allow progressive decoding while the user is writing, unlike the word-based recognition which requires the user to complete a word. The freedom of arbitrary sequences comes at a price of weakened contextual information from the vocabulary. Under such circumstances, ligatures become the only affordable contextual information, so we need more precise ligature models. To further improve the recognition performance, we utilize contextual information from the vocabulary by

introducing a language model on the letter sequence. The decoded letter sequence will depend on both the writing motion and the probability conditioned on the preceding letters. In Figure 20, we show how to embed the conditional probabilities of a bigram language model in the transition arcs from character nodes to ligature nodes.

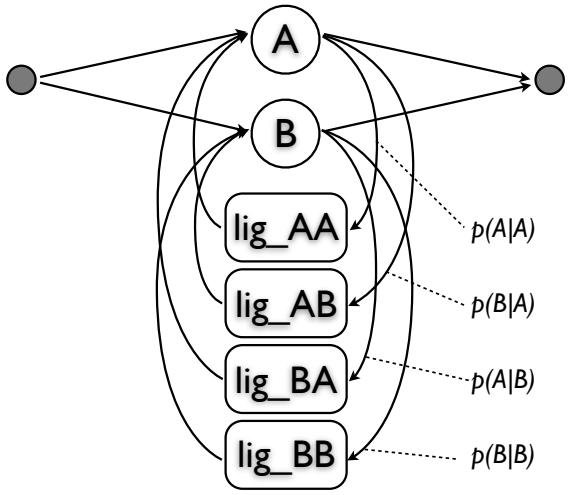


Figure 20: Decoding word network for letter-based word recognition.

5.2.5.1 Word-based Motion Word Recognition

For word-based word recognition, we use the refined HMMs of characters and 21 hard clustered ligatures to build the decoding word network as shown in Figure 19. In the word-based decoding network, each path is a word model synthesized from corresponding character and ligature HMMs, and the letter sequences are tightly restricted to the vocabulary. The word-based word recognition is formulated as a one-out-of-N problem, where N is the vocabulary size. We can view the ligature models as fillers that absorb the transition motions between characters. In our preliminary experiment, we tried a single filler model for all ligatures and still obtained a reasonable recognition performance. This is because the word-based recognition aggregates all the motion clues of a word to make a final decision out of the vocabulary taking full advantage of the letter sequence constraints. Refining ligature models with clusters improves the precision of character/ligature segmentation within a word but

Table 12: Results of motion word recognition on the 40-word vocabulary and 22 subjects

features	word-based WER (%)		letter-based (backoff) WER (%)		letter-based (backoff) CER (%)	
	average	std	average	std	average	std
PV	0.045	(0.144)	10.59	(6.63)	3.48	(2.67)
$\hat{P}\hat{V}$	0.023	(0.104)	9.20	(5.36)	2.86	(1.82)
AWO	0.0	(0.0)	14.93	(11.11)	5.70	(4.86)
$PVOWA$	0.0	(0.0)	11.57	(8.23)	4.15	(3.48)
$\hat{P}\hat{V}OWA$	0.0	(0.0)	10.61	(7.31)	3.65	(2.82)

may not affect the overall word-based recognition much. However, the single filler model fails in letter-based decoding, and we will discuss the need of more accurate ligature modeling later.

The first experiment is the leave-one-out cross validation on 22 subjects with the 40-word vocabulary described earlier. We only focus on the combined features. The average word error rates (WER) are listed in Table 12, which indicates the recognition performance of the user-independent case. Note that there is no CER in the word-based word recognition. The combined feature sets all perform very well, e.g., the WER is 0.045% (= 2/4400) for PV.

In the second experiment, we test the scalability of the word-based decoding network. The models for characters and ligatures are trained with the 40-word vocabulary recorded by all subjects except `M1`, and we use the $1k$-word vocabulary recorded by `M1` for testing. The synthesized word-based decoding network actually contains both vocabularies, i.e., 1040 words in total, and the results are shown in Table 13. The WER of AWO becomes the worst, and the WER of PV is slightly lower than $PVOWA$, which achieves a WER of 0.9% corresponding to nine errors out of 1000 words.

If we remove the factor of user variations and only look at the results of subject `M1` as the testing set in the first experiment, the WER is zero for all feature sets. It is not surprising that a larger vocabulary incurs more ambiguity of similar words and make

Table 13: The results of motion word recognition on the $1k$-word vocabulary and subject 'M1'

features	word-based WER (%)	letter-based (backoff) WER (%)	letter-based (backoff) CER (%)
PV	0.80	1.90	0.66
$\hat{P}\hat{V}$	0.80	2.80	0.97
AWO	1.60	7.00	2.59
$PVOWA$	0.90	4.10	1.42
$\hat{P}\hat{V}OWA$	0.90	5.00	1.73

Table 14: The average WER (%) of different designs of letter-based motion word recognition on the 40-word vocabulary and 22 subjects

features	decision-tree	decision-tree + bigram	decision-tree + bigram + 2-best	decision-tree + bigram (w/o backoff)
PV	17.27	10.59	4.73	2.73
AWO	23.09	14.93	8.16	2.75
$PVOWA$	15.16	11.57	5.77	2.18

the recognition more challenging. However, the word-based word recognition still achieves fairly low WER, e.g., the highest WER is 1.6% for AWO. The word-based word recognition is appealing for applications that require text input of a limited vocabulary and demand high accuracy.

5.2.5.2 Letter-based Motion Word Recognition

We choose the refined HMMs of characters and decision-tree-clustered ligatures to build the letter-based decoding word network as shown in Figure 20. In our preliminary experimen with 22 subjects and the 40-word vocabulary, use of ligature models clustered by a decision tree achieves about 2% absolute WER reduction over hard clustered ones. With the decision-tree-based ligatures as the only contextual constraint, we show the average WER of leave-one-out cross validation of 22 subjects in Table 14, which has a similar trend as the results of motion character recognition in Table 11.

To further improve the recognition performance, we utilize the statistics of letter

sequences in the vocabulary. We estimate the bigram language model for the 40-word and $1k$-word vocabulary separately. Good-Turing discounting and backoff are applied to compute the probabilities of unseen bigrams [35]. Each ligature model depends on its previous and next characters, so we embed the conditional probabilities of the bigram language model into the transition arcs from characters to ligatures as in Figure 20. We also need to adjust the weight of likelihoods between the language model and the motion models, i.e., characters and ligatures. Hence, a language model scale factor is introduced, which post-multiplies the language model likelihoods from the word lattices. We empirically set the scale factor to 15 in the decoding network. Fine tuning the scale factor depends on both the feature set and the accuracy of the language model to be discussed later.

We show the average WER and CER of leave-one-out cross validation with the bigram language model in Table 12. The CER of letter-based word recognition is computed as follows,

$$\text{CER} = \frac{S + I + D}{N_c}, \tag{13}$$

where S, I, and D are the counts of substitution, insertion, and deletion errors at the character level, and N_c is the total number of characters. For example, if a word `QUIZ` is wrongly decoded as `OLUIZ`, there are one substitution error and one insertion error out of four characters and one word error.

In Table 12, the pure inertial AWO still has the highest error rates. It is interesting that the pure optical PV outperforms the complete 6D $PVOWA$ with the help of the language model. Also, the 2D projected $\hat{P}\hat{V}$ and $\hat{P}\hat{V}OWA$ achieve lower error rates than their 3D versions. The bigram language model significantly improves the recognition performance, e.g., the absolute WER reduction is 5.61% for PV, 7.43% for AWO, and 3.71% for $PVOWA$ (see Table 14).

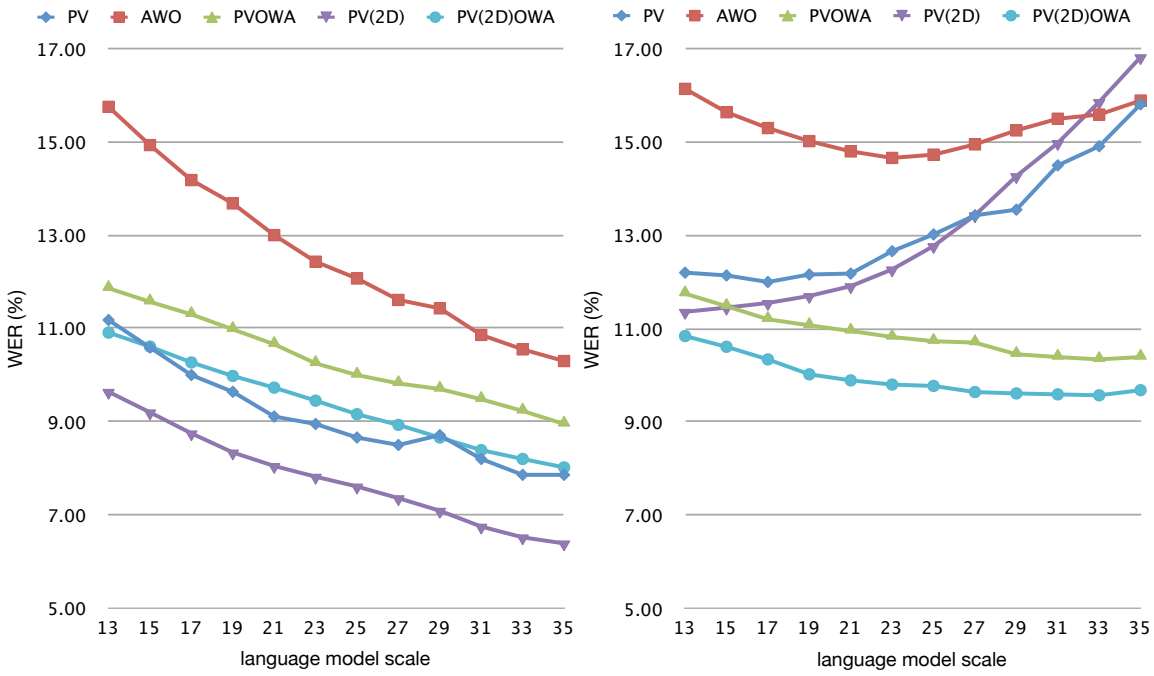
Technically, there is no scalability issues for letter-based word recognition. The

only difference is the estimation of the language model, which depends on the vocabulary. We show the recognition results for subject `M1` and the $1k$-word vocabulary in Table 13. For a fair comparison, we also list the WER of subject `M1` in the first experiment (40-word vocabulary and leave-one-out cross validation) as follows: 2% ($= 4/200$) for PV and $\hat{P}\hat{V}$, 4.5% for AWO, and 3% for $PVOWA$ and $\hat{P}\hat{V}OWA$. In Table 13, the WER is slightly higher for the $1k$-word case. However, more handwriting data is needed to draw a more rigorous conclusion on how the vocabulary size affects the word recognition.

By increasing the scale factor of the language model, the decoding word network emphasizes more on the language model than the motion models. We evaluate the WER of leave-one-out cross validation on 22 subjects with different scale factors and two language models estimated from the 40-word and $1k$-word vocabularies, respectively. The results are shown in Figure 21. In Figure 21a, we can see the WER decreases as the scale factor increases when the testing set is drawn from the same vocabulary as the one used to estimate the language model. In Figure 21b, emphasizing more on the language model may hurt the recognition performance because the language model, estimated from the $1k$-word vocabulary, does not truly represent the bigram statistics of the testing set.

The dimension of the feature set (observation vector) also matters for the language model scale factor. The log likelihood of motion models is accumulated along every dimension of the feature set. In general, a feature set of higher dimension results in larger log likelihood (not proportional) of the motion model part, but the likelihood of the language model part remains the same. Therefore, we need a larger scale factor to achieve roughly equivalent weighting between the language model and the motion models.

From Table 12 and 13, there is a performance gap between word-based and letter-based word recognition. We investigate two potential techniques that can further



(a) language model (40-word vocabulary) (b) language model (1k-word vocabulary)

Figure 21: The average WER (%) of letter-based word recognition of leave-one-out cross validation on 22 subjects with different scale factors and the language model estimated from (a) the 40-word vocabulary and (b) the 1k-word vocabulary.

improve the letter-based word recognition and show the comparison in Table 14.

n-best Recognition After examining the recognition results, the common character errors are usually due to ambiguity of letters, e.g., `O` and `C`, `D` and `P`, `W` and `N`. Another common character error results from similarity of sub-letters, e.g., `E` is wrongly decoded as `FZ`. If a letter sequence is wrongly decoded, the correct one usually has a likelihood close to the best one. In n-best recognition, a correct recognition means one of the top n hypotheses matches the ground-truth label. In general, the recognition errors can be roughly halved for 2-best recognition. With the 40-word vocabulary and 22 subjects, the WER of $PVOWA$ is reduced from 11.57% to 5.77% for 2-best recognition and 2.89% for 5-best recognition. We show the WER of 2-best recognition in the second column from the right of Table 14. At the point of view of

system design, it is helpful to provide a list of n-best recognition results for the user to choose the right one.

Restrictive Bigram Language Model It is possible to further reduce the search space of the decoding network by applying a more restrictive language model. Instead of backoff smoothing, zero probability is assigned to unseen bigrams, i.e., disabling the transition arcs of unseen ligature models in the vocabulary. In the last column of Table 14, the average WER for the 40-word vocabulary is significantly reduced to 2.73% for PV, 2.75% for AWO, and 2.18% for $PVOWA$. However, the absolute WER reduction is less than 0.5% for the $1k$-word vocabulary. As expected, the restrictive language model is more effective when the vocabulary spans a smaller set of bigrams (ligatures). We consider letter-based word recognition with a restrictive language model somewhere between the word-based word recognition (the strictest) and the letter-based word recognition with a complete bigram language model (the freest). The restrictive letter-based word network still allows progressive decoding but cannot handle out-of-vocabulary bigrams. If an application only accepts valid English words, the restrictive letter-based word recognition can be applicable for flexible text input.

5.3 Air-fingerwriting Detection and Recognition

In the previous section, we study the modeling and recognition of air-handwriting with different types of motion tracking data. With our 6-DOF motion tracking system, the air-handwriting is rendered by a handheld device. Writing with a hand or a handheld device through wrist and arm motion actually incurs more physical effort than writing with a finger. It's more comfortable for the user to rest the arm and write with a finger to avoid the so-called "gorilla arm" problem. As a motion-based interaction, fingerwriting can be faster and less fatiguing than writing with a handheld device or hand, particularly when the tracking system supports or requries short range motion

tracking. In this chapter, we will focus on writing with a finger in the air, i.e., air-fingerwriting.

Our current 6-DOF motion tracking system is not capable of finger tracking, so we choose the Leap, a controller-free optical hand tracking system, as the input device for air-fingerwriting. The Leap does finger-precision tracking and allows the user to write in the air easily with his or her fingertip. We can view air-fingerwriting as an extension of the proposed universal motion-based control.

In Section 5.2, we take advantage of the button on the handheld device of the tracking system and adopt the push-to-write scheme for word boundary segmentation. For the controller-free case, it is possible to replace a button (push-to-write) with a pinch gesture (pinch-to-write), but writing with explicit delimiters may still hinder the intuitiveness and user experience of air-fingerwriting. Therefore, we propose an algorithm that automatically detects and segments the writing part from the continuous tracking signal.

The air-fingerwriting problem is two-fold: detection and recognition. First, we need to distinguish handwriting from other finger movements on a motion-based user interface, such as cursor movements. Second, we have to perform air-fingerwriting recognition based on the detection results to evaluate the overall performance of the air-fingerwriting system. An overview of the air-fingerwriting system is shown in Figure 22. At the detection stage, detected writing segments are extracted from the continuous motion data. At the recognition stage, the detected segments are processed for the final result of recognition or rejection.

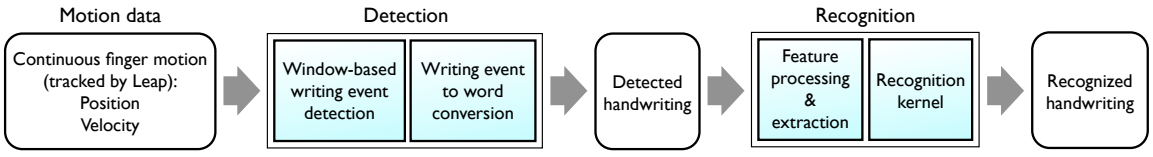


Figure 22: The system diagram for air-fingerwriting detection and recognition.

The rest of this section is organized as follows. In Section 5.3.1, we describe

the data recording procedures for air-fingerwriting with the Leap. We present the details of the proposed air-fingerwriting detection in Section 5.3.2 and our design of the recognizer in Section 5.3.3. Finally, we evaluate the system performance of air-fingerwriting in Section 5.3.4.

5.3.1 Data Recording with the Leap

Basically, the writing style of air-fingerwriting is identical to air-handwriting as introduced in Section 5.1. There are two major differences: the writing is rendered with a fingertip instead of a handheld device and the push-to-write paradigm is no longer applicable. In our design, air-fingerwriting involves no physical plane to write on and provides no haptic feedback or visualized writing trajectory. The box-writing style allows the user to write with mostly finger movements and a little bit wrist rotations. The range of the hand motion is minimized, and it reduces the effort required for air-fingerwriting.

For controller-free and glove-free hand tracking, we use the Leap[2], which is claimed to achieve a tracking precision of 0.01 mm. With our system setup (Intel core i7 CPU 2.66 GHz, Leap SDK 0.7.4 with USB 2.0 connection), the Leap tracks at a rate of 120 Hz. We place the Leap roughly 20 cm in front of the monitor so that the tracking volume covers the range of hand movements with the elbow resting on the desktop. The tracking coordinates of Leap are aligned to the monitor with x- and y-axes lying in the horizontal plane parallel to the screen. The positive x-axis points rightward, the positive y-axis points upward, and the positive z-axis points away from the screen (similar to Figure 6a). The origin of the coordinates is at the center of the Leap device.

The Leap SDK produces sampled data of position, velocity, and pointing direction of the "pointables", i.e., stick-like objects, within its view. There are other attributes

[2]Our Leap is a developer unit received from Leap Motion in January 2013, and the tracking performance may differ from the official release, which is not available at the time of this work.

of hands and gestures, which are still in the experimental stage. Here, we only use the relatively stable tracking results: the position and the velocity of the tip of a pointable. Because the Leap can only identify the fingers by the tracking history, we assign the finger closest to the screen as the pointing finger when loss of tracking occurs. Most of the time the tracking of the pointing finger is stable. The position of the pointing finger on the xy-plane is offset and linearly scaled to the pixel position of the cursor on screen, e.g., a finger movement of 32 cm corresponds to a movement of 1920 pixel on screen. To complete the basic functionalities of a 2D user interface, we implement clicking with a gesture of closing the thumb while the index finger points at the target. With our design, the user can use the fingers to do simple point-and-click task similar to using a mouse.

For data recording, we write a program that overwrites the mouse events on Windows with the finger motion. This program updates the tracking results from the Leap at 60 Hz, which shows no discernible delay in both writing and pointing-and-clicking operations. The recording program displays the word to write in a text box and has several buttons: `START` and `FINISH` in the center, and eight buttons, numbered from one to eight, around the center in a 3-by-3 grid. Figure 23 shows a screen shot of the recording program.

The recording procedure is as follows:

1. Click `START` to start recording.
2. Click one numbered button, which is randomly enabled after `START`.
3. Write the prompted word while pressing the `Ctrl` key.
4. Click one numbered button, which is randomly enabled after writing.
5. Click `FINISH` to stop recording.

Clicking the randomly enabled buttons in Step 2 and 4 introduces random cursor

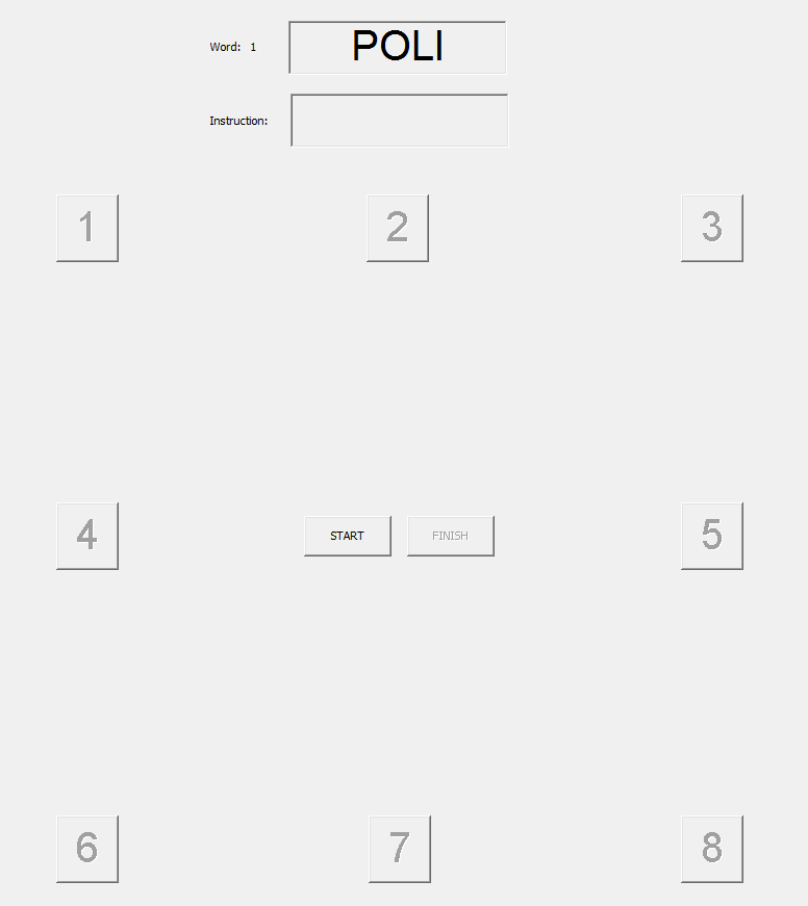


Figure 23: The screen shot of the recording program for air-fingerwriting.

movements in each recording. Pressing the `Ctrl` key (with the non-writing hand) provides the ground truth of writing segments. Note that the "ground truth" itself may contain imprecise segmentation due to the key operation. Each recording contains exactly a motion word with random motions before and after writing. In addition to the recording procedure, we ask each subject to write in a consistent way with the box-writing style. The writing position, scale, and speed are not constrained. To solve the air-fingerwriting problem without loss of generality and to expedite the data recording process, we only consider uppercase letters A to Z with a specific stroke order for each letter as shown in Figure 16. The subject needs to redo the recording of one word if the procedure or the specific stroke order is not followed.

We create a new $1k$-word vocabulary, which includes the most frequent 1000 two-,

three-, and four-letter words and four-letter prefixes from the Google Web 1T data set. Among the 1000 words, we carefully select 100 words as the common set, which covers 26 letters and 21 ligature types defined in Section 5.2.3. The remaining 900 words are shuffled and divided into 18 sets of 50 unique words. The special distribution of vocabulary helps us evaluate how user dependency and out-of-vocabulary words affect the recognition with limited user data as will be described later in Section 5.3.4.

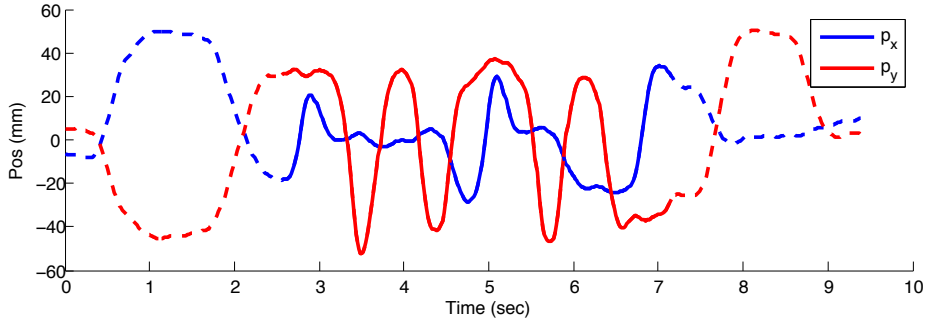
We recruited 18 participants (all right-handed, 13 male and 5 female) to record air-fingerwriting data. Eleven of them participated in our previous air-handwriting recording. Each subject wrote 150 words, which consists of the common set and one unique set. With 18 subjects, we collected a total of 2700 recordings that cover the $1k$-word vocabulary[3].

To give an idea of how the recording looks like, we plot the position and velocity in the xy-plane over time of the recording `TITL` by subject `C1` with the ground-truth label in Figure 24. We also show a 2D trajectory of a complete recording of the same recording in Figure 25a and the ground-truth writing segment in Figure 25b. Both position and velocity are smoothed with a 5-point moving average, and we offset the position in the y-axis to zero mean in Figure 24a for illustration purposes. In Figure 24, we observe that the signals of the writing part are different from those of the non-writing part with more frequent changes along time, which sheds some light on solving the detection problem.

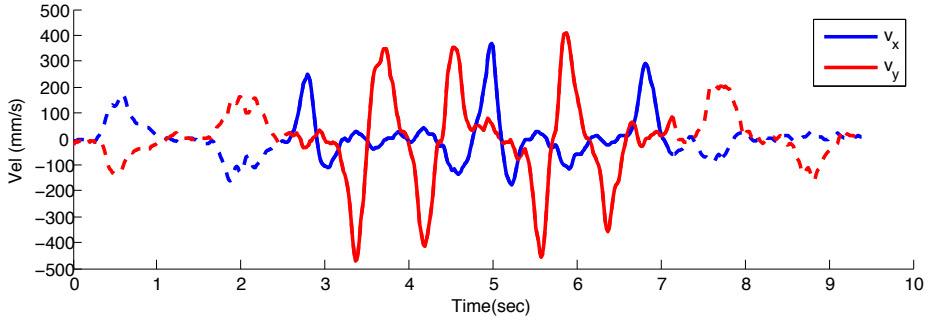
5.3.2 Air-fingerwriting Detection

Ideally, the detection algorithm should spot all handwriting segments (high recall), produce only a small amount of false alarms (high precision), and impose a minimal delay in the processing pipeline. Our first attempt to solve the detection problem is based on a quick distance-based classification of handwriting letters. The writing

[3]The air-fingerwriting data and vocabulary are available at `http://www.ece.gatech.edu/6DMG`

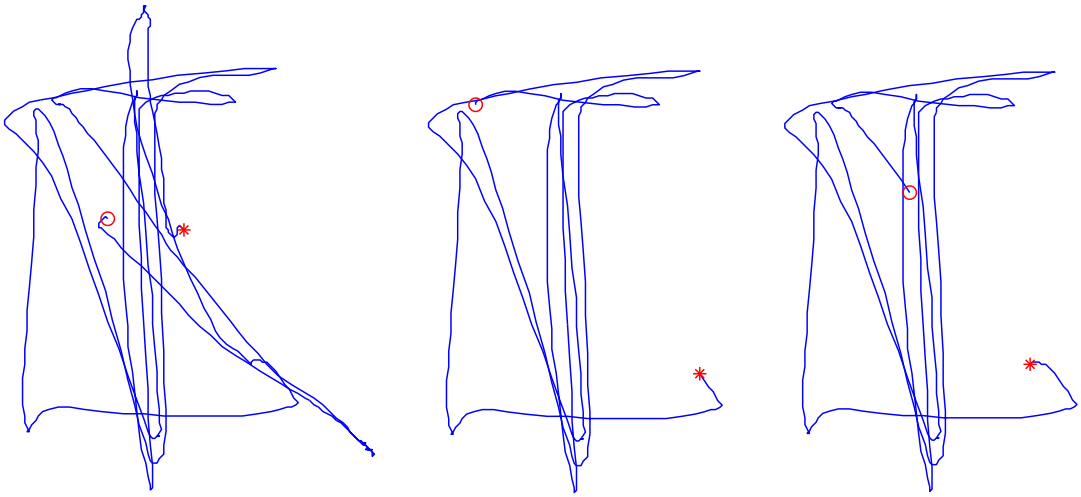


(a) position (p_x, p_y)



(b) velocity (v_x, v_y)

Figure 24: The recording `TITL` by subject `C1`. The solid lines are the "ground-truth" writing segment, and the dash lines are the non-writing parts.



(a) complete recording

(b) writing segment (ground truth)

(c) writing segment (detection)

Figure 25: The 2D trajectory of the recording `TITL` by subject `C1`. The circle sign is the start, and the star sign is the end.

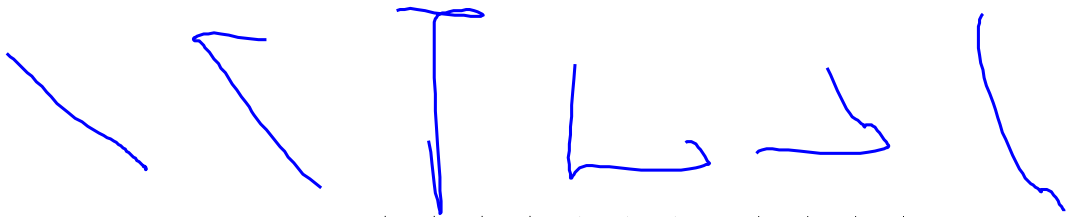
curve is converted to a Legendre series representation in real-time [24], and distance-based nearest neighbor classification is performed on the Legendre coefficients [25]. We compute the Legendre coefficients of several sliding windows of different window lengths and detect if there is a match for a letter. Classification based on the Legendre coefficients works quite well if the sliding window matches the correct letter segmentation. However, the Legendre coefficients are sensitive to the span of the sliding window. The classification is not robust in most cases where the sliding window doesn't correctly segment a letter. In our case, even an offset of five samples can distort the Legendre coefficients and leads to wrong classification.

5.3.2.1 Window-based Approach

Apparently, the approach of quick letter classification does not fit our need, so we choose the window-based approach instead. The window-based detector is only responsible for determining whether a writing event occurs in the window, which is slided through the continuous motion data. We form a potential writing segment by combining the detection results of all overlapping windows and pass it to the recognizer for further processing.

As we observe in Figure 24 and 25, a writing event usually involves sharp turns, frequent changes in directions, and complicated shapes rather than a drift or swipe motion. The sliding window has to be long enough to capture these writing characteristics to distinguish a writing event. However, a longer window means lower temporal resolution and introduces larger delay in the processing pipeline. In our case, we empirically choose a window length of 60 samples (1 sec) with a step size of 10 samples (167 ms). Before the window-based detection, it is important to smooth the motion data from the Leap with a 5-point moving average to remove jitters.

In Figure 26, we show the 2D trajectories of several sliding windows. It is clear that the windows in Figure 26c and 26d contain writing events, and the windows in



(a) i_s=021, w_c=0, ll=-65 (b) i_s=111, w_c=16, ll=-1.38 (c) i_s=161, w_c=60, ll=10.98 (d) i_s=381, w_c=50, ll=3.58 (e) i_s=401, w_c=30, ll=0.35 (f) i_s=421, w_c=10, ll=-4.29

Figure 26: The 2D trajectory of selected sliding windows from `TITL` by subject `C1`, where i_s denotes the first sample index of the window, w_c denotes the count of samples that are labeled as ground-truth writing, and ll denote the log likelihood of the writing event classification.

Figure 26a and 26f don't. The windows in Figure 26b and 26e are ambiguous with partial writing and non-writing events. The ground-truth labels and detection results of these example windows will be discussed later.

5.3.2.2 Writing Event Detection

The first step at the detection stage is to detect if a window contains a writing event. It is straightforward to determine a window that has tiny motion as a non-writing event, and we skip the "silent" window from further processing. Here, we define a "silent" window as *a*) both b_x and b_y, the edges of the bounding box of p_x and p_y, are smaller than 10 mm; and *b*) the velocity is smaller than 50 mm/s.

After "silence" suppression, we extract features from a non-silent window, which

can be derived as follows,

$$
\begin{aligned}
f_1 &= \sum |\Delta\theta_i| \\
f_2 &= \sum |\Delta\theta_i|^2 \\
f_3 &= b_x/(b_x + b_y) \\
f_4 &= ||\text{total travel distance}/\max(b_x, b_y)||_1 \\
f_5 &= ||\text{total travel distance}/\max(b_x, b_y)||_2 \\
f_6 &= s_2/s_1 \\
f_7 &= \#\text{ of zero crossings of } v_x \text{ and } v_y,
\end{aligned}
$$

where $\Delta\theta_i$ is the change in direction of p_x and p_y with a minimum step size of 5 mm; b_x and b_y are the edges of the bounding box of p_x and p_y; s_1 and s_2 are the eigen values (in descending order) of the point clouds of (p_x, p_y).

The angle features f_1 and f_2 capture the properties of sharp and frequent turns of handwriting. If the window contains a writing event, f_3 is likely to be around 0.5, and the normalized travel distances f_4 and f_5 become substantial. The ratio of eigenvalues (f_6) is an indicator of the shape complexity and tends to be close to one when the window contains a writing event. When computing the number of zero crossings of v_x and v_y (f_7), a threshold of ± 100 mm/s is used to avoid change in direction due to tremor or tracking noise.

Given the writing segment labeled by the subject, the ground-truth label for a window is determined as follows,

a) *writing*: the ground-truth writing segment spans more than 5/6 of the window

b) *non-writing*: the ground-truth writing segment spans less than 1/6 of the window

c) *mix*: otherwise

For example, Figure 26a and 26f are labeled as *non-writing*, Figure 26c and 26d are

labeled as *writing*, and Figure 26b and 26e are labeled as *mix*. Obviously, it is more ambiguous to classify a *mix* window as a writing or a non-writing event.

We use Gaussian mixture models (GMM) to achieve the binary classification of writing and non-writing events. To train and tune the classifier, only *writing* and *non-writing* windows are considered. We slide the window from the beginning of each recording and obtain around $7k$ *writing* windows and $5k$ *non-writing* ones. We use all of these windows to train and test the classifier because the preliminary results of k-fold cross validation do not show much difference.

The GMM classifier provides a soft binary decision with likelihood:

$$C = L(\mathbf{f}|G_1) - L(\mathbf{f}|G_0) - d, \tag{14}$$

where $L(\cdot)$ is the log likelihood, $\mathbf{f}$ is the feature vector of a window, G_1 is the GMM of *writing*, G_0 is the GMM of *non-writing*, and d is the threshold to adjust the operating point. A window is classified as *writing* if $C \geq 0$ and *non-writing* if $C < 0$.

We examine the feature distributions for *writing* and *non-writing* and model them with a single Gaussian mixture per model. Some feature distributions for *non-writing* are not quite close to a Gaussian distribution, e.g., a Gamma distribution is a better fit for f_2 and f_6. To better fit the feature distributions for *writing* and *non-writing* with GMM, we modify the features as follows,

$$\hat{f}_i = log(f_i + \epsilon), \text{where } i = 1, .., 6$$
$$\hat{f}_7 = log(f_7 + 1).$$

We experiment with different feature vectors $\mathbf{f}$ for GMMs and evaluate their receiver operating characteristic (ROC) curves. The ROC curves of individual features show that f_2 and f_3 are less effective to distinguish *writing* and *non-writing*, i.e., closer to the bottom right corner. Then, we experiment with four sets of feature vectors:

set 1) $f_{1,4-7}$, set 2) $\hat{f}_{1,4-7}$, set 3) f_{1-7}, set 4) $\hat{f}_{1-7}$. For GMMs, we also experiment with different types of covariance matrices of the feature vector: diagonal, full, and sparse covariance matrix. A diagonal covariance matrix means that we treat each feature in a feature vector independently. In contrast, a full covariance matrix means full dependency between features. In a sparse covariance matrix, we only correlates features that are intuitively dependent, i.e., the angle-based features (f_1 and f_2), and the distance-based features (f_4 and f_5), and set other cross covariances to zero. In Figure 27, we plot the ROC curves of the best two feature sets of each covariance matrix type. With a false alarm rate of 5%, $\hat{f}_{1-7}$ with a full covariance matrix has the highest true positive rate of 96.3% with $d = -1.4$, which is selected as the operating point of the classifier.

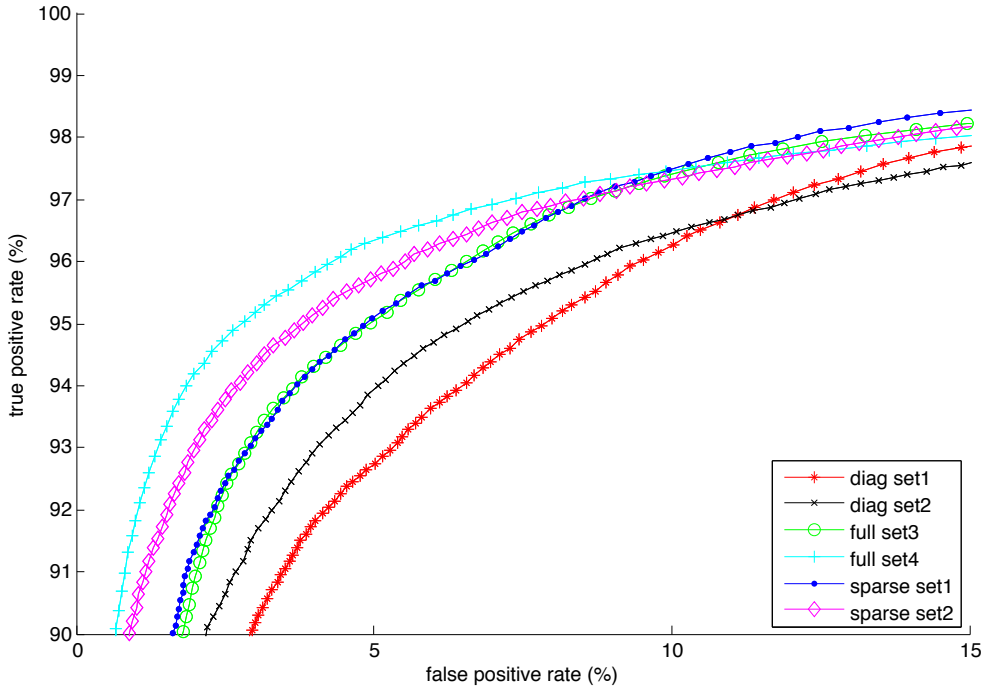


Figure 27: ROC curves of different feature vectors and covariance matrices, where the feature set 1 to 4 are $f_{1,4-7}$, $\hat{f}_{1,4-7}$, f_{1-7}, $\hat{f}_{1-7}$, respectively, and diag, full, and sparse indicate the type of covariance matrix for the GMM.

Basically, a window of a more complicated motion trajectory results in a higher score, e.g., $C = 10.98$ in Figure 26c. In contrast, a simpler motion results in a lower

score, e.g., $C = -4.29$ in Figure 26f. The windows in Figure 26a and 26f are correctly classified as *non-writing*. The windows in Figure 26c and 26d are also correctly classified as *writing*. The *mix* window tends to have a score close to 0. For example, Figure 26b is classified as *non-writing* with a score $C = -1.38$, and Figure 26e is classified as *writing* with a score $C = 0.35$. Therefore, ambiguity is expected for the windows that span around the boundary of a writing segment.

5.3.2.3 From Windows of Writing Events to Writing Segments

After window-based writing event classification, we need to convert windows of writing events to writing segments. With our setting on the window length and step size, every sub-window of 10 samples is covered by six windows. A sub-window is determined as *writing* if two or more of the six windows are *writing*, and we combine consecutive *writing* sub-windows into a writing segment. Figure 25c is an example of the 2D trajectory of the detected writing segment, which contains some distortion in the word boundary.

Different from typical conversion schemes, such as majority vote or sequential testing, our conversion scheme is more greedy in the detection of handwriting because the writing parts missed in the detection stage can never be recovered in the recognition stage. Hence, the recognizer needs to handle the imprecise segmentation or false alarms from the detector.

To evaluate the performance of air-fingerwriting detection, we categorize the detected writing segments into four types in the following order:

a) *discard*: the segment has a length less than or equal to 60 samples

b) *false alarm*: no overlap with the ground-truth segment

c) *imprecise*: less than 80% overlap with the ground-truth segment, or the offset of start/end point is greater than 50 samples

d) *precise*: greater than 80% overlap with the ground-truth segment, and the offset of start/end point is less than 50 samples

The *discard* segments are too short and will not be passed on to the recognizer. With the detector setting ($\hat{f}_{1-7}$ with full covariance matrix and $d = -1.4$), we have 2295 *precisie*, 478 *imprecise*, 68 *false alarm*, and 164 *discard* detected writing segments out of 2700 recordings. In terms of writing event detection, all the writing activities are detected except the word `II`, i.e., 1 of 2700. The limitation of the proposed detection method is that letter `I` by itself cannot be spotted due to its simple swiping down motion.

We observe that some subjects pause between letters when writing. If the pause is too long, it may result in separate detected writing segments (sub-word) of a word recording, e.g., `HARD` by subject `M3` is detected as three *imprecise* writing segments `H`, `AR`, and `D`. For these sub-word detections, we manually examine the writing segments and assign the correct labels of letters.

As introduced in Section 5.2.5, the letter-based word recognition can handle arbitrary letter sequences, i.e., words and sub-words make no difference. On the other hand, the word-based word recognition can only recognize words in the vocabulary. To handle the case of sub-word detection in one word recording, we have to expand the word network to include all possible sub-words in the vocabulary. This approach is impractical as the vocabulary size grows up. Instead, we merge detected segments that are no more than 60 samples apart and include the motion in-between as the ligature between sub-words. The merged detection results have 2225 *precise*, 483 *imprecise*, 30 *false alarm*, and 54 *discard* writing segments. After merging, there are still 18 *imprecise* segments containing partial words, which are excluded in the evaluation of word-based recognition. Note that the merge operation may connect nearby false detections and distorts the boundary of a writing segment, which explains the decrease of *precise* and *false alarm* segments.

5.3.3 Air-fingerwriting Recognition

For air-fingerwriting recognition, we modify the HMM-based recognizer in Section 5.2. According to the results in Table 12, we use the 2D position and velocity on the xy-plane as the feature (observation) vector for the HMMs. Here, we redefine the notation of motion data for the Leap. Let $P^o = [p_x(i), p_y(i)]^\top$ and $V^o = [v_x(i), v_y(i)]^\top$ denote the 2D position and velocity, respectively, where $i = 1, 2, ...N$, and N is the number of samples in a writing segment. The superscript o indicates the raw data from the detected segment (only 5-point moving average is applied at the detection stage).

The normalization process is required to make the recognizer scale and speed invariant. Normalization of P^o and V^o is accomplished as follows,

$$P = \frac{(P^o - \overline{P^o})}{\sigma_y}, \tag{15}$$

$$V = \frac{V^o}{\max ||V^o(i)||}, \tag{16}$$

where $\overline{P^o}$ is the mean of P^o, and σ_y is the standard deviation of p_y. The handwriting detection is not perfect and may introduce non-writing motions at the beginning and/or end of a detected writing segment. Therefore, the bounding box of P may be distorted and does not necessarily correspond to the virtual writing box. In Equation 15, we make P zero mean and unit variance in the y-axis because the "height" of letters is a more reliable measurement of the virtual writing box than the "width". The normalization of V is simply the 2D version of Equation 4.

The main difference between the push-to-write paradigm and automatic detection is the potential non-writing motions incurred in the detection stage. We should take the non-writing parts into account when designing the recognizer. For automatic speech recognition (ASR), the filler (or garbage) model is commonly used to absorb non-speech artifacts and handle out-of-vocabulary words for keyword spotting [8]. In our air-fingerwriting system, we use the filler model to handle the non-writing motion. The filler HMM is a single state model with self-transition and one Gaussian

mixture per state. For the character and ligature models, the chosen HMM topology is identical to what we define in Section 5.2.3.

To train the character and ligature models, we utilize the existing air-handwriting data for better initialization. We first apply Equation 15 and 16 to the 2D position and velocity from the 6-DOF air-handwriting data. Then, we train the refined models for characters and ligatures as described in Section 5.2.3 as the initial values of the character and ligature HMMs. The filler model is initialized with zero mean and global variance of all *precise* writing segments.

After we initialize the HMMs, we synthesize the HMM for each word in our vocabulary and append the filler model in the front and back. The leave-one-out cross validation on subjects results in 18 training sets. We perform embedded Baum-Welch re-estimation on all *precise* segments in each training set to obtain the final character and ligature HMMs. The training procedure is identical to that in Section 5.2.3. In addition, we perform forced alignment on the training data. When the end point of a segment is fairly close to the ground truth, the filler model is forced to pass through with occupancy of only one sample. When the detected segment is corrupted, the filler absorbs the non-writing motions as expected.

We use the trained HMMs of characters, ligatures, and the filler to build the word-based and letter-based decoding word network. We show the word-based decoding network in Figure 28 and a simplified example of the letter-based decoding network in Figure 29.

The word networks in Figure 28 and 29 are identical to the ones in Figure 19 and 20 except the addition of the filler model and the skip arc. The filler models in the front and back are intended to absorb the possible non-writing motions at the begin and end of a detected word segment. The skip arc between the fillers allows the decoding path of no writing at all, which is meant to reject false alarms of detected segments. In Figure 28, the word-based word recognition is formulated as

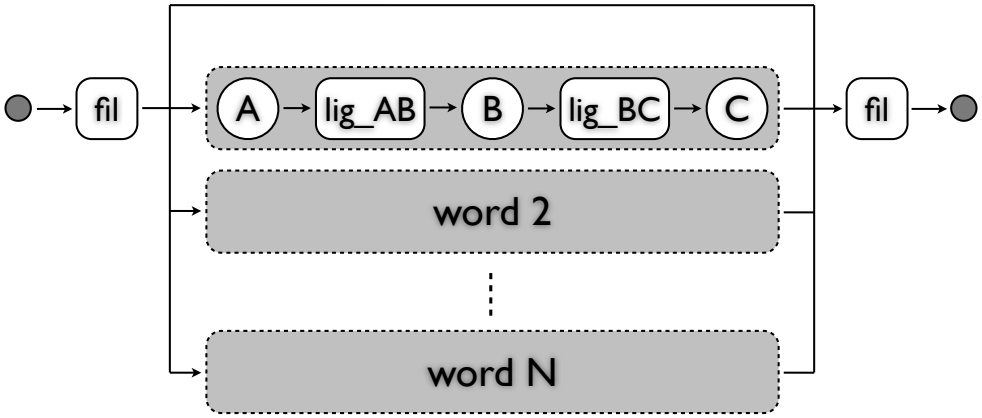


Figure 28: Decoding word network for word-based word recognition with fillers.

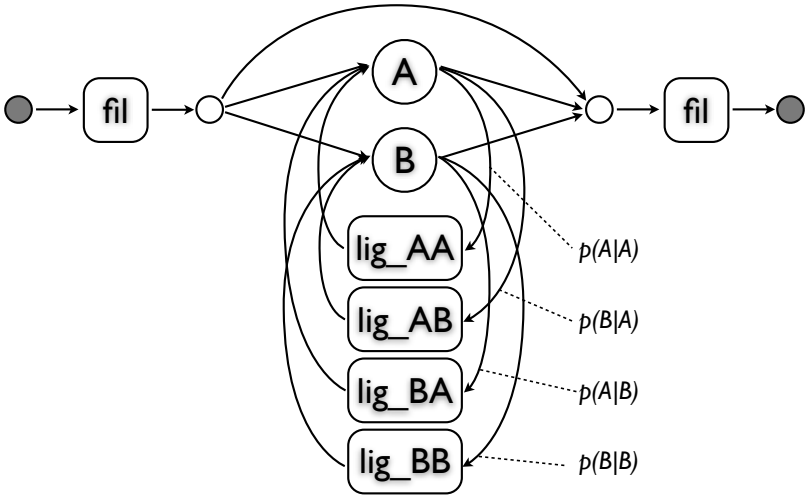


Figure 29: Decoding word network for letter-based word recognition with fillers.

a one-out-of-$N+1$ problem, i.e., N words in the vocabulary plus non-writing (skip). For the letter-based word recognition, we estimate the bigram language model from the vocabulary and embed the conditional probabilities in the transition arc from characters to ligatures (see Figure 29). Again, we use the Hidden Markov Model Toolkit (HTK) for fingerwriting modeling and recognition.

5.3.4 Experimental Results

It is important to understand how handwriting recognition is affected by the detection. We evaluate the recognition with all detected word segments except the *discard* ones. In our recording, each subject writes 100 common words and 50 unique ones. For

Table 15: The average segment error rate (SER) of word-based recognition on the merged detection results

	segment #	SER (%)
precise (common)	1481	0.34
precise (unique)	744	0.54
imprecise (common)	313	2.56
imprecise (unique)	152	5.26
false alarm	30	20.00
overall	-	1.15
ground truth	-	0.15

each testing set of leave-one-out cross validation, we separate the words into the common set (included in the training set) and the unique set (does not appear in the vocabulary of the training set). The unique testing set is the most challenging case, i.e., a new user walks in and writes unseen words in the training data.

We use the $1k$-vocabulary to form the word-based decoding network and estimate the bigram language model for the letter-based decoding network. The word-based recognition is evaluated with the merged detection results, and we show the average segment error rate (SER) in Table 15. The letter-based recognition is evaluated with the detected segments, and the average SER and character error rate (CER) are shown in Table 16. The SER and CER are calculated as follows,

$$\text{SER} = \frac{E}{N_s} \tag{17}$$

$$\text{CER} = \frac{S + I + D}{N_c}, \tag{18}$$

where E is the total segment errors, N_s is the total number of segments, S, I and D are the counts of substitution, insertion and deletion errors at the character level, and N_c is the total number of characters. When computing the overall SER with Equation 17, non-rejected false alarms are counted in E, but we do not include the number of *false alarm* segments in N_s.

For word-based recognition, the SER of the unique set is roughly two times larger than the SER of the common set, and 80% of the *false alarm* segments are rejected.

Table 16: The average segment error rate (SER) and character error rate (CER) of letter-based recognition on the detection results

	segment #	SER (%)	CER (%)
precise (common)	1530	5.16	1.52
precise (unique)	765	6.14	1.83
imprecise (common)	325	23.69	8.21
imprecise (unique)	153	24.84	8.43
false alarm	68	47.06	-
overall	-	9.84	-
ground truth	-	4.59	-

For letter-based recognition, the SER difference between the common and unique sets is relatively small, but the rejection rate drops to 52.94%. The scalability of vocabulary may be an issue for word-based recognition, but should be less of a concern for letter-based recognition. This observation conforms to our results of air-handwriting in Section 5.2.5.

The segmentation quality from the detection result is another factor that affects the recognition performance. For word-based recognition, the average SERs of *precise* and *imprecise* segments are 0.40% and 3.44%, respectively. The average *precise* and *imprecise* SERs are 5.49% and 24.90% for letter-based recognition. To better understand the effect of segmentation, we also list the SER of recognition of the ground-truth writing segments in Table 15 and 16. The SER of the *precise* segments (with less than 50 samples of end point offset and greater than 80% overlap) is close to the SER of the ground-truth segments, but the *imprecise* segments result in much higher error rates. The ground-truth segments are obtained by the push-to-write scheme, which leads to another interesting comparison with the WER of air-handwriting of the user-independent case in Table 12. For word-based recognition, the WER of $\hat{P}\hat{V}$ is 0.023%, which is much lower than the ground-truth SER 0.15% in Table 15 because of the small 40-word vocabulary. For letter-based recognition, the WER of $\hat{P}\hat{V}$ is 9.2%, which is roughly twice the SER of ground-truth segments, 4.59%, in Table 16.

The huge performance gap is resulted from the speculation that air-handwriting rendered with a handheld device involves larger variations than air-fingerwriting among different subjects.

We show that the quality of writing segmentation directly affects the recognition results, and the overall performance depends on both the detector and the recognizer. Imprecise segmentation can lead to insertion of non-writing motions or deletion of the writing part. It becomes problematic when the detected word boundary is off too much. For letter-based recognition, the non-writing part of a detected segment may be falsely recognized as a letter while the remaining part is correct. The non-writing part is less of a concern for word-based recognition due to the strong constraint on the vocabulary. For the deletion case, the letter-based recognizer may wrongly decode a partial letter as a filler (non-writing) or other letters. If the deleted part is not too much, it is possible for the word-based recognizer to recognize correctly based on other letters in the segment.

The number of segments for each detection case are also listed in Table 15 and 16. The *precise* segments are both around 82.7% of the total segments with and without merging. The overall SER of word-based recognition is 1.15%, and the overall SER of letter-based recognition is 9.84%. If user-assisted correction is allowed, providing the n-best recognition results for the user to select the right one is a good strategy to further reduce the error rates. For example, the overall SER of letter-based recognition is reduced to 5.19% and 2.52% for 2-best and 5-best recognition.

5.4 Usability Study

The usability of air-handwriting is an interesting topic by itself and will be covered in this section. We focus on the usability study of using air-handwriting for text input in our universal motion-based control with a handheld device. Pointing and clicking on a virtual keyboard displayed on screen is another possible method for text

input, so we consider the virtual keyboard as the comparison group. Air-fingerwriting involves a completely different tracking system, and hence we do not consider the usability study of air-fingerwriting here. However, we believe that the usability of air-fingerwriting should be similar or even better in terms of input efficiency, effortlessness and intuitiveness.

During the recording of air-handwriting, the subjects often started to feel arm fatigue after 10 to 15 minutes of writing. In general, longer words are harder to write in one stroke without making mistakes. The consumed time and efforts makes air-handwriting more appealing to input shorter text or commands. Although sentence-level handwriting recognition is feasible [5], it is unlikely that the user will choose air-handwriting to input a whole sentence.

5.4.1 Apparatus and Procedure

The apparatus for usability study is the same as our universal motion-based control framework. The user sits in a living-room like environment with a 65" full HD display and a handheld tracking device of our hybrid tracking system. The remote functions as a mouse, whose translation in the vertical plane is mapped to the cursor movement on the display with a scale of one meter to 2000 pixels. Button A works as the mouse left click, and Button B is used for push-to-write for air-handwriting. We create a logger program that displays the word to input, a status box, and a `start` button. The subject needs to click `start` to start logging the time and traverse distance of writing/typing. The logging stops automatically once the correct word is recognized or typed.

For air-handwriting, we use the word-based word recognition with the 1040-word vocabulary as described in Section 5.2.5. The time and traverse distance while holding Button B (writing) is recorded separately. The status box displays the recognized word or input status, e.g., writing or recognizing. If an error occurs, the subject

re-writes until the word is correctly recognized, and we accumulate the writing time of each trial. For the virtual keyboard, we use the built-in on-screen keyboard of Windows 7, which is resized to the lower half of the screen. The window of our logger is on the upper half of the screen, and the status box now shows the input letters.

Among the 20 subjects for user study, eight of them participated in the air-handwriting recording, and the rest are novice users. We let every subject get used to the system first and start the experiment once he or she feels comfortable with both input methods. Each subject is asked to "copy" 50 words, which consist of the whole 40-word vocabulary and 10 words randomly drawn from the $1k$-word vocabulary. These 50 words are shuffled and remain in the same order for both air-handwriting and virtual keyboard. We also randomize the order of the sessions of two input methods for all subjects. The program logs the completion time and motion footprint, i.e., total traverse distance, of each word input by air-handwriting and virtual keyboard. After the experiment, we also ask the subject to rate the intuitiveness, arm fatigue level, and their preference of these two methods.

5.4.2 Results and Discussion

We show the average writing/typing time and total traverse distance for words of different length in Table 17. Because air-handwriting is recognized on a word basis, we report the average number of attempts to correctly input a word. We can see that longer words tend to have higher recognition accuracy and hence need fewer attempts. A longer word is harder to write correctly in one stroke, e.g., the specific stroke order may not be followed for certain letters. The word-based recognition is robust to individual character errors because the correctly rendered part may be able to compensate the drop of the likelihood score caused by the character errors. Hence, a longer word is more likely to be recognized correctly because the portion of the correct writing may be larger. Regardless to the recognition results, the average

Table 17: Usability study of air-handwriting and virtual keyboard (objective metrics)

	Air-handwriting			Virtual keyboard		
word length	time (sec)	distance (cm)	attempt # per word	time (sec)	distance (cm)	extra key # per word
2	5.4	161	1.38	2.6	49	0.04
3	7.2	249	1.19	4.3	86	0.09
4	8.3	312	1.07	5.7	120	0.14
5	10.1	396	1.06	7.4	152	0.29
6+	14.0	566	1.04	9.2	174	0.29

writing time of a 2-letter word is 3.9 (= 5.4/1.38) second. For virtual keyboard, we report the average number of extra keystrokes, e.g., a typo and a backspace count as two extra keystrokes.

We use words-per-minute (WPM) as the performance metric for text input efficiency. The WPM of air-handwriting and virtual keyboard are 5.43 and 8.42, respectively. Compared to conventional text input methods, the WPM of pen-based handwriting without recognition is in the range of 15 to 25, and the WPM range of QWERT typing (hunt-and-peck style) is 20 to 40. Motion-based text input methods are roughly three to five times slower than the conventional ones because relatively large and unconstrained control motions are involved. Our study gives an idea of the speed for these alternative text input methods on a motion-based user interface.

The objective metrics show that air-handwriting is roughly 1.5 times slower and 3 times longer in motion footprint than the virtual keyboard. However, we get quite interesting results from the subjective evaluation as shown in Table 18.

Air-handwriting is a variation of conventional writing, and virtual keyboard follows the same metaphor of typing on a touchscreen. Both methods are intuitive to users, and virtual keyboard is considered more intuitive than air-handwriting. In terms of arm fatigue, both methods have neutral scores. Motions in the air involve more muscles than keyboard or touch-based interaction and thus cause more fatigue. Even though the motion footprint of air-handwriting is three times larger, it doesn't

Table 18: Usability study of air-handwriting and virtual keyboard (subjective rating from 1 to 5)

Question	air-handwriting	virtual keyboard
1. Intuitiveness [5: most intuitive]	4.10	4.75
2. Arm fatigue level [5: no fatigue]	3.05	3.10
3. Vote for inputing a short word (2-3 letters)	16	4
4. Vote for inputing a long word (4+ letters)	11	9
5. Satisfaction of recognition performance [5: most satisfied]	4.25	-

directly reflects in arm fatigue. The arm fatigue level actually relates to the writing or typing style. For example, air-handwriting should cause less fatigue for a user who rests the elbow and writes with the upper arm and wrist than a user who holds the whole arm in the air. The layout of virtual keyboard is fixed for all subjects. To cover all keys, it requires a larger range of movement, e.g., the distance between key `Z` and `Backspace` is about 60 cm (1200 pixels). Six subjects mention that the keyboard layout is too big. Reducing the size of the keyboard layout reduces the motion footprint. However, smaller keys can be prone to "typing" errors and require more precise pointing motions. The majority of users choose air-handwriting for short text input (2-3 letters), and about half of users prefer air-handwriting for long text input (4+ letters).

Based on our study, air-handwriting may not be fast enough for general-purpose text input, but it is suitable for infrequent and short text input on a motion-based user interface, where conventional writing or typing is not available. Although virtual keyboard is faster than air-handwriting, the backdrop of virtual keyboard is that it requires a display and precise pointing. Typing on a virtual keyboard requires two focuses of attention (FOA), i.e., the user needs to pay attention to the keyboard and then the input result. On the contrary, air-handwriting is a single-FOA task. The user doesn't necessarily need the visual feedback of writing and achieves "eyes-free" text input. Air-handwriting recognition doesn't require precise pointing and is applicable

to a broader range of motion tracking systems.

There are other usability issues of air-handwriting from user feedback. The box-writing style is easy to pick up, but it needs some practice to write with the specified stroke order. In our current system, writing with different stroke orders may cause errors in recognition, especially for shorter words. Five users suggest to write without constraints on the stroke order, and four users would like to write without holding a button. Although the usability study is specific for air-handwriting, we believe the usability of air-fingerwriting should be similar or even better in terms of effortlessness and intuitiveness.

CHAPTER VI

CONCLUSIONS

There are all sorts of interactions with different digital services surrounding us nowadays, particularly in the living room or office environment. A task or service specific device is usually required to control individual service. For example, we need a mouse or trackpad for point-and-click operation, a keyboard for text input, a remote for TV control, etc. Therefore, an unfulfilled demand arises for an effective user interface that can support all general interface functionalities in a single design.

In this dissertation, we answer the aforementioned demand by proposing a universal motion-based control framework. Motion-based interactions can naturally map the user's real-world experience to the control of user interfaces, which fulfill the design goal of the natural user interface (NUI). The motion tracking system is essentially the input device of a motion-based user interface. The proposed framework incorporates 3D motion tracking with minimum, untethered user-worn components and achieves one-to-one motion mapping, which allows the user to control position and orientation simultaneously. To precisely track the control motion, we build a hybrid tracking system of optical tracking and inertial sensing technologies. The tracking results actually contain both explicit 6-DOF (position and orientation in the global coordinates) and implicit 6-DOF (acceleration and angular speed in the device coordinates).

In Chapter 3, we categorize the supported functionalities in the universal motion-based control framework into three groups:

1. *2D user interface:* pointing, selection, dragging, and scrolling.

2. *3D user interface:* manipulation (translation and rotation), selection, navigation, and browsing.

3. *motion recognition:* motion gesture and air-handwriting.

With 6-DOF motion data, we carefully design interaction techniques to map the control motions to control functionalities on the user interface. We make the mapping logical and closely related to the user's real-world experience, which helps the user form a consistent mental model to increase the usability. We summarize the design of universal motion-based control and the integration between *2D user interface*, *3D user interface*, and *motion recognition* in Table 1.

Beyond the aspect of interface design, motion recognition involves problems of machine learning. We address motion recognition in two levels of sophistication: motion gesture recognition (Chapter 4) and air-handwriting recognition (Chapter 5).

A motion gesture is rendered by a handheld device in free space without regard to the posture, finger or body movements. We utilize the 6-DOF motion tracking system to track and record motion gestures. Two approaches are proposed for motion gesture recognition: the statistical feature-based linear classifier as a simple baseline and the HMM-based recognizer that takes account of the spatio-temporal nature of gesture signals. For the HMM-based recognizer, we propose a normalization procedure that effectively alleviates the large in-class variations of motions caused by different gesturing styles between users. Although motion gestures are usually defined by the spatial trajectory, we prove that signals in other dimensions, e.g., velocity, orientation, acceleration, and angular speed, still contain information to distinguish the gestures. Combining signals of different kinematic meanings can further improve the recognition performance. We compare the effectiveness of various features derived from different tracking signals in both user-dependent and user-independent cases. In the user-dependent case, both approaches work pretty well. In the user-independent case, the statistical feature-based linear classifier achieves 85.2% and 93.5% accuracy with implicit and explicit 6D data. The HMM-based recognizer has higher recognition rates, 91.9% and 96.9% respectively. Another advantage of the HMM-based

approach is the flexibility in choosing the tracking technologies while maintaining the accuracy above 96%.

Air-handwriting is tracked and recorded in the same manner as motion gestures. Therefore, air-handwriting is uni-stroke without pen-up/pen-down information and quite different from the ordinary pen-based writing. First, we adopt the push-to-write paradigm for explicit segmentation of the word boundary and address air-handwriting in two levels: motion characters and motion words. Motion characters are handled and recognized similar to motion gestures, i.e., each character is modeled with a HMM. A motion word is modeled by concatenating character and ligature models. We present the details of clustering ligatures and training the character and ligature HMMs. Two approaches of ligature modeling are proposed: hard clustering and decision tree. The former is proven to be sufficient for word-based word recognition. The latter provides better capability of ligature modeling, which improves the performance of letter-based word recognition. The word-based word recognition achieves relatively low WER but is not able to recognize out-of-vocabulary words. The word-based recognizer is suitable for applications that have a limited vocabulary and stringent requirement on the accuracy. On the other hand, letter-based word recognition has around 10% WER but can handle arbitrary letter sequences and progressive decoding. If the vocabulary is relatively small, a restrictive bigram language model can substantially reduce the WER of letter-based word recognition. To further improve the letter-based recognition accuracy, the system can provide suggestions with n-best decoding and lets the user choose the right one.

As an extension of air-handwriting, we investigate the detection and recognition of air-fingerwriting. The writing motion is rendered by a fingertip and tracked by a glove-free and marker-free hand tracking device, the Leap. We propose an approach that automatically detects writing events in the tracking signals and segments the writing part. The proposed window-based GMM detector classifies whether a writing

event occurs in the sliding window, and we train the GMM detector to work at an operating point of 5% false alarm rate and 95.7% true positive rate of writing windows. A writing segment is formed from processing and merging the consecutive writing windows and then passed to the recognizer for the final result of recognition or rejection. Regardless of the segmentation precision, the detector commits a very low false negative rate. We evaluate the recognition performance of the detected segments and compare with the recognition of the ground-truth segments. The writing segmentation quality from the detection stage has a great influence on the recognition performance. The overall SER of word-based and letter-based recognition is 1.15% and 9.84%, respectively.

To complete the evaluation of air-handwriting, we conduct usability study on the input speed, motion footprint, physical strain, and other subjective evaluation of two motion-based text input methods: air-handwriting and virtual keyboard. The results suggest that air-handwriting is suitable for short and infrequent text input on a motion-based user interface.

In sum, our design of the universal motion-based control is capable of supporting the general functionalities in both 2D and 3D user interfaces. Moreover, our design is supplemented by motion recognition. we achieve robust motion recognition of motion gestures and air-handwriting. Motion gestures provide another natural and intuitive way to interaction with the motion-based control framework. Air-handwriting serves as the primary text input method and completes our design. We hope that the work in this dissertation contributes to both the fields of human-computer interaction and machine learning.

6.1 Summaries and Contributions

The contributions of this dissertation can be summarized as follows:

1. Proposed the universal motion-based control framework that supports interactions with the 2D user interface, 3D user interface, and motion recognition in a single design.
    - Built a 6-DOF motion tracking system that provides comprehensive motion data of position, orientation, acceleration, and angular speed.
    - Exploited the design space of motion-based control to integrate 2D interactions, 3D interactions, motion gestures, and air-handwriting seamlessly.
2. Achieved robust motion gesture recognition.
    - Proposed a normalization procedure that effectively deals with the motion variations between users.
    - Evaluated the recognition performance with different dimensions of tracking signals, which is valuable for the system designer to choose the proper tracking technology.
3. Achieved robust air-handwriting recognition.
    - Formulated air-handwriting modeling with the elements of motion characters, ligatures, and the filler model.
    - Proposed letter-based and word-based word recognition to address different needs of system design.
    - Developed a systematic way to detect and recognize air-fingerwriting and evaluated the overall system performance.

Refer to the references chapter under the authors M. Chen, G. AlRegib, and B.-H, Juang for three journals (two under preparation) and five conference publications that have come out so far from this work.

6.2 Future Research Suggestions

More work can be done to extend and refine the developed universal motion-based control framework. We suggest other potential research directions as follows.

- Integrate our universal motion-based control with digital services in a living-room environment. The supported interactions can be tailored to better suit the targeting applications. Usability study is strongly recommended to ensure optimal user experience.

- Exploit other vision-based technologies for 6-DOF motion tracking and adapt the interaction techniques in the developed universal motion-based control to controller-free.

- Use the minimum classification error (MCE) principle to train the classifier to improve the motion recognition performance upon current maximum likelihood (ML) approach.

- Extend the air-handwriting recognition to handle lowercase letters and remove the contraints on allography and stroke orders. A substantial amount of data recording is required.

- Further improve the air-handwriting performance by exploiting other features, such as angle-based features, applying trigram language models, and rescoring for n-best results.

- Build a real-time demo system of Figure 22 for usability study of air-fingerwriting.

- Integrate motion gestures with air-handwriting to further expand the control capabilities, e.g., use gestures to undo or correct the text input of air-handwriting.

APPENDIX A

VOCABULARY FOR AIR-HANDWRITING

The 40-word vocabulary contains the names of common television channels and common digital/Internet services. The complete list is as follows.

Table A.1: The 40-word vocabulary

ABC	BBC	WEATHER	GAME
CBS	FX	NEWS	VOICE
CNN	HULU	MLB	CALL
DISCOVERY	TNT	NFL	MAIL
DISNEY	MUSIC	TRAVEL	MSG
ESPN	JAZZ	POKER	FB
FOX	ROCK	FOOD	YOU
HBO	DRAMA	KID	GOOGLE
NBC	MOVIE	MAP	SKYPE
TBS	SPORT	TV	QUIZ

The $1k$-word vocabulary for air-handwriting includes the most frequent 1000 two-letter and three-letter words and three-letter prefixes from the Google Web 1T 5-gram data set without overlap with the 40-word vocabulary. The new $1k$-word vocabulary for air-fingerwriting includes the most frequent 1000 two-letter, three-letter, and four-letter words and four-letter prefixes from the Google Web 1T data set. Space precludes us from showing the whole list for these two $1k$-word vocabularies. Instead, we list the 100 words of the common set of the new $1k$-word vocabulary for example. For the complete vocabularies, please refer to `http://www.ece.gatech.edu/6DMG`.

Table A.2: The 100 common words of the new $1k$-vocabulary for air-fingerwriting

SET	DAYS	ISSU	MAP	LONG
LIFE	MONT	GIVE	DIFF	SEND
COUL	PLAC	SECU	COND	FAMI
CHAR	AGAI	TRAV	ADDR	EBAY
OPEN	FOUN	CHEC	WEBS	SECT
STAN	BEFO	DID	OFF	NOTE
MUST	VISI	THOS	USIN	BUIL
SOUT	FEAT	COST	RELE	CODE
LEVE	POIN	HARD	BOAR	HOUR
DVD	HIST	DESC	UPDA	VERS
JOIN	VALU	TRAD	LARG	SOCI
REPL	TOOL	BETW	ADVA	DIST
TOPI	WOME	ROOM	ARCH	PERF
MEET	BLAC	TITL	LIVE	OWN
BEIN	MUCH	FEED	BOTH	WEST
SMAL	ASSO	WHIL	ENGL	SIZE
SOUR	NEXT	SEX	EXAM	JAZZ
ZIP	FAQ	REQU	QUIT	YORK
POKE	KNOW	OBJ	GPS	PSY
PROJ	KEY	SQUA	XBOX	ROCK

REFERENCES

[1] "Gestural user interfaces — 3gears systems." http://www.threegear.com/, 2013.

[2] "Leap motion." http://www.leapmotion.com/, 2013.

[3] Albinsson, P.-A. and Zhai, S., "High precision touch screen interaction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, (New York, NY, USA), pp. 105–112, ACM, 2003.

[4] Amma, C., Gehrig, D., and Schultz, T., "Airwriting recognition using wearable motion sensors," in *Proc. of the 1st Augmented Human Intl. Conf.*, AH '10, pp. 10:1–10:8, 2010.

[5] Amma, C., Georgi, M., and Schultz, T., "Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3d-space handwriting with inertial sensors.," in *ISWC*, pp. 52–59, IEEE, 2012.

[6] Ashbrook, D. and Starner, T., "Magic: a motion gesture design tool," in *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, CHI '10, (New York, NY, USA), pp. 2159–2168, ACM, 2010.

[7] Baum, L. E., Petrie, T., Soules, G., and Weiss, N., "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. pp. 164–171, 1970.

[8] Bazzi, I., *Modelling out-of-vocabulary words for robust speech recognition*. PhD thesis, Massachusetts Institute of Technology, 2002.

[9] Blickenstorfer, C. H., "Graffiti: Wow!," *Pen Computing Magazine*, pp. 30–31, Jan. 1995.

[10] Bowman, D. A., Chen, J., Wingrave, C. A., Lucas, J. F., Ray, A., Polys, N. F., Li, Q., Haciahmetoglu, Y., Kim, J.-S., Kim, S., Boehringer, R., and Ni, T., "New directions in 3d user interfaces," *IJVR*, vol. 5, no. 2, pp. 3–14, 2006.

[11] Bowman, D. A., Kruijff, E., LaViola, J. J., and Poupyrev, I., *3D User Interfaces: Theory and Practice*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004.

[12] Cao, X. and Balakrishnan, R., "Visionwand: interaction techniques for large displays using a passive wand tracked in 3d," in *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, (New York, NY, USA), pp. 173–182, ACM, 2003.

[13] Chen, M., AlRegib, G., and Juang, B.-H., "Characteristics of spatio-temporal signals acquired by optical motion tracking," in *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pp. 1205 –1208, Oct. 2010.

[14] Chen, M., AlRegib, G., and Juang, B.-H., "6D motion gesture recognition using spatio-temporal features," in *Proc. of IEEE intl. conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2341 –2344, Mar 2012.

[15] Chen, M., AlRegib, G., and Juang, B.-H., "Feature processing and modeling for 6d motion gesture recognition," *Multimedia, IEEE Transactions on*, vol. 15, no. 3, pp. 561–571, 2013.

[16] Chen, M., AlRegib, G., and Juang, B.-H., "Air-fingerwriting detection and recognition with the leap," submitted for publication.

[17] Chen, M., AlRegib, G., and Juang, B.-H., "Air-handwriting recognition with 6-dof motion tracking," submitted for publication.

[18] Chen, M., AlRegib, G., and Juang, B.-H., "An integrated framework for universal motion control," in *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '11, (New York, NY, USA), pp. 513–518, ACM, 2011.

[19] Chen, M., AlRegib, G., and Juang, B.-H., "6DMG: A new 6D motion gesture database," in *Proc. of the third annual ACM conf. on Multimedia systems*, MMSys '12, 2012.

[20] Chen, M., AlRegib, G., and Juang, B.-H., "A new 6d motion gesture database and the benchmark results of feature-based statistical recognition," in *Proceedings of the First IEEE Conference on Emerging Signal Processing Applications*, ESPA '12, 2012.

[21] Dix, A., Finlay, J. E., Abowd, G. D., and Beale, R., *Human-Computer Interaction (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2003.

[22] Fu, A. W.-c., Keogh, E., Lau, L. Y. H., and Ratanamahatana, C. A., "Scaling and time warping in time series querying," in *Proceedings of the 31st International Conference on Very large data bases*, VLDB '05, pp. 649–660, VLDB Endowment, 2005.

[23] Godwin, A., Agnew, M., and Stevenson, J., "Accuracy of inertial motion sensors in static, quasistatic, and complex dynamic motion," *Journal of Biomechanical Engineering*, vol. 131, no. 11, p. 114501, 2009.

[24] Golubitsky, O. and Watt, S. M., "Online stroke modeling for handwriting recognition," in *Proceedings of the 2008 conference of the center for advanced*

studies on collaborative research: meeting of minds, CASCON '08, (New York, NY, USA), pp. 6:72–6:80, ACM, 2008.

[25] Golubitsky, O. and Watt, S. M., "Distance-based classification of handwritten symbols," *Int. J. Doc. Anal. Recognit.*, vol. 13, pp. 133–146, June 2010.

[26] Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., and Janet, S., "Unipen project of on-line data exchange and recognizer benchmarks," in *Pattern Recognition, 1994. Vol. 2 - Conference B: Computer Vision amp; Image Processing., Proceedings of the 12th IAPR International. Conference on*, vol. 2, pp. 29–33 vol.2, 1994.

[27] Hoffman, M., Varcholik, P., and LaViola, J., "Breaking the status quo: Improving 3d gesture recognition with spatially convenient input devices," in *Virtual Reality Conference (VR10)*, pp. 59 –66, Mar. 2010.

[28] Hu, J., Brown, M., and Turin, W., "Hmm based online handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 10, pp. 1039–1045, 1996.

[29] Isokoski, P. and Raisamo, R., "Quikwriting as a multi-device text entry method," in *Proceedings of the third Nordic conference on Human-computer interaction*, NordiCHI '04, (New York, NY, USA), pp. 105–108, ACM, 2004.

[30] Jaeger, S., Manke, S., Reichert, J., and Waibel, A., "Online handwriting recognition: the npen++ recognizer," *International Journal on Document Analysis and Recognition*, vol. 3, no. 3, pp. 169–180, 2001.

[31] JIN, L., Yang, D., Zhen, L.-X., and Huang, J.-C., "A novel vision based finger-writing character recognition system," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1, pp. 1104–1107, 2006.

[32] Jota, R., Nacenta, M. A., Jorge, J. A., Carpendale, S., and Greenberg, S., "A comparison of ray pointing techniques for very large displays," in *Proceedings of Graphics Interface 2010*, GI '10, (Toronto, Ont., Canada, Canada), pp. 269–276, Canadian Information Processing Society, 2010.

[33] Juang, B.-H. and Rabiner, L., "The segmental k-means algorithm for estimating parameters of hidden markov models," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 9, pp. 1639–1641, 1990.

[34] Karat, C.-M., Halverson, C., Horn, D., and Karat, J., "Patterns of entry and correction in large vocabulary continuous speech recognition systems," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '99, (New York, NY, USA), pp. 568–575, ACM, 1999.

[35] Katz, S., "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, no. 3, pp. 400–401, 1987.

[36] Kohavi, R., "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, IJCAI'95, (San Francisco, CA, USA), pp. 1137–1143, Morgan Kaufmann Publishers Inc., 1995.

[37] Költringer, T., Isokoski, P., and Grechenig, T., "Twostick: writing with a game controller," in *Proceedings of Graphics Interface 2007*, GI '07, (New York, NY, USA), pp. 103–110, ACM, 2007.

[38] Kratz, S. and Rohs, M., "Protractor3d: a closed-form solution to rotation-invariant 3d gestures," in *Proceedings of the 16th International Conference on Intelligent User Interfaces*, IUI '11, pp. 371–374, 2011.

[39] Kristensson, P. O. and Denby, L. C., "Text entry performance of state of the art unconstrained handwriting recognition: a longitudinal user study," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 567–570, ACM, 2009.

[40] Kristensson, P. O., Nicholson, T., and Quigley, A., "Continuous recognition of one-handed and two-handed gestures using 3d full-body motion tracking sensors," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, IUI '12, (New York, NY, USA), pp. 89–92, ACM, 2012.

[41] Lee, H.-K. and Kim, J. H., "An hmm-based threshold model approach for gesture recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, pp. 961–973, Oct. 1999.

[42] Liang, J. and Green, M., "Jdcad: A highly interactive 3d modeling system," *Computers and Graphics*, vol. 18, no. 4, pp. 499 – 506, 1994.

[43] Liu, J., Zhong, L., Wickramasuriya, J., and Vasudevan, V., "uwave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657 – 675, 2009. PerCom 2009.

[44] MacKenzie, I. S. and Soukoreff, R. W., "Text entry for mobile computing: Models and methods, theory and practice," *HUMAN-COMPUTER INTERACTION*, vol. 17, pp. 147–198, 2002.

[45] Madgwick, S., Harrison, A. J. L., and Vaidyanathan, R., "Estimation of imu and marg orientation using a gradient descent algorithm," in *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pp. 1–7, 2011.

[46] Makhoul, J., Starner, T., Schwartz, R., and Chou, G., "On-line cursive handwriting recognition using hidden markov models and statistical grammars," in *Proceedings of the workshop on Human Language Technology*, HLT '94, (Stroudsburg, PA, USA), pp. 432–436, Association for Computational Linguistics, 1994.

[47] Mäntyjärvi, J., Kela, J., Korpipää, P., and Kallio, S., "Enabling fast and effortless customisation in accelerometer based gesture interaction," in *Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*, MUM '04, pp. 25–31, 2004.

[48] Mine, M., "Virtual environment interaction techniques," tech. rep., UNC Chapel Hill CS Dept, 1995.

[49] Mitra, S. and Acharya, T., "Gesture recognition: A survey," *IEEE Transactions on Systems, Man and Cybernetics - Part C*, vol. 37, no. 3, pp. 311–324, 2007.

[50] Oh, J.-Y. and Stuerzlinger, W., "Laser Pointers as Collaborative Pointing Devices," in *Proc. Graphics Interface*, pp. 141–150, May 2002.

[51] Perlin, K., "Quikwriting: continuous stylus-based text entry," in *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, (New York, NY, USA), pp. 215–216, ACM, 1998.

[52] Pitsikalis, V., Theodorakis, S., Vogler, C., and Maragos, P., "Advances in phonetics-based sub-unit modeling for transcription alignment and sign language recognition," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pp. 1 –6, june 2011.

[53] Plamondon, R. and Srihari, S., "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 63 –84, Jan. 2000.

[54] Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T., "The go-go interaction technique: non-linear mapping for direct manipulation in vr," in *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, UIST '96, (New York, NY, USA), pp. 79–80, ACM, 1996.

[55] Rabiner, L. and Juang, B.-H., *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[56] Rubine, D., "Specifying gestures by example," *SIGGRAPH Comput. Graph.*, vol. 25, pp. 329–337, Jul. 1991.

[57] Ruiz, J., Li, Y., and Lank, E., "User-defined motion gestures for mobile interaction," in *Proceedings of the 29th International Conference on Human Factors in Computing Systems*, CHI '11, ACM, 2011.

[58] Schick, A., Morlock, D., Amma, C., Schultz, T., and Stiefelhagen, R., "Vision-based handwriting recognition for unrestricted text input in mid-air," in *Proc. of the 14th ACM intl. conf. on Multimodal interaction*, ICMI '12, (New York, NY, USA), pp. 217–220, ACM, 2012.

[59] Sin, B.-K. and Kim, J. H., "Ligature modeling for online cursive script recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 623–633, Jun. 1997.

[60] Starner, T., Weaver, J., and Pentland, A., "Real-time american sign language recognition using desk and wearable computer based video," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 1371 – 1375, dec 1998.

[61] Teather, R., Pavlovych, A., Stuerzlinger, W., and MacKenzie, I., "Effects of tracking technology, latency, and spatial jitter on object movement," *Proceedings of IEEE Symposium on 3D User Interfaces*, vol. 9, pp. 43–50, 2009.

[62] Vanacken, L., Grossman, T., and Coninx, K., "Multimodal selection techniques for dense and occluded 3d virtual environments," *International Journal of Human-Computer Studies*, vol. 67, no. 3, pp. 237 – 255, 2009. Current trends in 3D user interface research.

[63] Wang, R., Paris, S., and Popović, J., "6d hands: markerless hand-tracking for computer aided design," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, (New York, NY, USA), pp. 549–558, ACM, 2011.

[64] Wang, R. Y. and Popović, J., "Real-time hand-tracking with a color glove," *ACM Transactions on Graphics*, vol. 28, no. 3, 2009.

[65] Welch, G. and Foxlin, E., "Motion tracking: no silver bullet, but a respectable arsenal," *Computer Graphics and Applications, IEEE*, vol. 22, pp. 24 – 38, Nov 2002.

[66] Wobbrock, J. O., Morris, M. R., and Wilson, A. D., "User-defined gestures for surface computing," in *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 1083–1092, ACM, 2009.

[67] Wobbrock, J. O., Wilson, A. D., and Li, Y., "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proc. of UIST '07*, pp. 159–168, 2007.

VITA

Mingyu Chen was born in Taipei, Taiwan, in June 1983. He received the B.S. degree in electrical engineering from the National Taiwan University in Taipei, Taiwan, and the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA in 2009. He will receive the Ph.D. degree in electrical and computer engineering from Georgia Institute of Technology in 2013. Since July 2009, he has been with the Center for Signal and Information Processing (CSIP), Georgia Institute of Technology, as a Graduate Research Assistant. His research interests include motion tracking, motion recognition, and motion-based human-computer interaction.