Diss. ETH No. 17614

A Design Framework for 3D Spatial Gesture Interfaces

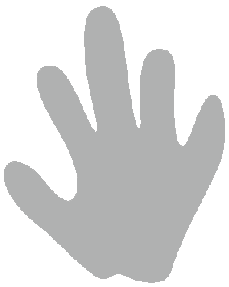
A dissertation submitted to
ETH Zurich

for the degree of
Doctor of Sciences

presented by
Doo Young Kwon
B.A. Ajou University, Republic of Korea
M.S. University of Washington, Seattle, USA
born September 4, 1973
citizen of Kang Neung, Kang Won, South Korea

accepted on the recommendation of
Prof. Markus Gross, ETH Zürich, Switzerland, examiner
Prof. Jürg Gutknecht, ETH Zürich, Switzerland, co-examiner

2008



"Men judge generally more by the eye than by the hand,
for everyone can see and few can feel.
Every one sees what you appear to be,
few really know what you are."

Niccolo Machiavelli (1469 - 1527), The Prince

Abstract

Gestures have been employed for human computer interaction to build more natural interface in new computational environments. As humans, we have the opportunity to create and learn gestures, and improve our interactions. Among the various types of gestures, 3D spatial gestures possess the most distinctive features of an individual's gestural identity and the unique qualities that result from a combination of innate physical factors and expressive characteristics.

There has been a great deal of research on robust gesture recognition, and many gesture-input systems have been proposed with new input devices and application scenarios. However, while they provide sophisticated methods for processing human gestures, their functionality has mostly been ad-hoc and not presented within a generative design framework. Therefore, it remains challenging to design a new gesture interface that leverages the growth of 3D gesture vocabulary and fully utilizes the advantages of the gestures.

This dissertation proposes a design framework for a 3D spatial gesture interface. The framework supports both gesture acquisition and the modeling of the physical and expressive characteristics that are unique to an individual gesture. Acquisition is accomplished using a multiple sensory approach that combines visual and body sensors. Novel gesture input devices are designed to facilitate the use of both sensor types and support various gesture-based inputs. The gesture modeling supports the registration and evaluation of 3D spatial gestures as well as their recognition. The framework is evaluated in terms of gesture recognition and learning task, and its use is demonstrated with the development of unique 3D spatial gesture interfaces.

Kurzfassung

Gebärden erlauben eine natürlichere Form der Interaktion zwischen Mensch und Maschine. Als Mensch haben wir die Fähigkeit Gebärden zu kreieren, zu lernen und dadurch die Interaktion zu verbessern. Unter den verschiedenen Typen von Gebärden besitzen dreidimensionale rümliche Gebärden die markantesten Charakterzüge und einizigartige Eigenschaften der individuellen gestischen Identität, die auf einer Kombination von immanenten physikalischen Faktoren und Ausdrucksfähigkeit beruhen.

Es existiert bereits eine grosse Anzahl von Forschungsarbeiten über die robuste Erkennung von Gebärden, und viele Gebärdeneingabesysteme mit neuen Eingabegeräten und Anwendungszenarios wurden vorgestellt. Obwohl diese technisch ausgefeilte Methoden zur Verarbeitung von Gebärden anbieten, sind deren Funktionalität meistens ad hoc und nicht innerhalb einer generativen Entwicklungsumgebung. Deshalb besteht die Herausforderung, eine Schnittstelle zu entwickeln, die das Wachstum des dreidimensionalen Gebärdenvokabulars und deren Vorteile vollständig nutzt.

Diese Dissertation präsentiert ein Designframework für Schnittstellen von dreidimensionalen räumlichen Gebärden. Das Framework unterstützt sowohl die Gebärdenakquisition als auch die Modelierung von einzigartigen physikalischen und charakteristischen Eigenschaften individueller Gesten. Die Akquisition erfolgt durch einen Mehrsensorenansatz, der visuelle Sensoren und Körpersensoren kombiniert. Neue Gebärdeneingabegeräte, die beide Sensorentypen verwenden und dadurch unterschiedlichste Gebärdeneingaben verarbeiten können, werden präsentiert. Die Gebärdenmodelierung unterstützt sowohl die Registrierung und Auswertung von dreidimensionalen räumlichen Gebärden, als auch deren Erkennung. Eine abschliessende Evaluierung des Frameworks zeigt dessen Fähigkeiten zur Gebärdenerkennung und Erlernung, und dessen Einsatz zur Entwicklung von einzigartigen dreidimensionalen räumlichen Gebärdenschnittstellen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides the motivation for this dissertation. In Section 1.1 the motivation of using 3D spatial gestures for human computer interaction is briefly discussed. The major contributions are listed in Section 1.2. In Section 1.3 an overview of the gesture processing pipeline is discussed. Section 1.4 gives the overview, and organization of this thesis.

1.1 Motivation

With advances in sensor and network technologies, physical environments are getting embedded with computer systems [CFBS97]. In recent years many researchers have recognized the value of such new computing environments and have suggested computing paradigms such as mobile computing, ubiquitous computing, and wearable computing. They focus on understanding human capability and maximizing the opportunity to access digital information regardless of time and location.

On the other hand, computer graphics and display technologies have advanced and transformed parts of our living and working environments by connecting the physical and virtual world. Virtual Reality (VR) systems [RWC+98, GWN+03b] enable users to immerse completely in virtual environments by integrating multiple projection devices and screens into our physical environments. Augmented Reality (AR) and Mixed Reality (MR) systems go one step further by merging computer generated objects and our physical environment.

With these technical improvements, several new computational environments have been proposed. These environments are typically embedded with sensors and equipped with various displays such as traditional monitor displays, large-screen displays [KFA+04], 3D immersive displays [GWN+03a, CNSD93], tabletop displays [RS99], and hand-held displays [BHI93, RS99]. The names of these environments also vary depending on the main purpose of a target application (e.g. responsive environments, collaborative environments, and smart environments).

In such ubiquitous computing environments, the standard human-computer interaction models such as command-line, menu-driven, or Graphical User Interface (GUI)-based, are not always optimal. Therefore, more natural and intuitive inputs have been pursued. In this context, the use of 3D spatial gestures has emerged as an attractive solution for more natural and intuitive human computer interaction under less constrained environments.

Various applications have driven the research of a large scientific community including computer vision, human-computer interaction, and computer graphics. For instance, gesture is used as a means of pointer and manipulator in different displays namely desktop monitors, tabletop displays, large wall displays, and immersive VR displays. Gesture is also applied to command execution aiming for complete eyes-free interaction in mobile and wearable computing domains [BLB+03].

Although a lot of solid research has been done, it has been mainly designed to test the system performance for gesture recognition. Little is applied to the actual use. Therefore, the success of current gesture interfaces has been restricted to relatively controlled environments requiring a special hardware setup and well-defined gestures. Their practical utility has mostly been ad-hoc and not presented within a generative design framework.

Moreover, as a main limitation, gestures exhibit a great deal of human variability due to differences in user performance and physical condition [PKE+06]. For instance, one person may prefer performing gestures in a fluent manner (i.e. smoothly without any hesitation) while another may take more time in one gesture and pause between gestures. This inter- and intra- personal variation sets limits to the recognition of gestures. 3D spatial gestures preserve the large gestural variations since they are usually performed in 3D space without any restriction on the movement.

Most of the available gesture input systems use a limited set of simple gestures to accommodate different users. Only pre-defined and fixed sets of 3D gestures are available in the target application and the addition of new gestures is typically not possible. Thus, users have to spend time and effort to rehearse the programmed gestures to match the system's parameters. All these limitations inhibit the growth of gesture vocabulary and make it difficult to fully utilize all the advantages of gesture in HCI. To utilize individual body characteristics and physiologies, end-users are required to create and register new 3D spatial gestures to the gesture vocabulary and to learn how to perform gestures.

This dissertation proposes a design framework for 3D spatial gesture interface that can be used as a prototyping tool to facilitate and encourage the design of a new gesture interface. The framework provides all the necessary hardware and software components within a single framework upon which a gesture interface can be developed. In addition, it promotes a systematic approach to design 3D spatial gesture interface. This dissertation can be considered as the first to comprehensively present and define 3D spatial gesture interface as a theoretical and technical study within a single unified framework including hardware and software in general.

There are two main research issues: the gesture acquisition for the optimal set of gesture features combining visual and body sensors, and the gesture model to register, recognize, and evaluate gestures by putting emphasis onto the extensibility of the model. During the acquisition, gestures are acquired by different sensor technologies and processed to find necessary information (e.g. gesture types and dimensional information). The gesture model facilitates the use of 3D spatial gestures by enabling users to create and learn gestures with their style and to find optimal gestures suitable to their preference and physical condition.

1.2 Contributions

In the framework, the lower level details of the gesture processing techniques are abstracted and provided as a suite of configurable hardware and software tools. As shown in Figure 1.1, we defined three major design components of 3D spatial gesture interfaces: *input devices*, *gestures*, and *contexts*. During the design process, a proper gesture input device which considers target applications and gestures should be selected or developed. Gestures can be designed for end-users in terms of naturalness, adaptability and coordination. Finally, contexts should be designed to provide a set of context information where and when the gestures are used along the application scenarios. Throughout this dissertation, each component and their relationships within a unified framework are analyzed.

The major contributions of this dissertation are:

- **Gesture acquisition using a combination of visual sensors and body sensors.** For the acquisition of 3D spatial gestures, the current available sensor technologies are invested namely wireless sensor networks, visual sensors, and Micro Electro Mechanical Systems (MEMS). To this end, visual and body sensors are combined and used to support 3D spatial gesture based inputs. The combined sensor data also provides robust and invariant features for gesture recognition and evaluation.
- **Versatile 3D spatial gesture input devices: *mWire* and *mCube*.** Gesture input devices are developed using visual sensors and body sensors. The *mWire* integrates visual sensors and body sensors as a semi-wearable input device. It can be worn on the wrist or held in the hand. The *mCube* supports the combined hand-held and desktop interaction so that it can be used in a variety of display platforms such as monitor display, large screen display, and table-top display.
- **3D spatial gesture representation scheme: *Motion Chunk*.** *Motion Chunk* was developed as a standard unit to represent 3D spatial gestures. *Motion Chunk* represents continuous human gesture with a sequential combination of chunks. As the core representation of the gesture model, *Motion Chunk* serves gesture design and the automatic segmentation, registration, recognition, and evaluation.

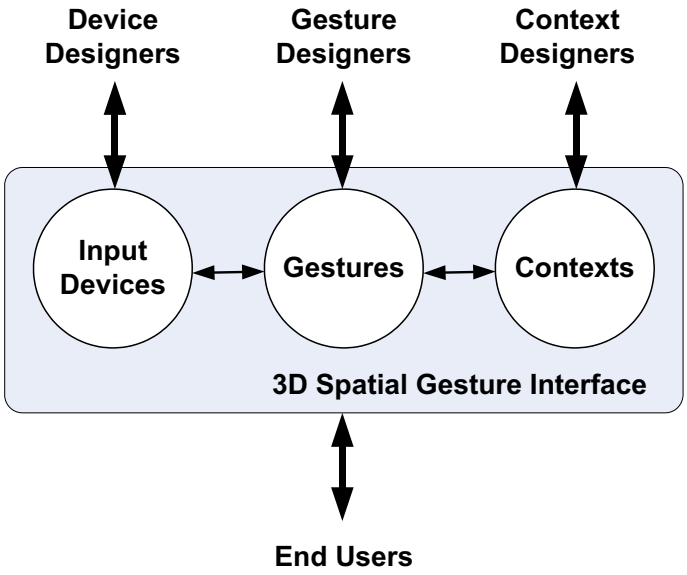


Figure 1.1: Three design components for 3D spatial gesture interfaces: input devices, gestures, and contexts. Each design component can be used by respective design experts: device designers, gesture designers, and context designers. Ultimately, end-users need to optimize the 3D spatial gesture interface by choosing or modifying the three design components.

- **Gesture model using DTW and HMMs for gesture registration, evaluation, and recognition.** The gesture model is developed based on *Dynamic Time Warping* (DTW) and *Hidden Markov Models* (HMMs). The model deals with the noisy nature of the measurements while preserving the important nature of gestures without loosing the discriminant power between different gestures. HMMs provide efficient gesture registration and recognition accounting for dynamically time-varying gesture sequences. The explicit distinction of postures and dynamic gestures within the HMM model facilitates the addition of new gestures in a flexible and convenient way. On the other hand, DTW allows us to test one of the fundamental usability goal of spatial gesture interfaces that do not require excessive gesture training data. DTW also provides gesture evaluation that compares an input gesture with a template gesture.
- **Experimental evaluations to validate the framework.** Two experimental evaluations are conducted to validate the proposed approaches. The first evaluation explores the selection of gesture features and recognition methods under various conditions (e.g. different locations and different users). The second evaluation analyzes user performances in learning 3D spatial gestures.

- **A spatial context aware gesture interface.** A spatial context aware gesture interface is developed to exemplify the use of the framework in designing a practical gesture interface. This interface recognizes a system action (e.g. commands) by integrating gesture information with additional context information within a probabilistic framework. Two ontologies of spatial contexts are introduced based on the spatial information of gestures: *gesture volume* and *gesture target*. Prototype applications are developed using a smart environment scenario that a user can interact with digital information embedded to physical objects using gestures.
- **A versatile gesture interface.** A versatile gesture interface is developed as another case study. A unique gesture input device called mCube is invented to support both desktop and hand-held interaction at the same time. This interface using mCube supports gesture-based inputs namely a command input, a manipulative input, and tool and menu selections in different display platforms. Users do not have to switch other input devices during the interaction so that we can improve the work flow of interactions in ubiquitous computing environments.

1.3 Gesture Processing Pipeline

This section provides a high-level overview of 3D spatial gesture processing for overall understanding of the required components in a gesture interface. First, we present the gesture production model showing how a gesture is created and produced from a user and observed by a system. Second, in the gesture production model, we describe the inverse process of the gesture production model which estimates the used mental concept from the observed gestures.

1.3.1 Gesture Production

We use a broad range of gestures from the simple to the complex. We communicate with people with gestures and manipulate physical objects located in our environment. Some gestures are so natural that we can perform them with little thought or learning. On the other hand, some gestures are designed by a person or a group for a special purpose such as a visual communication in military operations and sports. Theses artificial gestures might need a relatively longer period of learning to acquire those gestures.

Generally speaking, gestures are created and planned from the user's mental concept. Once a mental concept is set, the user considers the given various contexts (e.g. locations and situated application tasks). Gestures are performed and expressed through the motion of different body parts such as the arms and hands. Using sensors, gestures can be observed as two types of signals: *continuous* and

discrete. Appropriate sensors should be selected to meet the required continuous quantities and the degree of freedom in the gesture movement.

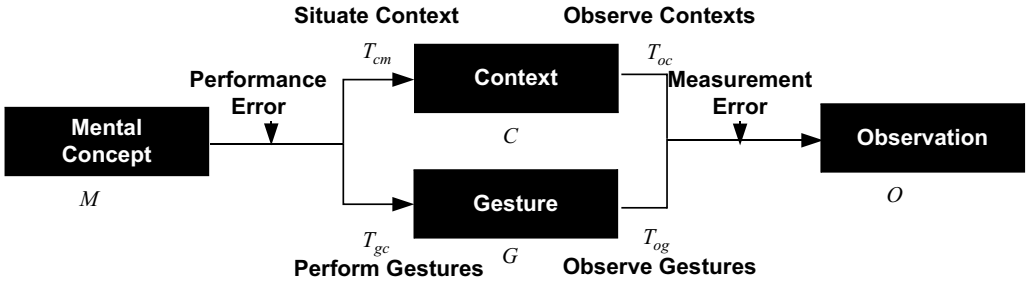


Figure 1.2: The conceptual model of gesture production. Gestures originate from the mental concept M, are situated (T_{cm}) with context C and expressed (T_{gc}) through gestures G, and are observed (T_{gp}) as observation of signals O. There are two error sources: performance error of the user and measurement error of the system.

The conceptual model of gesture production is illustrated in Figure 1.2 and summarized in the following form:

$$C = T_{cm}M, \tag{1.1}$$

$$G = T_{gm}M, \tag{1.2}$$

$$O = \{T_{og}G, T_{oc}C\}. \tag{1.3}$$

Transformations T can be viewed as different models: T_{cm} is a model of context given the mental concept M and T_{gm} is a model of gestures given the mental concept M. T_{og} and T_{oc} describe how signals are formed as an observation O given gesture G and context C respectively. The sensors define the parameter space of each model.

During the gesture production, there are two error sources: *performance* and *measurement* as depicted in Figure 1.2. The performance error is driven from the user's movements when performing a gesture. When users perform a certain gesture, the actual response often differs from their expectation due to imprecision of control their body and even inability to exactly generate the desired postures and gestures. The measurement error is generated from the sensor system because of noises and distortions.

1.3.2 Gesture Perception

In the previous section, we discussed the outputs of gesture performance which result in a stream of signals. *Gesture perception* is the inverse process, the conversion of gestural observations to the mental concept. There are two major processes in gesture perception: *analysis* and *recognition*.

Analysis

The goal of analysis is to estimate the parameters of a gesture H and a context S using measurements from the observation O of the performed gesture. However, the direct mapping (T^i_{oh} and T^i_{os}) of observations to parameters would be extremely complex. In practice it is more convenient to introduce an intermediate step to this process. For instance, one would have to select features to describe gesture and context, given a sequence of observation of gesture. More specifically, we can infer how a hand gesture parameter H is represented in observation and what spatial context information S (e.g. physical locations and objects) is related to the hand gesture.

This is depicted in Figure 1.3,

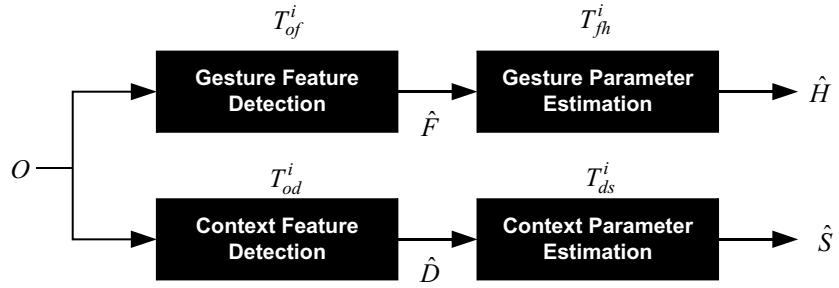


Figure 1.3: The analysis stage for feature detection and parameter estimation in terms of gesture and context. Gesture features $\hat{F}$ and context features $\hat{D}$ are extracted from observations O, then gesture parameters $\hat{H}$ and context parameters $\hat{S}$ are estimated from the extracted features individually.

$$\hat{F} = T^i_{of} O \tag{1.4}$$

$$\hat{H} = T^i_{fh} O \tag{1.5}$$

$$\hat{D} = T^i_{od} O \tag{1.6}$$

$$\hat{S} = T^i_{ds} O. \tag{1.7}$$

There are two sequential tasks involved in this analysis. The first involves the detection of relevant features estimating parameters for the chosen gestural and contextual model. Since features are highly related to the sensor resources, it is necessary to select appropriate sensors for the target gesture and context. Moreover, both gesture and context models can be estimated from the same observations so that we can minimize the number of sensors.

There are two major features for gesture: *position* and *rotation*. These gesture features should be invariant to the different positions and orientations so that the gesture model can be used varying a user location. While there could be various

context features for more accurate gesture perception, two typical ones (*location* and *reference*) can be closely related to 3D spatial gesture.

During the feature detection, it is often necessary to use stochastic estimation from noisy sensor measurements. One of the most well-known tools is the Kalman filter that minimizes the estimated error covariance using a predictor-corrector type estimator.

The second task is parameter estimation using the detected features that can be decided based on the characteristics of gesture model. For instance, if only gesture recognition is pursued, the exact knowledge of gesture might be unnecessary. On the other hand, the detailed gesture knowledge is required when the model is used to evaluate the quality of gesture.

Recognition

There are three recognition phases: *gesture*, *context*, and *concept* as depicted in Figure 1.4.

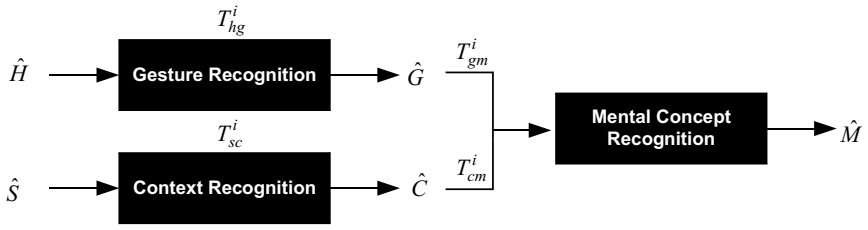


Figure 1.4: Three recognition phases: gesture, context, and concept. The estimated context parameters $\hat{C}$ and the estimated gesture parameters $\hat{G}$ are combined and used to recognize the underlying concept of the gesture $\hat{M}$.

Gesture recognition identifies the type of an input gesture by interpreting the gesture parameters. In accordance to the previous notation, this can be formally written as

$$\hat{G} = T^i_{hg}\hat{H} \tag{1.8}$$

In general, the task of gesture recognition is to find one of the gesture templates that most closely matches the input gesture. To recognize simple gestures, a heuristic approach that observes simple trends or peaks in one or more of the sensor values is used. However, 3D spatial gestures are spatio-temporal actions that are performed in 3D space with various time durations. Therefore, we need to handle temporal nature of gesture using techniques from the field of time series analysis such as HMMs [Rab89] and DTW [Cor01].

Context recognition also the phase to find out the situated context when a gesture is performed based on the context parameters.

$$\hat{C} = T^i_{sc}\hat{S} \tag{1.9}$$

Concept recognition also finds out the mental concept of a user which initiates the performance of a gesture. The outputs of the mental concept recognition can be mapped to a certain system action (e.g. command) used in applications. The mental concept M can be inferred from their observations O using a suitable model T_{om} , or

$$\hat{M} = T^{i}_{om} O \tag{1.10}$$

where T^{i}_{om} denotes some (inverse) mapping from observations O to mental concepts M. Concept recognition requires the combination of multiple observations and facts. In our case, we combine the outputs of gesture recognition and context recognition using Bayesian networks (BNs) [HGC94].

1.4 Overview and Organization of Thesis

Figure 1.5 shows an overall structure of the framework. There are four major components: two technical components (*gesture acquisition* and *gesture model*) and two functional components (*spatial context aware gesture interface* and *versatile gesture interface*).

The framework begins with gesture acquisition which captures 3D spatial gestures through visual sensors and body sensors. We extensively studied the use of accelerometers as a body sensor, and cameras as a visual sensor. A unique gesture input device called mWire is provided with the framework. The mWire supports users easily test different body sensors and the combination of visual and body sensors. A wide range of gesture information from detailed to approximate can be acquired from the combined sensor data. For the required tasks of target applications, gesture features are driven from the acquired sensor data.

The second technical component is the gesture model that consists of several sub modules. First, the model represents the acquired signals using a gesture unit, called *Motion Chunk* into which gestures are segmented and stored. The three subsequent modules namely gesture registration, recognition, and evaluation are processed using two signal analysis techniques (DTW and HMM).

Our framework contains two unique 3D spatial gesture interfaces as a functional component. These interfaces use the gesture acquisition and model and they are also equipped with additional hardware and software components for the required inputs by target applications.

The context-aware interface contains modules to represent spatial contexts, and supports users registering their own contexts into the system. This interface uses the Dynamic Bayesian Network (DBN) to recognize programmed system actions combining the the results of gesture recognition and the selection of spatial contexts. A versatile interface provides both MDOF manipulations and gestures commands. A unique versatile input device mCube supports various interaction techniques in both desktop and hand-held positions.

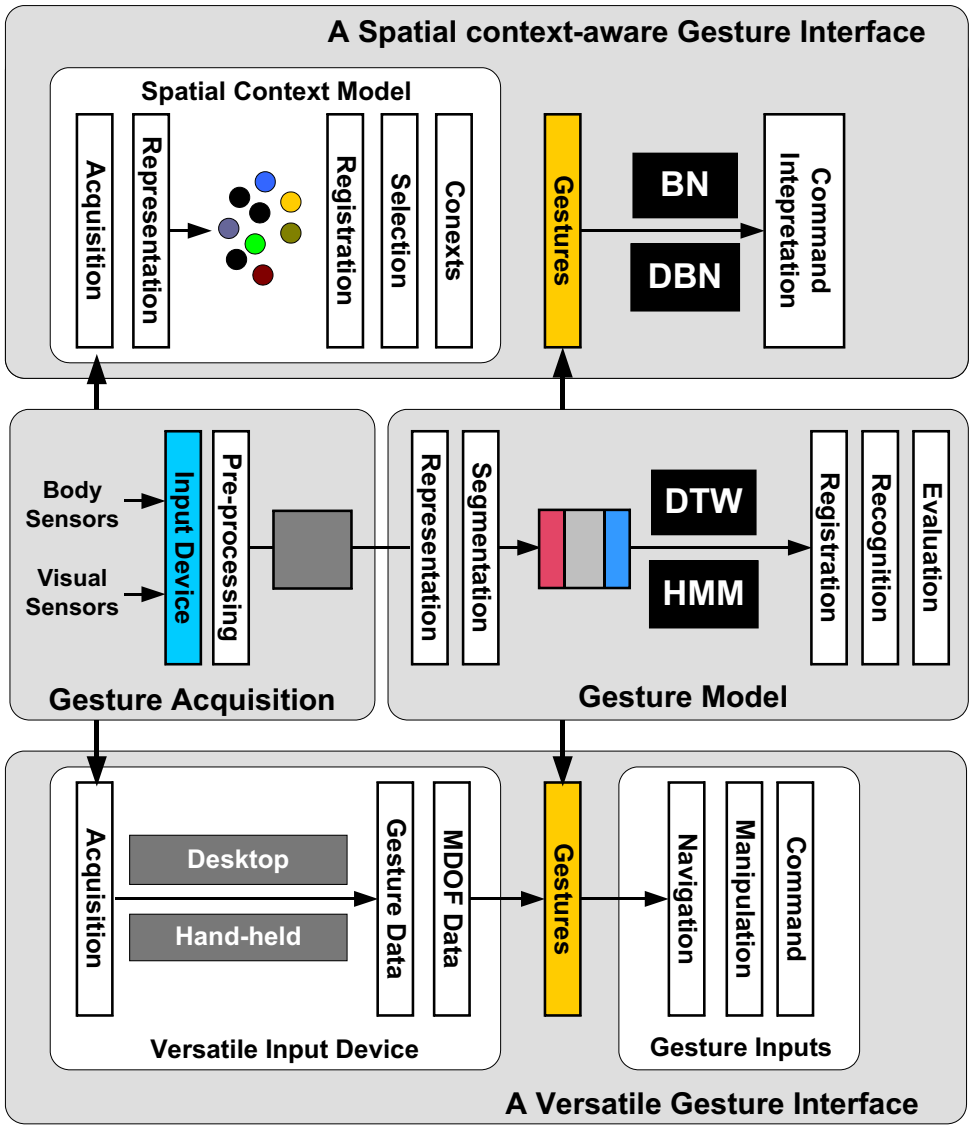


Figure 1.5: An overview of the design framework for 3D spatial gesture interfaces. There are two technical components (gesture acquisition and gesture model) and two functional components (spatial context aware gesture interface and versatile gesture interface). During acquisition, gestures are acquired by body and visual sensors. Then, the acquired data is represented and segmented based on the structure of motion chunk. The model supports the registration, recognition, and evaluation of 3D spatial gestures using DTW and HMMs. Two gesture interfaces use the acquisition and gesture model to provide the required gesture-based inputs.

We describe these four components in the following chapters. The dissertation is outlined as follows:

- **Chapter 2** presents several definitions of 3D spatial gesture with related terms in Section 2.1. Section 2.1.2 provides a brief background on gesture classification. Then, we discuss the previous work that constitute the context of this dissertation through three subsequent sections. First, we introduce a variety of sensors (Section 2.2) and devices (Section 2.3) that can be used to acquire 3D spatial gestures. Section 2.4 describes two different approaches for gesture recognition. Section 2.5 presents the previous work for gesture-based inputs categorized into four groups: spatial, symbolic, manipulative, and expressive. Section 2.6 reviews four practical applications of 3D spatial gestures: computer aided design, game control, motion training, and digital art.
- **Chapter 3** describes our acquisition system combining two different types of sensors (visual and body). Section 3.2 explains the characteristics of accelerometers as a body sensor. Section 3.3 shows the use of cameras as a visual sensor and presents visual features that are invariant to the location and orientation of a user. Section 3.4 presents a gesture input device called mWire that facilitates the use of visual sensors and body sensors.
- **Chapter 4** explains the gesture model which is a main technical component of the dissertation. After a short overview of the framework (Section 4.1), we describe sub-modules subsequently including gesture segmentation (Section 4.3) and representation (Section 4.2). Section 4.4 describes algorithmic details of Dynamic Time Warping (DTW) that measures the similarity of two different time series data. In Section 4.5, Hidden Markov Modeling (HMM) is described with a set of core algorithms. We also describe how they are tailored for three major functionalities: gesture registration (Section 4.6), evaluation (Section 4.8), and recognition (Section 4.7).
- **Chapter 5** describes two experimental evaluations to validate our framework. These experiments show how the developed methods can be performed in gesture recognition and instruction tasks. In the recognition task (Section 5.1), we analyze the performance of our gesture recognition methods. In the gesture instruction task (Section 5.2), we present the experimental process, based on a martial art training scenario, and the costs and benefits of learning complex 3D spatial gestures.
- **Chapter 6** presents a context-aware 3D spatial gesture interface. Section 6.1 gives a short overview of the system. In Section 6.2, spatial contexts (gesture volume and target) are presented as a specific context for 3D spatial gesture. This section presents the methods to register and select the spatial contexts, to recognize a final system action. Section 6.4 shows two prototype applications: the smart home environment and smart museum environment. Each application shows how users can interact with various objects using 3D spatial gestures in digitally augmented environments.
- **Chapter 7** presents a versatile 3D spatial gesture interface that supports

various gesture-based inputs in both desktop and hand-held positions. After a short system overview (Section 7.1), we introduce a novel gesture input device called mCube designed to provide the required versatility of the interface (Section 7.2). Section 7.3 demonstrates some of interaction techniques such as command inputs and Multi-Degree-Of-Freedom (MDOF) manipulative inputs. Finally, we show the results of our experimental evaluation for the mCube device in Section 7.4.

- **Chapter 8** concludes the dissertation. Section 8.1 provides a summary of the important contributions and potential effects on design of 3D spatial gesture interfaces. Section 8.2 mentions potential future directions in terms of gesture input devices, gesture recognition, and user evaluations.

Chapter 2

Background

This chapter presents the literature review regarding gesture taxonomy, gesture acquisition and recognition technologies, and gesture-based inputs and application areas. It begins with a brief summary of the different definitions and classifications of gesture in Section 2.1. Section 2.2 introduces available sensor technologies, and Section 2.3 presents a set of input devices for gesture acquisition. In Section 2.4, we describe two pattern matching techniques for gesture recognition: statistical analysis technique and template matching technique. Section 2.5 presents gesture-based inputs categorized into four groups: spatial, symbolic, manipulative, and expressive and Section 2.6 describes application areas for 3D spatial gestures.

2.1 Gestural Taxonomy

There have been extensive research on the analysis of human gestures focusing on the conceptual understanding of gesture. The research domain includes linguistics, anthropology, cognitive science, psychology, neurology, choreography, and physical therapy. Even though their approaches do not aim to justify their theory by developing a computational model, they provide qualitative and theoretical deep analysis on human gesture. This section summarizes various definitions of gesture and some approaches for gesture classification, and defines 3D spatial gesture and its important characteristics.

2.1.1 Definitions of Gesture

There have been various definitions of gestures. For instance, McNeill [McN95] defined gesture as "movements of the arms and hands which are closely synchronized with the flow of speech." He focuses the relationship between speech and gesture. More broader definition of a gesture can be found in a American Heritage Electronic Dictionary: "a motion of the limbs or body made to express thought or to emphasize speech." While these definitions can be useful for a general purpose,

we need a more clear definition of gesture to apply that term to computational implementation for human computer interaction.

There are also some more concrete definitions of gesture. Cadoz [Cad94] defined three functional roles of human gesture: *semiotic*, *ergodic*, and *epistemic*. He defined the gestural channel as a means of action on the physical world as well as communication means. The semiotic gestures are used to communicate meaningful information. The purpose of ergotic gestures is to create and manipulate artifacts. The epistemic gestures are used to learn the environment through tactile or haptic exploration.

Bobick [BI98] described a gesture with three relevant terms: *movement*, *activity* and *action*. He defined movements as the most primitive form of motion that can be interpreted semantically. Activity is a sequence of either movements or static configurations. Actions are the high-level entities that people typically use to describe what is happening.

Luciani et al. [LEC+06] compared three relevant terms: *gesture*, *motion* and *action*. While motion and gesture are similar terms, and actually two words are almost interchangeable, they distinguished motion from gesture. In their theory, motion is considered as a movement of a physical object (e.g. human body itself, input device attached on the human body or held by the human) and as a result of the performance. On the other hand, gesture is an input that causes a performance so that gesture is more related to the mental concept of a performer. Action is a high level result (e.g. "to drink a glass of water") caused from a set of input gestures [SW84]. Each action can be executed by many different resource movements and described at a symbolic level.

Laban Movement Analysis (LMA) was introduced as a method for observing, notating, and interpreting human movements [Zha01]. Its notation system is used to describe the body-centered movement showing how each motion is toward, away, around, or in another way from the center of the body. The final motion can be defined in relation to the body along a coordinate axis with a certain dimensional scale. Using LMA, various applications have been implemented to improve awareness, efficiency, and ease of movement.

2.1.2 Classifications of Gesture

Several classifications of gestures have been suggested in the literature dealing with psychological aspects of gestures. Kendon [Ken80] analyzed the relationship between the gesture and the speech. He defined a gesture as a label for actions that have the features of manifest expressiveness. In his theory, gesture is integrated with speech in different levels with the amount of obligatory presence of speech in gesture, and categorized into five types: *gesticulation*, *language-like gestures*, *pantomimes*, *emblems*, and *sign languages*. From left to right, the obligatory presence of speech declines and the presence of language properties increases. The linguistic component of the expression present in speech is replaced by gestural

signs going from gesticulation to sign languages.

McNeil [McN95] provided a unified conceptual framework that includes both gesture and language. Three rules are addressed to synchronize gesture with speech: *phonological*, *semantic*, and *pragmatic*. In the phonological rules, the gestural stroke proceeds or ends at, but doesn't have to follow the phonological peak syllable of speech. Using the semantic rules, gesture and speech must cover the same idea if they co-occur. This rule can be applied when multiple gestures and speeches co-occur. For the pragmatic rules, gestures and speech serve the same pragmatic functions if they co-occur.

Rime and Schiaratura [RS91] proposed the gestural taxonomy consisting of *symbolic*, *dietic*, *iconic*, and *pantomimic*. Symbolic gestures have a single meaning within each culture such as an emblem like the "OK" gesture. Dietic gestures are used for pointing or directing the listener's attention to a specific event or a certain target object. Iconic gestures convey information about the size, shape or orientation of the object of discourse. For instance, iconic gestures visualize a flying path of a plane as moving their hand through the air. Pantomimic gestures are typically used in showing the movements of an invisible tool or object in a hand.

Quek [Que94] provided a rather complete gesture taxonomy focusing on hand movements. In his classification scheme, all hand movements are categorized into two classes: *gestures* and *unintentional movements*. If hand movements do not convey any meaningful information, they are considered as unintentional movements. On the other hand, gestures are classified into sub-classes: *manipulative* and *communicative*. Manipulative gestures are used to move and rotate objects. Communicative gestures are used for an inherent communicational purpose. These gesture types can be used either *acts* for the interpretation of movement itself or *symbols* for a linguistic role.

2.1.3 3D Spatial Gesture and Characteristics

Gestures can be categorized by the dimensions of the gestural space. For example, pressing a pressure-sensitive key can be considered a *1D gesture* and moving hands on the flat surface can be considered a *2D gesture*. A *3D gesture* is typically performed in 3D space. In addition to this dimensional definition, we can further define *3D spatial gesture* that preserves a certain spatial relationship between gesture and the environment where the gesture is performed.

3D spatial gestures are closely linked to *sensory-motor control skill* which is the basis for skilled human movements. 3D spatial gestures are routinely used in our everyday movements in the 3D world (e.g. manipulating physical objects such as driving cars, and throwing and catching a ball). In our movements, we use different body parts or other objects as a kind of external and internal stimuli that affect our perceptions and motor skills.

Among different human senses, proprioception plays an important role in performing 3D spatial gestures. Proprioception is the awareness of the position of

one's body parts, relative to other neighboring parts of the body [IvDC+00]. It provides a feedback solely on the status of the body internally, and indicates whether or not your body is moving with required effort, as well as where the various parts of the body are located in relation to each other. For instance, the ability to catch a baseball requires a finely tuned sense of the position of the joints, so that the eyes can concentrate on the ball and let the proprioception handle moving the body as needed to catch the ball.

2.2 Sensors

From the engineering point of view, every physical quantity can be measured through different types of sensors. To use gestures as an input, the performed gestures should be quantified through sensors in a certain range of precision. The type of a sensor is usually selected with the purpose of application, the types of gestures, and available infrastructure and cost. This section describes some of typical sensors that can be applied to capture 3D spatial gestures. We categorize the sensors into two groups: *body sensors* and *visual sensors*.

2.2.1 Body Sensors

Body sensors are usually attached on the body or held by the hand, and provide straightforward information from the body when the movements occur.

Data gloves [5DT] have been widely used to capture hand gestures including finger movements. Many projects use data gloves for hand gesture interfaces measuring directly hand and finger movements [FH95, BBL93]. These systems offer precision, a relatively large range of motion, and very fast update rates. However, glove-based approaches require the user to wear a cumbersome device and generally carry a load of cables that connect the device to a computer. This hinders the free movement of users.

To track the whole body, special body suits have been developed using optical or electromechanical tracker technologies [Mot,Pup]. Users wear special suits which are equipped with electromechanical body sensors, dots or small balls which serve used as markers. However, the main purpose of the body suits is to capture body motions for character animation. The body suit approach is too expensive and cumbersome as a gesture interface, to support more natural interfaces.

There have been several approaches to building an acquisition system that customizes different sensor types. One of the most popular sensors is the accelerometer which measures either acceleration or rotational angles along a certain axis. Benbasat and Paradico [BP02] proposed an inertial measurement framework for gesture recognition. While most approaches using inertial systems are developed in an ad hoc fashion for a specific application, their goal is to generalize a framework for gesture recognition using a compact inertial measurement unit

(IMU). The framework also provides a light-weight gesture recognition algorithm and scripting functionality to specify gestures and their combinations with system commands.

Amento et al. [AHT02] developed a small wristband style wearable device to acquire gentle fingertip gestures such as tapping, rubbing, and flicking. They use sound that travels throughout the hand and bone conduction when a fingertip is touched. Based on this sensor mechanism, they developed a unique bio-acoustic interface that recognizes some finger-tip based gestures. Rekimoto [Rek01] developed the *Gesturewrist*, a wristwatch-type hand gesture recognition device using acceleration sensors to detect forearm and hand gesture. Randell el al. [RAM+05] used the multi-layer fabric sensors, ElekTex [Ele] and designed *Sensor Sleeve* mounted in a garment sleeve. They used pressure activated elasto-resistive sensors to detect embracing, pressing and stroking gestures.

2.2.2 Visual Sensors

Visual sensors like cameras capture the shapes and properties such as texture and color. Facial gesture interfaces use visual sensors to capture facial and lip movements [NLP+02]. Bhatt et al [BLSB03] used visual sensors to implement a vision surveillance system. The system tracks a baby's mouth, hands and head using skin detection algorithms and recognizes whether the baby puts objects to his or her mouth.

Cameras are also used to capture hand gesture performed on desktop surfaces. Installed behind transparent or semi-transparent screens, cameras track the position of the hand and a hand-held input device [BKKK04b]. Cameras are installed above the desktop pointing downward and capture the 3D positions of the user's fingertips. Malik and Laszlo [ML04] introduced a *Visual Touchpad*, a low-cost vision-based input device that allows for two-handed interactions with desktop PCs. Even though the system requires a specific color on the planar surface, a variety of hand gesture interaction techniques can be designed.

Various hand and arm gesture interfaces have been developed based on cameras technologies [SP, SAA00, CB96, WF98, BOP97]. In general, the systems compute the positions and hand shapes, and track movement trajectory in 3D space. Furthermore, some systems have been developed to recognize human activity [AP04] or analyze body postures [BLK01, BLK03] by tracking full body gestures with cameras. However, to acquire 3D spatial gestures using cameras, a set of algorithms is needed to derive visual features from the acquired camera images [SAA00, CB96]. The computing time of these algorithms is often major bottleneck to developing a real-time application.

Starner et al. [SAA00] developed a camera-based wearable device called a gesture pendent for capturing hand gestures. While most of approaches install cameras on the surrounding infrastructure (ceiling, wall, or table), they placed a camera in the pendent to capture hand movements on the pendent. The system

efficiently reduced the gesture candidates by combining other context sources. Several applications demonstrates the use of the device such as medical diagnosis, therapy, and emergency services.

2.3 Gesture Input Devices

The previous section described different sensors to quantify human gestures. This section introduces input devices for human gestures. Input devices should be carefully selected or designed considering types of gestures and available hardware setups such as display and interaction zones. We categorize gesture input devices into two groups based on the type of major operating positions: *desktop devices* for operation on a desktop surface and *hand-held devices* for operation in 3D space.

2.3.1 Desktop Devices

A computer mouse is a typical example of a desktop device that provides 2D coordinates on the monitor screen for graphical user interface. Stroke-based gestures often use a computer mouse for shortcut command inputs. The traditional mouse functionalities also have been extended by embedding additional sensors [BP98, Kur93, Ven93, BBKF97, HSH+99]. The main purpose is to provide more efficient menu selection in complex tasks and to overcome the limitations of a computer mouse. For instance, Balakrishnan and Patel [BP98] designed the PadMouse that combines a touch pad with a conventional computer mouse. The Rockin' Mouse was introduced to enhance the manipulation of 3D virtual objects [BBKF97].

Another typical desktop device is isometric devices such as Spaceball [Spaa] and the Spacemouse [Spab]. These devices measure the force or torque of the sensor as manipulated by fingertips [FHSH06]. Such devices are applied to get the additional degree of freedoms (DOFs) and use these to overcome the inherent limitations of a computer mouse for graphics applications in 3D environments. Even though these devices do not capture the movements of the hand, they have potential to obtain detailed finger movements and provide a force feedback to the fingers.

Digital pen technologies [Wac] are used for acquiring pen movements on a tablet surface. The acquired 2D coordinates are processed for stroke-based gestures invoking command inputs [Jr01]. The recognition algorithms are the same as the computer mouse. However, digital pens enable users to use handwriting skills and make the stroke more natural and accurate than a computer mouse. Digital pens also provide additional information of handwriting such as the number of strokes, the stroke order, and the direction and velocity profile of each stroke which can be used for designing new stroke-based gestures.

Rekimoto [RS00] designed a multi-functional input device called *Toolstone*.

Using this device, users explore their manipulative skills to control computer mediated tasks such as scene navigation and object manipulation. They used the digital pen and tablet technology [Wac] to measure MDOFs such as the position, orientation, and tilt angle of the device. Since the pen sensor is embedded in the device, users should operate the device on the tablet surface.

There were also some approaches to use bare hands instead of using additional devices on the desktop. For this purpose, several touch and pressure sensitive devices were developed supporting direct finger pressure on the screen surface [Rek02, DL01]. They usually transform the display surfaces into multipoint touch sensitive surfaces combining input and output in a co-located manner. This enables users to directly use freehand gestural interaction, involving fluid touches, on a wall or tabletop surface.

2.3.2 Hand-held Devices

The hand-held devices have been used usually when the desktop surface is not available. We categorize hand-held devices into three groups based on the type of used sensors: *electromagnetic*, *accelerometer*, and *camera*.

Electromagnetic Devices

The notion of the *Flying Mouse* [WJ88] was introduced to categorize 3D input devices such as the Wanda^{TM} [Wan] and the 6D Mouse^{TM} [6DM]. Input devices based on electromagnetic sensing technology [Flo] are able to provide six degrees of freedom (6-DOF), three for X, Y, and Z translation and three for 3D rotation (yaw, pitch, and roll).

Such 6-DOF interaction devices are integrated in immersive virtual environments, like the CAVE^{TM} [CNSD93]. which supports virtual scene navigation and object manipulation [Zha98, Gro02]. For instance, the *Cubic Mouse* [FP00] was designed by embedding an electromagnetic sensor into a cube-shaped box. Three perpendicular rods pass through the center and buttons are located on the top for additional control. Users can navigate a virtual world by rotating the device and pushing and pulling the rods to determine constrained motion along the corresponding axes.

The EGG (Elastic General purpose Grip) [Zha98] was developed to support an elastic translational input. A set of elastic springs are combined with the electromagnetic sensor. The Fingerball [Zha98] enables users to roll a 3D virtual object by controlling a ball style handle with fingers during translation in space.

Electromagnetic sensors provide rather fast and accurate data, but the range of sensing is limited depending on the strength of magnetic resource. In addition, magnetic trackers are rather expensive and need a rather big infrastructure. Thus, such 6-DOF interaction devices have been mainly used for specific

applications (3D modeling and navigation) [FP00] in 3D immersive display platforms [CNSD93].

Accelerometer-based Devices

The unique characteristics of accelerometer have been explored in building new input devices specially for gesture-based inputs. Even though accelerometers do not provide complete 6-DOF, these sensors have several advantages over other sensing technologies. For instance, small size enables sensor embedding into other objects. Several approaches have embedded the accelerometer to everyday objects namely clothes, watches, badges, pendents and shoes [Rek01, PHH99, SAA00, SGB99, FMea99]. The main goal is not to minimize the problem of restricting the user's movements. Many of them are used because accelerometers are relatively low in cost and their cost continues to decrease.

Using accelerometers, Laerhoven et al. [LVS+03] presented a low-cost and practical approach to build a gesture input device using a cube shape object. They combined sensors, processors, batteries and wireless communication modules. Their system captures a set of simple gestures associated with the manipulative states of the cube such as shaking, twisting, and knocking. Keir et al. [KPE+06] developed a 3D spatial gesture input device called *3motion* with a general-purpose software development kit. The device contains a 3-axis accelerometer and transmits continuous gesture streams to a host device via wireless Bluetooth technology.

Tuulari and Ylisaukko-oja [TYo02] introduced a light, matchbox-sized device called SoapBox for 3D gesture input. The device contains a set of basic hardware components namely processors, pre-defined sensors, and wireless and wired data communications. The pre-defined sensors include a three-axis acceleration sensor, an illumination sensor, a magnetic sensor, an optical proximity sensor and an optional temperature sensor. The device is particularly designed to support users integrating other types of sensors.

Tsukada and Yasumura [TY02] proposed *Ubi-Finger* for a gesture interface in a smart environment. Perng et al. [PFea99] developed a glove equipped with six 2-axis accelerometers on the fingertips and on the back of the hand to capture hand movements. Using this glove, they developed a text-editor application to type a letter of the alphabet using hand gestures.

Accelerometers are often embedded to various mobile devices like cellular phones, MP3 players, and wristwatches. They take the feasibility and advantages of gestural inputs over other input technologies (keypads and voice commands). As an example, Wigdor and Balakrishnan [WB03] presented the *TiltText* interface. Users can scroll texts on the small screen display of a mobile phone by tilting a device in one direction.

Camera-based Devices

Several input devices use cameras particularly for tracking the position of the device. For instance, the VisionWand [CB03] was introduced as a hand-held input device to control 2D objects in large-screen displays. Two cameras are used for tracking the 3D positions of the two color ends of the wand. Even though the device is simple and cost-effective, tracking errors often occur because of the different lighting conditions. Bae et al. [BKKK04a] developed a gesture-based modeling system where designers can manipulate virtual curves on a large display screen using two hands. They created a graspable input device using a transparent groove embedded with a light-emitting diode (LED) in the middle. Cameras are installed behind the wall and used to track the position of the device.

Some devices use other sensors with cameras. For instance, the XWand [WS03] uses both visual and embedded sensors to support pointing gesture and simple command gestures. Users can select an object in a living environment by pointing and control its functions with a set of command gestures. For robust positional tracking, the device is equipped with one infrared LED (IR LED) at the end of the wand and the angular tracking is done by the combination of accelerometers and magnetometers. Agarawala and Carpendale [Tse04] developed a MDOF input device called *SuperSkewer* for 3D interaction on a large wall display. Similar to the XWand, they placed two sets of infrared LEDs on the ends of a seven inch rod to minimize the noise in tracking 3D positions.

2.4 Gesture Recognition Techniques

Recognizing human gestures is a rapidly growing sub-field and has become more prominent in the HCI community. Different pattern classification techniques have been developed which can be applied to recognize human gestures [RHS01].

Gesture recognition is usually accomplished with an algorithm that matches an input gesture signal to a set of stored template signals. Gesture features are first acquired from available sensors and used for estimating the states of a chosen gestural model. However, various aspects such as uncertainties in the measurements, systematic error, and feature generations must be considered.

This section presents two of the most important approaches for gesture analysis and recognition: *template matching* and *statistical analysis*.

2.4.1 Template Matching Techniques

Template matching techniques are the simplest and most straightforward method for recognizing gestures. This technique computes the difference (or distance) between input and template gesture patterns. One popular template matching technique is a *Dynamic Time Warping* (DTW). DTW has been extensively used to

recognize spoken words [RJ93]. DTW supports non-linear time alignment differences between two patterns using the Viterbi algorithm. Thus, DTW has been used for recognizing gestures performed with variable speed [Cor01].

Using DTW, gestures are recognized by comparing an input gesture to a set of template gestures and measuring any similarity between them. With this method, there must be a representation stage where the raw sensor data is stored as a template. DTW is a non-parametric technique employing the original gesture frames directly for gesture recognition. After DTW measures the similarity between the input and the templates of values, the input can be either admitted as a member of the same class as the template to which it is most similar (or nearest), or rejected as belonging to none of the possible classes if the measurement is higher than the similarity threshold (too far from the nearest template).

DTW cannot accommodate the probabilistic nature of the signal and it is still challenging to make the templates adaptive. Adaptability could play a critical role in the system's performance, since most gestures aren't reproduced consistently even by the same user. With different users, the variation becomes even greater. Also, template matching does not have the formal and iterative approach to training that statistical classifiers and neural networks have. However, DTW has several potential benefits. DTW works even when only one training dataset is available. DTW is also easy to develop, computationally efficient, and is very accurate.

Several applications use DTW for gesture recognition, and some of them provide useful extensions of the DTW to fulfill their specific tasks [AAS05, Lip91]. For instance, the Dynamic Space-Time Warping (DSTW) algorithm [AAS05] was proposed to combine spatial domain alignment with the time domain alignment of the original DTW. A multi-resolution template matching technique [Lip91] was introduced to achieve more efficient computation comparing the templates with multi-resolutions in different levels. During the recognition process, the templates are examined first at the lowest resolution and only if successful at that level would the template proceed to matching at a higher resolution level.

2.4.2 Statistical Analysis Techniques

Functionally, statistical analysis techniques operate in the same way as the template matching techniques. However, the mapping function uses statistical methods, such as Bayesian likelihood theory, to decide which class the input gesture most likely belongs to. While there are several different approaches to the statistical analysis techniques, in this section, we focus on Hidden Markov Models (HMMs) [Rab89] that have been extensively studied in continuous speech recognition research.

In gesture recognition, HMMs are used to represent the temporal variations of the gesture sequences in the probabilistic framework. Using HMMs, each gesture can be modeled separately so that new gesture models can be added independently

to existing ones. One of the drawbacks of the HMMs technique is that we need a significant amount of training data to parameterize and condition the HMMs-based gesture model.

One of the early applications using HMMs is sign language recognition. Several techniques for speech recognition are adapted because sign language and continuous speech share many common characteristics. For instance, the signals are generated from the observations of hand movements (position, shape, orientation of the hands etc) over time, just like speech. Silences in both speech and sign-language are relatively easy to detect. Language modeling based on the words as a unit can be applied to the recognition of both problems.

Starner and Pentland [Sta] developed a system to recognize forty American Sign Language gestures using HMMs. They defined features of sign language gestures and designed HMMs that consisted of a probability distribution of the hand features over a set of states. The system computes the probability of input gesture data in each HMMs and picks the model which has the highest probability of producing the input gesture data.

Standard HMMs have been applied to various gesture recognition systems. Pentland and Liu [PL99] used HMMs to model the state transitions among a set of dynamic gesture models. Bregler [Bre97] used HMMs for not only modeling the low-level dynamics, but also the semantics in some gestures. Nam and Wohn [NW96] present a HMMs-based method to recognize gestures using not only hand movement, but also hand postures and palm orientations. Chambers [CVWB02] developed a hierarchical HMMs to represent gesture by composing a set of its sub-gestures. Campbell and Becker [CB96] developed a gesture model by changing the topology structure of the HMMs depending on the gesture complexity.

There are also many variations of HMMs. For instance, Yang et al. [YXC94] modeled the gesture by employing a multi-dimensional HMMs, which contains more than one observation symbol at each state. Multi-path gestures are modeled to increase the recognition rate by integrating multiple modality. Ghahramani and Jordan [GJ95] introduced the factorial HMM (FHMM) to generalize the HMMs which integrates two or more streams of input data. Wilson and Bobick [WF98] introduced special HMMs to model the parameterized gestures. They extended the standard HMMs to quantify the gesture as an output as well as to classify a gesture. They explored the use of a parametric variation in the output probabilities of the HMM to handle parameterized movements such as a size gesture or a point gesture.

Even though HMMs are used in gesture recognition, it is well known that HMMs are limited to interpret multiple observations. Therefore, Bayesian networks [HGC94] have been used in time series modeling as a name of dynamic Bayesian networks (DBNs). Hereafter, it is tempting to fully explore the roles and benefits of Bayesian networks in interpreting gestures. The main reason to use DBNs is that it is statistically advantageous to combine multiple observations from the same source. Brand et al. [BOP97] developed the coupled HMMs

(CHMMs) as a special type of DBNs. CHMMs allow the backbone nodes to interact each other keeping their own observations. In CHMMs, each process can have different state structures and degrees of influence on each other. In particular, CHMMs were used for applications requiring sensor fusion across modalities.

2.5 Gesture-based Inputs

This section summarizes gesture-based inputs previously introduced in various applications. Since there is no standard scheme to categorize the gesture-based inputs, we define four input types based on the four aspects of gesture proposed by Hummels and Stappers [HS98a]: *spatial inputs* using the locations where a gesture occurs, *manipulative inputs* using the positional and rotational path which a gesture takes, *symbolic inputs* using the meaning that a gesture represents, and *affective* using the feeling or the emotional quality that a gesture contains.

In addition, we divide gesture-based inputs into two groups: *direct* and *perceptual*. For direct inputs, the sensor data is usually mapped to the parameters of relevant functions. The manipulative inputs can be a typical example of the direct inputs. On the other hand, perceptual inputs require additional processing to find out the meaning of the sensor data so that these inputs could be less reliable and require more computational time than the direct inputs. Symbolic inputs can be regarded as a perceptual inputs.

2.5.1 Spatial Gesture-based Inputs

Among the different gesture styles described in Section 2.1.2, deictic gestures are particularly important for spatial inputs requiring information about the position and spatial location of the user when the gesture is performed.

Spatial inputs have been used in different applications. Bolt presented the *Put that there* [Bol80] where users can interact with an object on a large screen display through pointing. Similarly, virtual reality systems often enable users to point and select the virtual objects using gestures.

Recently, spatial inputs have been widely applied to applications for smart environments. With the advance of sensor and network technologies, our environments are designed and programmed to understand gestures automatically to a certain degree. These smart environments understand the user's intention, and provide a set of programmed functions such as turning on/off lights and controlling temperature.

In this smart home scenario, spatial information plays an important role in understanding the gesture correctly. Wilson [WS03] presented a gesture interface where users can interact with the smart environment using gestures (selecting and controlling appliances through pointing, turning it on/off and moving the volume up/down).

Spatial inputs are also employed to facilitate the use of context in HCI. "Context is the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user" [CK00]. Dey et al. [DHB+04] presented *CAPpella* (context-aware prototyping environment) designed with the concept of programming by demonstration. In this system, end-users can program desired context-aware behaviors (situation and associated action) by demonstrating them to the system and by annotating the relevant portions of the demonstration.

Jih el al. [JHT06] developed a context-aware elderly care system for smart environments. The system interacts with the elder through a wide variety of appliances for data gathering and information presentation. The system tracks the location and specific activities of the elder through sensors, such as pressure-sensitive floors, cameras, bio-sensors, and smart furniture. The status of the elder is monitored and used to provide appropriate system actions. For example, when sensing that the elder has fallen asleep, the system switches the telephone into voice mail mode, and informs and plays back any incoming messages when the elder awakens.

2.5.2 Symbolic Gesture-based Inputs

The main purpose of symbolic gesture-based inputs is to convey, to a computer, the user's intentions as a set of executable commands. Usually, semaphoric gestures are widely adapted for this type of input. Applications are equipped with methods to interpret a symbolic information imposed in individual gestures. The typical example might be sign language interpretation which does not rely on other input modalities like speech for interpretation of the user intention [SP95, SP].

Eye-free interaction uses symbolic gesture-based inputs which aim to keep our visual attention on other different tasks during gestural interaction. For instance, users can focus on navigating the physical environment without paying attention to the GUI of the application. These days, eye-free interaction has been applied to the mobile computing area [BLB+03]. The advancements in mobile technologies enable personal mobile devices to be smaller in size and allow their CPU memory to be more powerful. The main idea is to allow users to perform symbolic gesture-based inputs to control small personal devices such as PDAs, smart phones, iPods, and Palms.

2.5.3 Manipulative Gesture-based Inputs

Manipulative inputs are closely connected to the selection and manipulation of virtual objects. Manipulations can include two-dimensional (2D) movement inputs along the XY axis and can also be extended to the 3D, including another axis (Z). The input dimension is usually defined with the types of display platforms.

2D manipulative inputs are used for table top displays. In particular, table top displays enable collaboration between multiple users [VB05]. The *DiamondTouch* table [DL01] was designed to uniquely identify each user by electrically coupling users to the table so that multiple users (up to four) can control the system at the same time. Forsberg et al. [FLZ98] presented *Ergodesk*, a back projected drafting table that supports users making an input with a single pen.

Ullmer and Ishii's [UI97] *metaDesk* prototype demonstrated tangible user interface techniques. A variety of physical objects are integrated with multiple displays and projected onto tabletops. *SmartSkin* [Rek02] is another technology that can enable tabletop sensing. Unlike the *DiamondTouch*, it does not distinguish input by user. However, it offers a more complete 2D image of surface contacts, resulting in the unambiguous detection of multiple hand input points and shapes.

3D manipulative inputs are closely related to 3D spatial gestures because of the 3D nature observed in 3D space. 3D manipulation has been extensively studied in 3D graphics applications. For instance, Bowman et al. [BKLP01] presented a set of tasks for 3D manipulative inputs such as navigation, modeling, and menu selections. Mine et al. [MBS97] used 3D inputs to manipulate 3D objects in immersive Virtual Environment (VE) [MBS97] with a set of novel interaction techniques. They analyzed the importance of using proprioception to move objects in 3D space in order to compensate for the lack of real haptic contact with objects. Berry [Ber98] explored the use of hand gestures not only for navigating virtual space but also for moving virtual objects as an interactive device.

2.5.4 Affective Gesture-based Inputs

Compared to other gestural input types, affective inputs are rather new in the HCI domain and it is very challenging to understand the affective information from human gestures. However, there are several promising approaches that use affective gestures such as bio-mechanics, psychology, and digital art performance.

Picard [Pic97] presented an affective interface and demonstrated its potential in various applications. Ravindra et al. [SOMM06] recognized different types of emotion based on gestural intensity that drives movements of the human body, and used the recognized type of the emotion to control the difficulty level of a computer game. Berthouze et al. [BFH$^+$03] also interpreted the affective state of a user based on the captured visual signals from a certain body posture. Sundstrom et al. [SSH05] built a mobile service system called *eMoto* that sends and receives affective messages addressing a type of emotion.

There are also several approaches for sensing emotion using deductive techniques such as measuring the heart rate, the skin resistance, the blood pressure and the muscle activity [PH97]. Facial expression was also used to capture affection for human computer interaction [CBF05].

2.6 Application Areas

The use of gesture is growing interest in many application areas. For example, virtual and augmented reality, tele-robotics, and wearable and ubiquitous computing all explore the use of human gesture in new computational environments. This section provides a higher-level understanding of using gesture in applications, and provides an overview of the practical usage of gesture. Considering the characteristics of 3D spatial gestures, we categorized application areas into four groups: *computer graphics*, *game control*, *motion training*, and *digital art performance*.

2.6.1 Computer Graphics Applications

The input technologies for computer graphics applications have been evolved along with the development of display technologies. A large display wall encourages hand use instead of a computer mouse on the desktop. Bae et al. [BKKK04a] presented a unique input device which supports automotive design on a large display screen with a set of unique interaction techniques. Using the graspable input device with a transparent groove, designers use rich gestures for the manipulation of virtual curves on the large screen surface. Grossman et al. [GBK+01] presented a modeling interface where users can manipulate virtual objects in 3D space using a electromagnetic input device that provides 6-DOF information.

While computer graphics applications mainly use manipulative gestures, there are also approaches that use symbolic gestures. Nishino et al. [NFU99] used the pre-defined set of symbolic hand gestures to create complex shapes for 3D modeling. Hummels and Stappers [HS98b] introduced a gesture-based modeling interface that employs designer's perceptual-motor skill in the expressive and creative design process. Several gestures were designed to provide dynamic interactive sketching in the early stages of product design.

2.6.2 Game Control Applications

These days, the use of gesture in game control is extensively studied and discussed as an active application area [PKE+06]. The major tasks involve control of vehicle movement, navigation of a game environment, and control of avatars in a virtual world.

Various researchers have explored the use of the human body for the purpose of game control. ALIVE [MDBP96] is a full-body recognition interface. A user can interact with autonomous agents in a virtual environment by varying gesture and body position. The user stands in front of a large screen projection panel on which a virtual environment is displayed. Computer vision techniques are used to recognize the users body position and their gestures such as pointing. Cameras are also used by *Magic Mirror* to enable users to see themselves within the virtual

world. The system captures a scene of the user via video cameras, and projects the re-created user image onto the screen.

Hämäläinen et al. [HIH+05] proposed *Kick Ass Kung-Fu*, a martial arts game installation where the player fights virtual enemies with kicks and punches as well as acrobatic moves such as cartwheels. The video image of the user is embedded inside 3D graphics using cameras and computer vision techniques.

Rusdorf et al. [RB05] developed a VR table tennis application which processed fast motions used in table tennis. For high speed interaction, they minimized the latencies by developing a prediction method and analyzing the issue of synchronization between the player's movements and the visual output on the projection. Mueller et al. [MA05] developed a computational environment where users can play soccer with others located in remote places. Using cameras, the system captures the ball movements and shows them to people who are miles apart so that people can play a physically exhausting soccer game together.

There are also several commercial gesture input devices for game control. Playstation2 has introduced *EyesToy*, a camera that tracks hand movements for interactive game control. *Wii* was developed to enable gestural input in commercial game consoles. Nintendo's game console, *Revolution*, takes gesture as a main form of input and enables various gesture-based inputs (aiming and throwing to steer the virtual game characters).

2.6.3 Motion Training Applications

Computers have been recently used to analyze human motions and to help athletes improve their movements. While simpler gestures are preferred for command inputs, motion training applications should be able to process more complex 3D spatial gestures.

Several motion training systems were proposed using different sensors and focusing on different motions. Davis [DB98] developed a vision-based motion training system, called *Virtual PAT* (Personal Aerobics Trainer). They used IR light sources to capture a silhouette of the user and then recognized six stretching and aerobic movements. Moreover, the system provides manually pre-recorded instructive videos and audio feedback so that users can create and personalize an aerobics session.

Becker [Bec97] developed a Tai Chi training system. He tracked the user's head and hand movements with a pair of stereo cameras and then recognized the different types of Tai Chi gestures using HMMs. Chua et al. [CDSC03] also developed a Tai Chi training system. They used a light-weight HMD display and optical motion capture device to build a wireless VR based system. The trainees' motions are evaluated based on skeleton matching to measure how they mimic avatar motions.

Yang [Yan99] developed a dancing system called *Just Follow Me* (JMF) where users can learn a set of dancing motions following the avatars. The task of the student is to follow the ghost version of the teacher as closely as possible in CAVE.

An optical motion capture system was used for tracking motions. Evaluation is accomplished by re-targeting students' motion to the pre-generated avatar templates based on the methods proposed in Baek et al. [BLK03].

Takahata [TSST04] designed a karate training environment. A multimodal room was equipped with cameras, microphones, video displays and loud speakers, wearable devices using a sensor and sound generator. Even though they did not explore a profound method for gesture recognition, they demonstrated how gesture performance can be improved using sound feedback. For this purpose, they provide audio feedback generated by converting the captured gesture signals from accelerometers to sounds.

2.6.4 Digital Art Performance Applications

There are also various applications using gesture in digital art performance [BLSM04]. The main purpose is to use gesture for artistic control of the computer , and encourage users to express their artistic concepts using perceptual-motor skills. The role of applications is to quantify the quality of the performed gestures and respond to the user by providing various audio and visual outputs. This can be accomplished by capturing gestures using various sensors and mapping the acquired gesture data to a certain variable that controls the outputs.

To better understand, we can make an example of a digital art performance application in a dancing scenario [BGK$^+$05]. A dancer creates a digital sound or visual image using their dance movement. The dancer constantly reacts to various resources (dancing movements, tensions on the body, and audio and visual outputs from the system). Since a dancer usually performs enormous gestural expressions, she potentially controls the characteristics of the sound (like pitch and duration) and the visual images (like color and shapes).

Similar to the dancing example, various applications have been proposed to create artistic forms of interaction using gestures. One pioneering work is *Videoplace* [Kru91]. The system was developed around 1970 as a gesture interface using cameras. Live video frames of the user are processed to find the 2D silhouette of the user by subtracting the background. Many gestures used in the everyday world have been implemented as an input. For instance, users can select a menu by touching their index finger to the menu item and manipulating the object by using the finger and thumb from both hands.

Theramin could be a very early example of using gesture to generate electronic sound [Gli00]. Theramin is an electrical musical instrument developed around 1920. Two proximity sensor bars (one for vertical and the other for horizontal) respond to the hand position and its movement. When the hand is moving around the bars, a sound is generated based on the distance between the hands and the sensors.

There are also several other gesture-based musical performance applications. While the gesture recognition techniques were quite specific to the application at

hand, they present a great potential for future research areas that connect gestures to digital art performance.

Sawada and Hashimoto [SH97] presented a system that controls a Musical Instrument Digital Interface (MIDI) instrument using gesture. The system uses accelerometers for gesture acquisition and recognizes ten gestures based on a simple feature set. The *Brain Opera* [Par99] was developed in the MIT Media Laboratory as a large-scale interactive musical exhibit. The *Expressive Footwear* project [PHBT00] was developed with dynamic sensors (gyroscopes, accelerometers, magnetic compass). Sensors are mounted on the side of a dance shoe and capture multi-modal information describing dancer's movements.

Chapter 3

Gesture Acquisition

This chapter presents 3D spatial gesture acquisition system which consists of hardware and software components. The key idea is to use both visual and body sensors to get optimal gesture features for robust gesture recognition and support various requests of target applications such as 3D object manipulation and scene navigation.

Section 3.1 shows a brief overview of our acquisition framework. Then, two sensor technologies are presented: body sensor (Section 3.2) and visual sensor (Section 3.3). Each section describes the characteristics of the sensors for the acquisition of 3D spatial gestures. Section 3.4 introduces our gesture input device which is designed to support easy integration of visual and body sensors.

3.1 Overview

Figure 3.1 shows an overview of the gesture acquisition system. When a user performs gestures, the system acquires the gestures through body sensors and visual sensors. Different body sensors are integrated into a gesture input device that the user attaches to the body or holds in the hand. The device is equipped with a micro-controller for collecting sensor data and Bluetooth wireless technology for transmitting the measurements to a computer.

The current setup uses two machines: one for the acquisition system and one for the application. Necessary data is transmitted via the network between two computers. The acquisition receives data from the input device via Bluetooth and from video cameras via Firewire ports. We use the Java Communications API package to develop a module to receive the transmitted body sensor data. The acquisition module for visual sensors tracks the positions of LEDs on the captured image frames and computes 3D positions.

The acquisition machine runs small functional units called *filters* that process corresponding sensor data in a specific way (converting it to another data type). Different filters are implemented separately with a modular design concept. Java

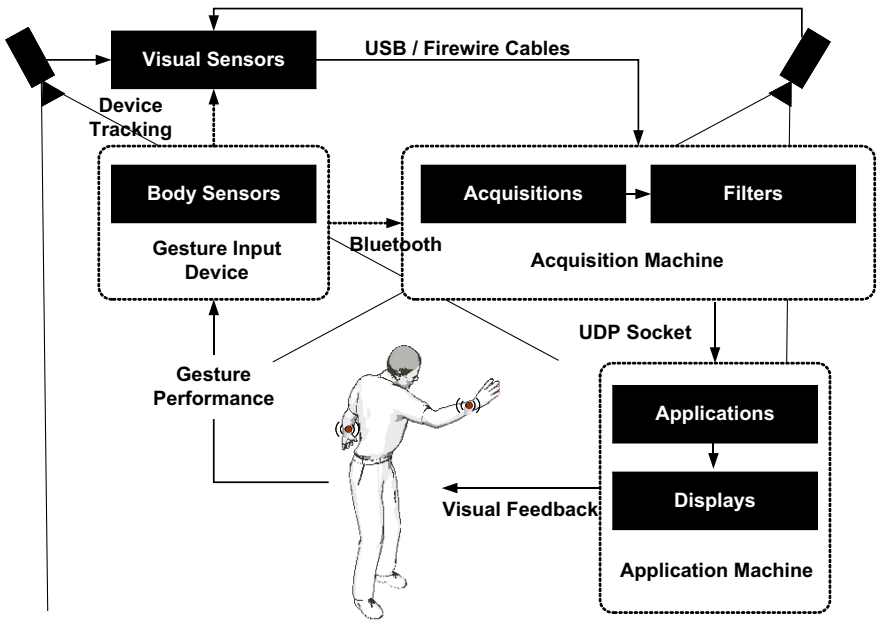


Figure 3.1: Overview of the hardware and software framework for gesture acquisition. The performance of gestures are acquired by both visual and body sensors in acquisition machine. Processed gesture features are transferred to the application via a socket.

multi-threading techniques are used to implement filters as an independent thread and divide the resources optimally between the different threads.

Finally, the processed sensor data is transferred to the machine for application and is used to fit the needs of the target applications. The application machine is programmed to provide visual feedbacks to the user so that he/she can check whether the gesture is processed correctly. The acquisition and applications can be run on a single machine for a simpler setup.

3.2 Body Sensors

The body sensors' main role is to measure the subtle movements of the human body that can be difficult to capture using visual sensors. Body sensors need to be attached to the body or held in the hand and the form factor of sensors should be designed to minimize the restriction of the user's movements. The advanced Micro-Electro-Mechanical Systems (MEMS) make the size of sensors smaller and provides the potential to build an input device for acquiring 3D spatial gestures.

3.2.1 Accelerometers as a Body Sensor

In the framework, we explore the use of accelerometers for capturing 3D spatial gestures. Accelerometers are widely used for detecting tilt, movement, and vibration and they provide two advantages for gesture interfaces. First, the small size of accelerometers allows for embedment in small objects such as a wrist watch. Second, accelerometers provide rich information about 3D spatial gestures (tilt angles and inertia). Other sensors, gyroscopes and electromagnetic sensors could also be used to measure tilt angles. However, while electromagnetic sensors provide rather accurate measurements, they are not completely wireless and require additional infrastructure to generate a magnetic field. Gyroscopes can also sense rotation around the three axes. However, it is common for them to include a temperature sensor and a voltage reference to condition the signal. The manufacturing process is also more complicated causing the package size and price to increase.

Sensing Mechanism

The sensing mechanism of accelerometers can be explained using a basic physical principle of mass-spring systems where the mass is fixed with a torsion bar that has a specific spring constant in the accelerometer. Figure 3.2 illustrates this basic concept of the accelerometer along a specific axis. The ball represents the mass and the surrounding spring acts as a capacitor. A ball that is attached to two springs on opposite sides and placed in a tube. When the sensor is accelerated, the ball moves in the opposite direction of the acceleration. This results in a change of the distance between the ball and the spring. This displacement of the mass yields a change in the capacity of the sensor.

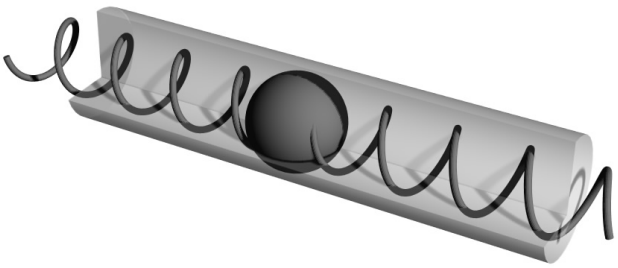


Figure 3.2: A conceptual diagram of an accelerometer sensing mechanism. A ball is located in the middle of a tube and attached to two springs. The length of each spring is affected by the movements of the ball and measured as a acceleration value.

There are two types of movement information that can be measured by accelerometers: *shaking* and *tilting*. When the accelerometer is shaken, the value of acceleration is proportional to the scale and the speed of shaking. When the

accelerometer is tilted, the mass is moved to a certain direction because of the gravity and this causes the increase or decrease of the value of acceleration. The acceleration for shaking and for tilting are often called *dynamic acceleration* and *static acceleration* respectively.

Measuring Postures and Gestures using Accelerometers

We use two types of acceleration for measuring postures and gestures as explained in the previous section. In our framework, postures are static body configurations that are characterized with certain positional and rotational information. For the rotational information of a posture, we use static acceleration. Figure 3.3 shows two hand postures with certain pitch and roll angles that can be measured by attaching an accelerometer on the wrist.

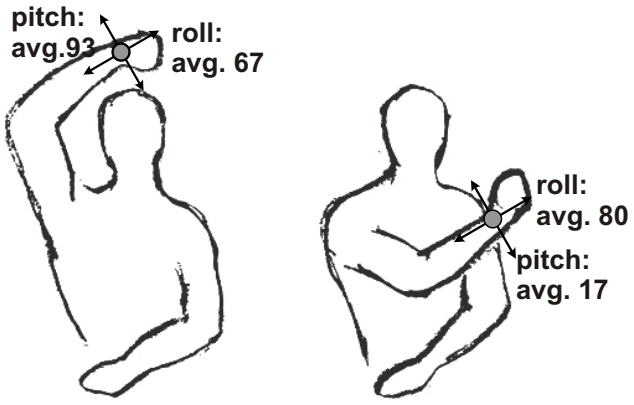


Figure 3.3: Different pitch and roll values for postures. When the accelerometer body sensor is attached on the wrist, postures are characterized with certain pitch and roll values.

On the other hand, gestures are dynamic movements that are typically performed by connecting two postures - a user starts with one posture and ends with another posture. Even though the start and end postures are same, the in-between gestures can vary depending on velocity and inertia. Our framework uses dynamic acceleration to observe the velocity and inertia of the gestures. Figure 3.4 shows how four examples of acceleration signals are generated when gestures are performed with two postures (from the left posture to the right posture shown in Figure 3.3). While static accelerations for the start and end postures are approximately the same between the four gesture performances, the in-between gesture data (dynamic accelerations) are all different with different changes of power and velocity over time.

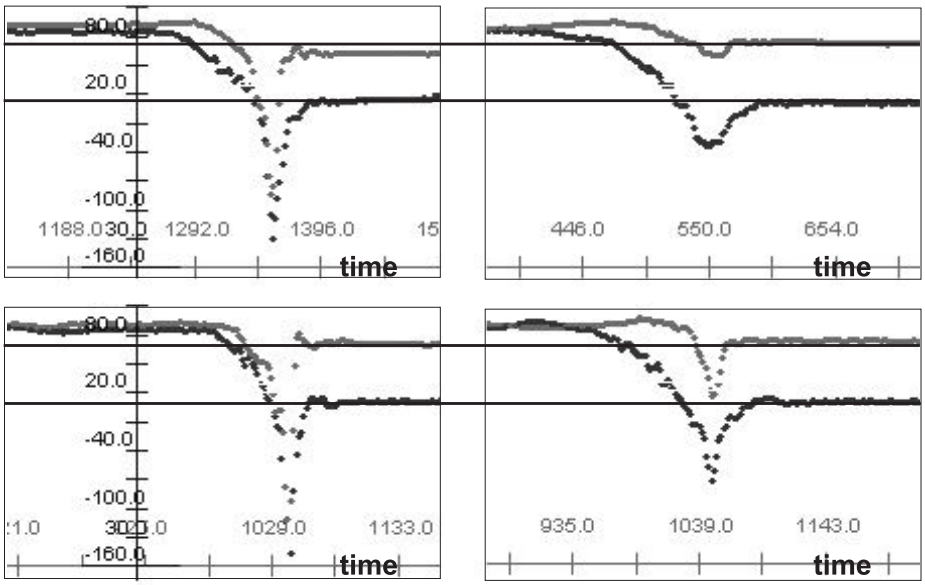


Figure 3.4: Examples of two-axes accelerometer signals when 3D spatial gestures are performed. 3D spatial gestures are performed by connecting one posture to another posture with different speeds and powers. Each gesture starts and ends with approximately the same average roll and pitch values and the in-between parts vary.

3.2.2 Computing 3D Rotation

We use body sensors to compute 3D rotation which is usually represented in the Euler coordinate system with three different angular values: *yaw*, *pitch*, and *roll*. Our framework uses the computed 3D rotation to describe postures and also to rotate a virtual object in 3D space.

Figure 3.5 shows the representation of 3D rotation using Euler coordinate system when a body sensor is attached to the wrist like a wristwatch. The roll axis of the sensor is parallel to the forearm, and the pitch axis is horizontal and perpendicular to the roll axis. Yaw values point out the upright position of the hand. This section describes how 3D rotation can be robustly measured combining accelerometers and magnetometers.

Again, pitch and roll can be measured using accelerometers. When the sensor is rotated, the change of the gravity force is applied to the accelerometer and the static acceleration is observed. For more accurate measurement, we consider that the sensing axis becomes less sensitive and the noise increases as the accelerometer is tilted beyond a 45 degree angle in either axis. This noise can be critical when we use pitch and roll to rotate a virtual object in 3D space. Our framework uses a technique that improves the tilt sensing using tri-axis accelerometers. The main idea is to compensate the noise of one axis value with another axis. In theory, the basic three tilt angles can be computed from the tri-axis accelerometer outputs as follows:

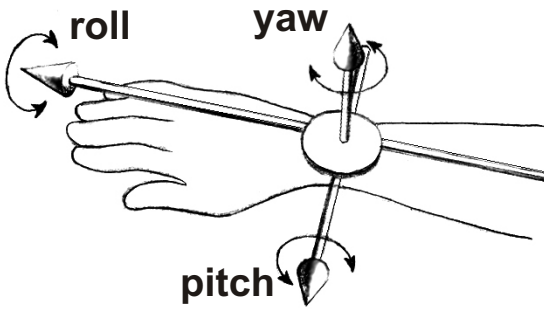


Figure 3.5: The representation of 3D rotation using the Euler coordinate system (roll, pitch, and yaw). Pitch and roll are measured by an accelerometer, and roll is measured by combining an accelerometer and a magnetometer.

$$\begin{aligned} \phi &= \arcsin(a_x) \\ \rho &= \arcsin(a_y) \\ \theta &= \arccos(a_z) \end{aligned} \tag{3.1}$$

To improve sensing precision and accuracy, another axis (Z) is combined with both of the tilting axes. Then, both the pitch (ϕ) and roll (ρ) is computed using the outputs of all three axes following:

$$\begin{aligned} \phi &= \arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \\ \rho &= \arctan\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \end{aligned} \tag{3.2}$$

To sense yaw, we use a magnetometer that can sense the earth's magnetic field. Typically two or three anisotropic magnetic sensors are used as a compass in orthogonal angles (perpendicular to each other) to measure the earth's magnetic field into Cartesian coordinates X, Y, and Z. The direction is computed using X and Y sensor outputs according to:

$$\alpha = \arctan\left(\frac{Y}{X}\right) \tag{3.3}$$

Since the arc-tangent function repeats itself every 180 degrees, we need to determine where the device is pointed in the 0 to 360 degree heading circle.

To get the optimum compass accuracy, the magnetometer should be operated in a flat position without any tilt. However, a gesture input device should be able to support users performing gestures in various positions and orientations. To minimize this error, we integrated the accelerometer data with the magnetometer data for retainining a more accurate yaw value in the presence of pitch or roll

angles. The X and Y compass values are computed with the pitch and roll angles following:

$$\begin{aligned} X_f &= X\cos(\phi) + Y\sin(\rho)\sin(\phi) - Z\cos(\rho)\sin(\phi) \\ Y_f &= Y\cos(\rho) + Z\sin(\phi) \end{aligned} \tag{3.4}$$

The final tilt is computed from the arc-tangent heading equation using the newly computed (or flattened) X_f and Y_f compass values.

3.3 Visual Sensors

Our framework uses color cameras as a visual sensor to capture color scene images in a certain range at a certain frame rate (e.g. 30 Hz). The acquired images are processed to produce visual features that are eventually used to interpret human gestures. One of the important advantages of visual sensors over body sensors is that users do not have to wear body sensing devices.

However, the performance of visual sensors relies heavily on the accurate tracking of objects in the camera images. Even with the advanced vision technologies, this task is still challenging under varying illumination and background conditions. For more robust tracking, several approaches are proposed using passive or active markers.

Occlusion also prevents continuous tracking of the desired body part from a single view. Multiple cameras are used to partly solve this occlusion problem by taking images from different view angles. In this context, multiple cameras are utilized to reconstruct a full 3D user body which are useful in analyzing human motions [GWN+03a]. Moreover, multiple cameras enlarge the interaction area, so that if the tracked object disappears from the field of view in one camera, it can be viewed in another camera.

3.3.1 Computing 3D Position

Our framework uses a pair of cameras to track the 3D positions of the object-of-interests (input devices or hands). The tracked 3D positions are used to manipulate objects in 3D space and to generate visual features for gesture recognition. Figure 3.6 shows a prototype setup that uses a pair of stereo cameras. Two cameras capture the user's movements and the acquisition system processes the data and tracks the 2D and 3D positions of the hands or input devices.

In our setup, one projector is connected to the machine for the application system. The user interacts with the display feedbacks in the interaction space between the cameras and the projection screen. Again, different camera configurations could be experimented to enlarge the interaction area and reduce the occlusion problems depending on the target application.

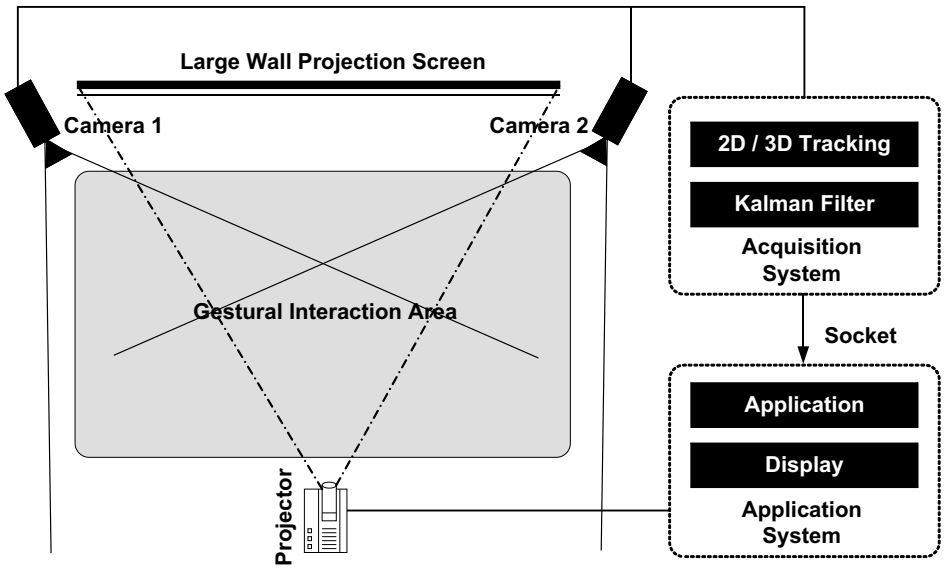


Figure 3.6: A prototype system setup using a pair of stereo cameras. The gestural interaction area can be refined with a pair of cameras, a projector, and a large wall projection screen. The acquisition system processes the acquired image frames to track 2D and 3D positions of the device with a Kalman filter. The positional data is then transferred to an application system that provides a display to the user via a projector.

As shown in Figure 3.7, the cameras are aligned in parallel and installed from the top facing down toward the user. Figure 3.7 shows the actual image views from the cameras and gives an impression of the captured image quality.

As mentioned earlier, tracking multiple objects using cameras is a challenging task because lighting and background conditions can change over time. In our earlier tests, the color tracking highly depends on lighting and color, and the approximately tracked position is not precise enough to handle small scale gestures. We tested IR light sources, but they do not convey color information which is required for tracking multiple points.

Since real-time gesture processing is our goal, several compromises were made. We used bright color LEDs for fast, precise and robust tracking. Their brightness provides relatively robust tracking results in indoor environments. Moreover, using different color LEDs allows us to track multiple points (multiple users or multiple body parts of the same user) at the same time.

We developed a simple vision tracking algorithm to find the pixel positions within a certain color and brightness range. To find each color LED, the algorithm scans the image until it finds a pixel of the appropriate color. In our method, the

Figure 3.7: The prototype camera installation and views from the cameras.(Top) A user is performing a 3D spatial gesture in the prototype system setup that consists of a pair of cameras and a projection wall in front of the user. The camera is pointing down to the user. (Bottom) The images from the cameras.

resolution of 3D position varies from 0.3*cm* to 0.5*cm* depending on the location of the LED on the captured image and the color of the LEDs. In our experiments during development, the focus area of the cameras gives the better results than the surrounding areas. A white color LED also provides more accurate tracking results than other colors such as red, blue, and green.

The *Kalman* filter [MS83] is used to improve the object tracking from frame to frame against different movement conditions and occlusions. The *Kalman* filter models the dynamic properties of the tracked object as well as the uncertainties of both the dynamic model and the low level measurements. Consequently, the output of the filter is a probability distribution representing both the knowledge and uncertainty about the state of the object. The solution is recursive in that each updated estimate of the state is computed from the previous estimate and the new input data, so only the previous estimate requires storage.

The framework uses conventional triangulation from a pair of calibrated cameras. The camera pair is calibrated using standard algorithms from the Intel Open Computer Vision Library [Ope]. The purpose of camera calibration is to deter-

mine the intrinsic and the extrinsic parameters of cameras.

The intrinsic parameters (focal length, image coordinate origin) of cameras are pre-calculated by taking a set of pictures of a printed checkerboard shown in Figure 3.8. The extrinsic parameters are the position and orientation of the cameras and they are calibrated using the checker board images. The calibration of extrinsic parameters needs to be done only once as long as the cameras' position and orientation are not changed. Once these parameters are known, three-dimensional information can be inferred from two-dimensional images and vice versa. Calibration algorithms can be found in Zhang [Zha99].

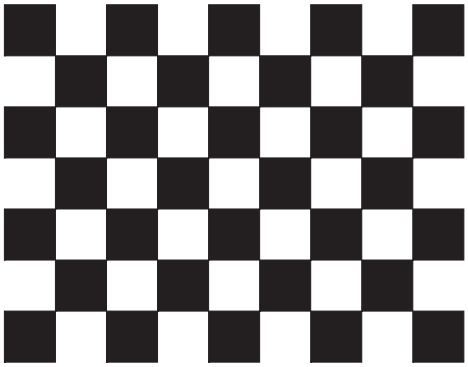


Figure 3.8: Checker board pattern used in camera calibration. This pattern is printed on a paper and attached to a flat board.

3.3.2 Invariant Visual Features

In our context, the selection of gesture features should be such that the resulting gesture model is invariant with regard to translations and rotations. While the acceleration fulfills these properties, raw positional information of visual features should be further processed to be invariant.

There are several possible ways to compute invariant features from position data including angular velocity, curvature, and velocity. The different features trade off against each other. For instance, consider a person at a fixed location and orientation with respect to the camera setup, performing a hand gesture in a perfect circle. Curvature and speed (ρ, ds) will be completely invariant to rotations and translations but will be constant and thus exhibit no meaningful gesture information. Now consider the velocity of the hands (dx, dy, dz) as a training and testing feature. This feature is shift invariant but not rotation invariant. Particularly when the hand is almost stationary it tends to generate high noise.

The main problem of these features is the differential nature of these features including computations of the first and second derivatives. This problem is present in the curvature features because curvature is a function of the second derivative and thus inherently more noisy than velocity features. At the other extreme, absolute coordinates (x, y, z) can easily distinguish the top, bottom, and sides of the

circle, but they require it to be performed each time at the same location. Overall, all of these features have limited reliability and do only work for larger and longer gestures.

This leaves us with the relative Cartesian positions (*rx*, *ry*, *rz*) according to a certain fixed position. The relative position can be reliably derived from the visual sensor and characterize a 3D spatial gesture. In addition, while a gesture with absolute coordinates is confined to a particular location in an environment, relative coordinates enable us to use a gesture at any location in an environment.

In vision research, the relative position of the hand with respect to the head has been widely used for measuring dynamics of hand gesture because visual tracking of hand and head can be done using the unique appearance of the human skin in color space. Our approach to use multiple color LEDs allows us to use other potential body parts such as the chest or the shoulders as a suitable stationary origin.

3.4 mWire: 3D Spatial Gesture Input Device

Extensibility is an important feature of our framework in both hardware and software. Since our main goal is to enable users to easily design new gesture interfaces, it is necessary to support them testing different sensors including visual sensors and body sensors. Therefore, users can find the optimal sensors that provide the necessary information for the desired gesture interface.

Our framework supports a novel input device called *mWire* (Figure 3.9), which facilitates easy sensor integration. Users connect various types of body sensors and also multiple color LEDs for vision-based position tracking. The device was designed considering the particular muscle groups used in performing 3D spatial gestures, including the wrist, arm, and hand. Aiming to be a more versatile device, mWire is a wearable input device which users can either wear on the wrist (Figure 3.9-b) like a wrist watch or hold in their hand (Figure 3.9-c). This idea is inspired by commercial hand-held devices like MP3 players or cellular phones. The current version is a prototype that can easily be miniaturized for professional production. For instance, a wristwatch can be built using the same hardware concept.

3.4.1 Device Configuration

Figure 3.10 shows the device configuration of mWire. The physical interface of mWire consists of two separate units: a *sensor unit* for sensing signals from the wrist and a *processor unit* for processing and transmitting sensor data to the host computer. The processor unit is equipped with a Bluetooth transmitter, a micro controller, a 9V battery, and a standard RS232 serial connector. This unit can be attached to any convenient body part using appropriate holders (on the waist

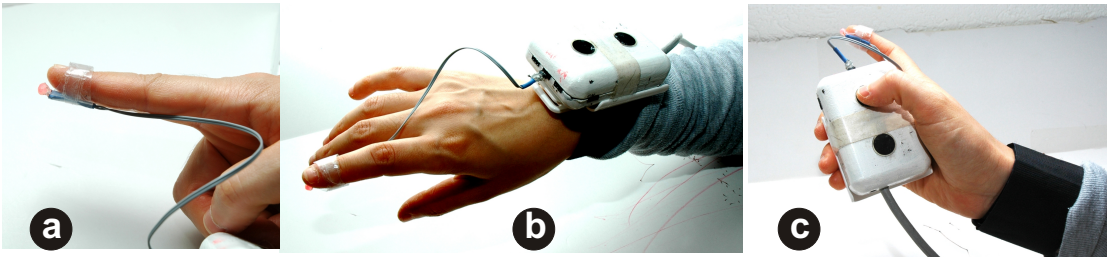


Figure 3.9: A gesture input, mWire which facilitates the use of different body sensors and LEDs. (a) A connected red LED on the index finger with a transparent ring and a wire. (b) The mWire on the wrist (c) The mWire in the user's hand while operating a pressure sensor with another finger.

belt or in pockets). To assure minimum movements of the device, a detachable armband is used to firmly attach into the wrist.

The sensor unit contains one 2D-axis accelerometer as the main body sensor and two pressure sensors attached on the top surface of the case. The accelerometer on the wrist measures subtle acceleration of the hand that are used for the body features invariant to rotations. From these measurements, we can also infer the orientation of the hand and whether the palm is facing back or front, or up and down.

When the device is in the hand, a user can use their thumb to press one of the pressure sensors as shown in Figure 3.9-c. When the device is mounted on the wrist, the user can use the non-dominant hand to use the pressure sensors.

The sensor unit supports two LED connectors. With additional extension wires, different color LEDs can be connected to the device and attached to various body parts such as fingers (Figure 3.9-a), elbows, and shoulders. The sensor unit provides two sensor connectors so that users can easily connect other relevant sensors like bend sensors or digital compasses.

3.4.2 Major Hardware Components

During the design of a gesture input device, the choice of hardware components should be done by analyzing their technical and design characteristics. For instance, the device has to remain small enough for users to handle it with one hand or attach it to their body. The device also needs to be accurate and autonomous so it can work properly for long periods without requiring cabling for power and communication.

Figure 3.11 shows the inside of the mWire's processor unit which contains a micro-controller, a Bluetooth transmitter, a battery, and a sensor unit connection. The micro-controller collects different sensor values. For this, the micro-controller is equipped with different types of ports for measuring sensor data and

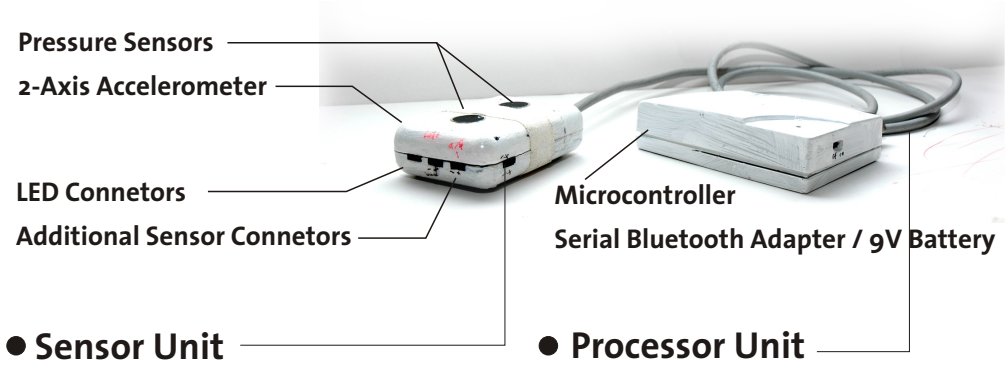


Figure 3.10: Hardware configuration of the mWire. On the left, we see the sensor unit with two pressure sensors, one 2D-axis accelerometer, two LED connectors, and two sensor connectors. On the right is the processor unit with a micro-controller for acquisition, a Bluetooth transmitter, and a 9V battery.

input and output ports for transmitting the measured data. The acquired sensor data is transmitted using a specific protocol to the acquisition machine via the Bluetooth transmitter. In the following sections, we present the three major hardware components of the mWire in detail: *micro-controller*, *bluetooth*, and *sensors*.

Micro-controller

The micro controller we have chosen is the *C-Control* [C-c]. This controller contains necessary electronic components for connecting sensors and programming them, and supports binary switches, frequency, and voltage measurements for sensor inputs. The C-Control requires a 5V power source.

Sensors

mWire is equipped with two types of sensors (accelerometer and pressure) in default. For the accelerometer, we use a dual-axis accelerometer (ADXL311) for measuring shaking and tilting movements. Two pressure sensors are attached on the flat surface of the device. The pressure sensor changes resistance according to the force applied to its active surface so that we can capture the continuous grip gestures applied to the device. Pressure sensors are often called *Force Sensing Resistors* (FSR), and various sizes and shapes are available.

Bluetooth

Bluetooth wirelessly transmits the acquired sensor data from microcontroller to the acquisition machine. As an industrial standard for personal area networks,

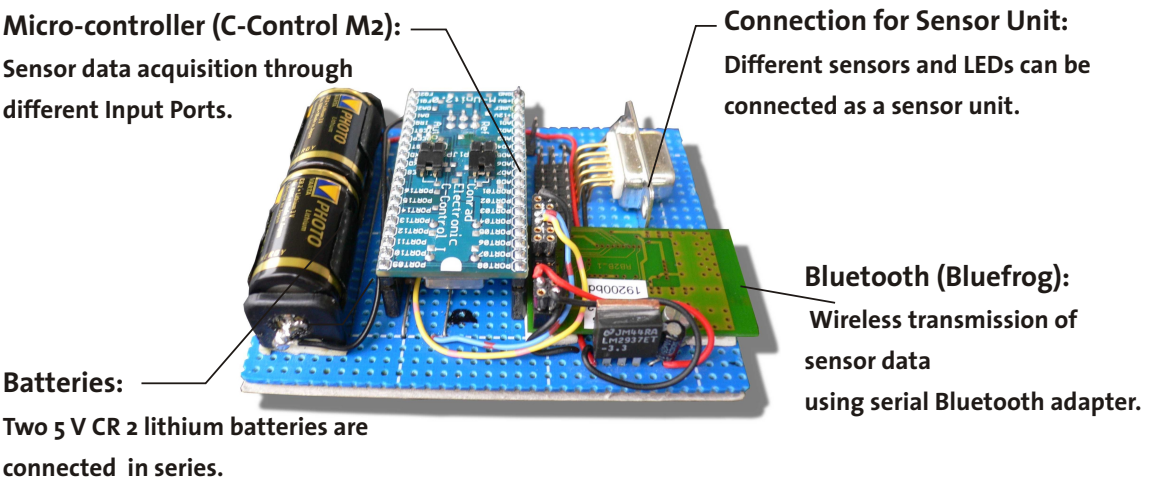


Figure 3.11: The hardware components of the mWire processor unit. The processor unit contains a micro-controller for acquisition and processing, a Bluetooth transmitter for forwarding sensor data, a battery power supply, and a sensor unit connection.

Bluetooth supports short range communication by connecting different devices such as computers, PDAs, and printers.

In our prototype, we selected a Bluefrog adapter that transmits and receives data via FM over an approximate range up to fifty meters indoors with relatively low power consumption for an RF module (8 mA) and data rate (64 kbps). The Bluefrog module supports the Serial Port Profile (SPP). The SPP is used to connect the Bluetooth to the serial port communication of the microcontroler, and it acts as a transparent bridge for serial communication without using external RS-232 serial converter. Therefore, we can easily pre-configure the Bluetooth for a virtual serial port in the computer.

Chapter 4

The Gesture Model

The previous chapter described how 3D spatial gestures are acquired through different sensors and input devices. Now, we explain how the acquired gesture data are modeled to support various tasks of gesture interfaces. Our goal in gesture modeling is not only to recognize the type of gesture, but also to evaluate the motion qualities associated with the underlying 3D spatial gestures. Another important aim of our gesture modeling is robust gesture registration so that users can easily manage the gesture model.

The chapter is organized as follows. First, in Section 4.1 we provide a short overview of the gesture model that consists of a set of software components. Section 4.2 describes the representation scheme of 3D spatial gestures which introduces the concept of motion chunk to store continuous gesture data. We explain a way of segmenting gestures in Section 4.3. Dynamic Time Warping and Hidden Makov Models, two modeling techniques, are described in Section 4.4 and Section 4.5 respectively. Lastly, three major tasks of the proposed gesture model are described: gesture registration (Section 4.6), recognition (Section 4.7), and evaluation (Section 4.8).

4.1 Overview

Figure 4.1 shows an overview of the gesture model. As described in the previous chapter, at the gesture acquisition step, visual and body sensors acquire the gestures, and the acquired sensor data are processed to generate features that describe the salient characteristics of the gesture. The features are forwarded to the gesture model and used for modeling gesture.

Our gesture model contains various subsystems that support the development of a gesture interface. First, the system processes the acquired sensor data following two steps: *representation* and *segmentation*. A central element of our gesture representation is a gesture unit called *motion chunk* into which continuous gesture data is segmented. The output of representation is a set of parameters and features extracted from the raw sensor data in a convenient form for the registration and

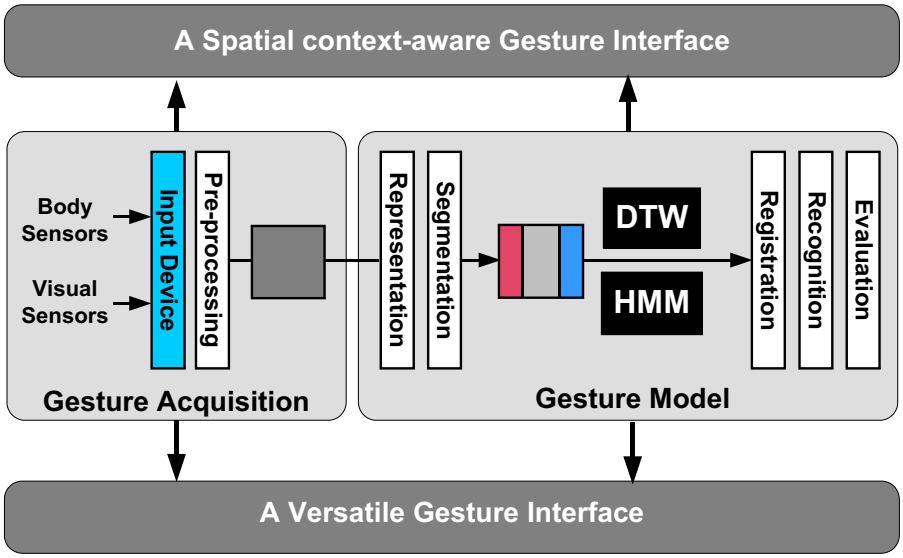


Figure 4.1: The overview of a 3D spatial gesture model. During acquisition, 3D spatial gestures are captured from both body sensors and from visual sensors. The captured data is represented, segmented, and processed using DTW and HMMs for the registration, recognition, and evaluation of 3D spatial gestures. Outputs of gesture acquisition and model are used in different gesture interfaces (a spatial context-aware gesture interface and a versatile gesture interface).

evaluation, as well as the recognition.

The gesture model uses two pattern matching techniques: Dynamic Time Warping (DTW) and Hidden Markov Models (HMMs). Using these two techniques, the gesture model operates three major software components: *registration*, *recognition*, and *evaluation*. Using the registration, users can design their own gestures and add them to the system. The recognition identifies the type of the unknown input gesture. The evaluation measures the quality of the input gesture, and the evaluation results are presented through the gesture interface. The following sections describe each component in detail.

4.2 Gesture Representation

Speech recognition systems usually utilize atomic sets of speech units, called phonemes [RJ93]. Based on the structure of phonemes, the input speech signal is represented and modeled. Similar to the human voice, human motion is sequential in time and we can assume that human motion can be represented with a sequential combination of chunks similar to speech analysis. However, there is no commonly agreed-upon concept to represent gestures in HCI domain. Therefore, most of the existing gesture models have been developed with ad-hoc gesture

units for a particular recognition task. In this section, we introduce motion chunk to process and decompose unstructured human motion.

4.2.1 Motivation

In our framework, we developed motion chunk to represent 3D spatial gestures. Motion chunk was developed based on a series of systematic observations and experiments in modern psychological and linguistic research on gesture. Efron suggested three phases of gesture identification: preparation, stroke, and retraction [Efr41]. The gesture starts with the preparation phase by placing a hand in the initial position, the actual gesture is performed through the stroke phase, and the retraction phase results with the hand in a relaxed position.

Following the results of Efron's study, several approaches were proposed in the psychological and linguistic domain. For instance, Kendon [Ken80] developed his gesticulation theory which investigates the relationship between a gesture and speech. He introduced a concept of gesture phrase based on the three gesture phases of Efron. Kendon introduced a *gesture phrase* which is "the nucleus of movements with definite form and enhanced dynamic qualities preceded by a preparatory movement and succeeded by a rest movement". Gesture phrase consists of three steps: preparation, nucleus, and refraction. The preparation is a resting position or previous gesture. The nucleus is a definite form and poses enhanced dynamic qualities. The refraction is again a resting position for the next gesture.

We extended the gesture phrase idea to generally structure human gestures. To this end, we developed the motion chunk which stores gestural information by segmenting the input signals into subunits. It decomposes the input gestures into a form that reveals relevant gesture information based on the gesticulation theory proposed by Efron [Efr41] and Kendon [Ken80]. Motion chunks enable the physical tension and relaxation of making a gesture to correlate pleasantly with the mental tension and relaxation involved in performing a primitive task in the application. The motion chunk is used as the core representation of our gesture model and serves as a basis for automatic segmentation, registration, recognition, and evaluation.

4.2.2 Definition and Structure

For the motion chunk-based gesture representation, we assume that all input gestures are performed as a set of gesture units: *start posture*, *gesture*, and *end posture*. While static position (also referred to as posture, configuration, or pose) is not technically considered as a gesture, we include it as a motionless part of gestures for the purpose of gesture representation. The start and end postures contain important gesture information in terms of kinematic and bio-mechanical features

which can be useful for the analysis of the transitions from one gesture to the next. The gestures represent the dynamics of the gesture.

Based on this assumption, any continuous gesture stream can be decomposed into two different types of chunks, a *static chunk* and a *dynamic chunk* being equivalent to postures and gestures respectively. For instance, a typical gesture could consist of three types of sub-chunks, a start-static chunk, a dynamic chunk, and an end-static chunk. Figure 4.2 illustrates a sequence of two motion chunks, each of which consists of two static chunks.

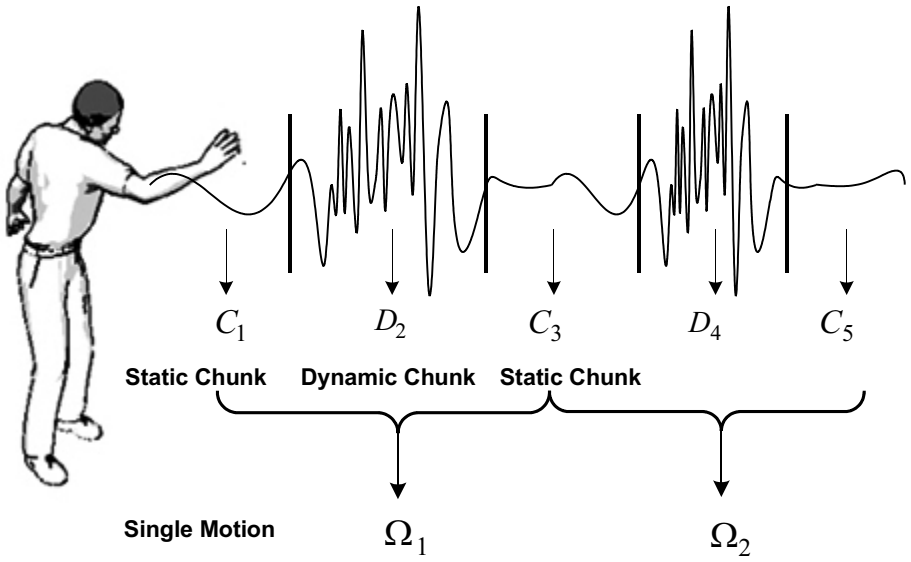


Figure 4.2: The structure of a motion chunk. A 3D gesture is represented as a sequence of motion chunks which consist of two static chunks S and E and one dynamic chunk D.

To recognize an input single motion, each of the n single motions known to the system are assigned a motion type Ω_n. The recognition rule r_n maps the observation sequence on the basis of the start-static chunk C_{i-1}, the end-static chunk C_{i+1}, and the dynamic chunk D_i to a motion index κ of motion type Ω_K, i.e.

$$r_n : [C_{i-1}, D_i, C_{i+1}] \mapsto \kappa \tag{4.1}$$

Furthermore, we can induce several types of motion structure using the motion chunk templates based on the analogy between gesture recognition and speech recognition. There are three types of recognition problems: single motion recognition, recognition of a sequence of motions, and overall motion understanding. A single motion such as punching, blocking, kicking, and striking can be considered as a *phoneme level* and *word level* in speech recognition. They do not involve sequences of other gestures.

Using word level gesture recognition, we can detect sentence level gestures by evaluating transition probabilities between the word level gestures. We use the motion chunk templates to formalize higher-level gesture recognition called *activity level* gestures which are equivalent to a *sentence level* in speech recognition. For example, sparring could be a sentence level motion involving sequences of punching and blocking.

Gesture designers decide what gesture sequences and which level of sequences will correspond to a motion chunk. In the very simplest case, a single discrete gesture can be assigned a motion chunk. However, this approach might be inflexible for some gestures which can be separated into sub-gestures. For example, continuous long gestures require new models to be created and trained every time their sub gesture is changed. Our ultimate goal is to use motion chunk to define sub-gesture units in each model and construct a single gesture from these. Then we can construct a network of motion chunks and have paths through the network indicating the recognized gestures. It reduces the computational complexity and makes it possible to reuse the constructed gesture model in various applications.

4.3 Gesture Segmentation

As described in the previous section, we assume that gestures in our framework are performed based on the motion chunk. The gesture segmentation is relatively straightforward. Three temporal phases of a motion chunk are distinguishable through the general motion sequence: start posture, gesture and end posture which are characterized by rapid change in the body and visual sensor signals. For instance, a single motion chunk starts and ends with a posture so that the start and end points of a motion chunk are easily detected.

The framework uses a simple sliding window technique similar to algorithms in speech recognition. First, we compute a standard deviation of the samples in the window (typically of size 20) which slides along the signal with a sampling rate of 30Hz. The standard deviation σ of a window of the accelerometer signal is given by

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{4.2}$$

where $x_1, ..., x_n$ are the samples in the window and μ is its mean.

A simple moving average filter is applied to the raw sensor data for more accurate segmentation. This filter takes as input a certain number of input frames and provides the average value for each sensor value over the input frames. The number of input frames is the size of the data window for the filter. The calculated average values are then forwarded to the output buffers. When the output is sent, the window is always shifted by one data frame and the next average value is calculated.

Figure 4.3 shows a a simple state machine used for gesture segmentation based on the structure of motion chunk. We assume that a gesture starts with a preceding start posture if the standard deviation is above the starting threshold, and subsequently a gesture ends with a following end posture if the standard deviation is below the ending threshold. To obtain more robust segmentations, we also check the length of the static chunk and the dynamic chunk. As shown in Figure 4.3, the length of the start-static chunk (t_s), the dynamic chunk (t_d), and the end-static chunk (t_e) are checked with a corresponding threshold. For instance, the motionless periods t_s and t_e before and after the gesture must be longer than the respective thresholds l_s and l_e. This eliminates possible spurious static parts in the dynamic chunk. Similarly, the length of dynamic chunk t_d should satisfy $t_d > l_d$.

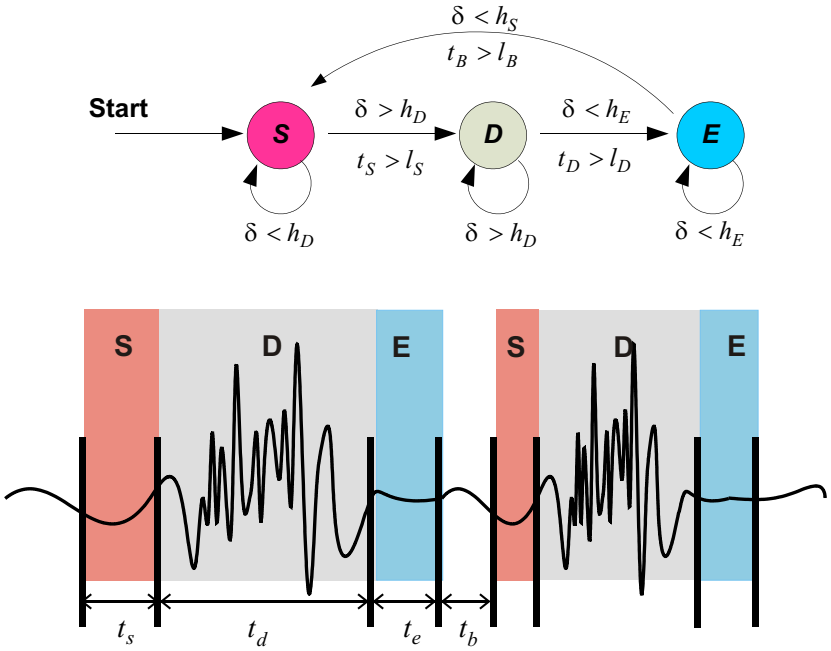


Figure 4.3: (Top) A state machine for gesture segmentation based on a sequence of a motion chunk structure: start-static posture (S), dynamic chunk (D), and end-static chunk (E). (Bottom) The length of each chunk should be within a certain range.

The sensitivity of the segmentation depends on the window size and on the thresholds. These parameters are defined with a simple test and stored to a property file. Alternatively, we compute a second standard deviation over the previously computed standard deviations to segment regularly-vibrated gestures. Figure 4.4 shows the results of segmentation performed on the actual acceleration signal.

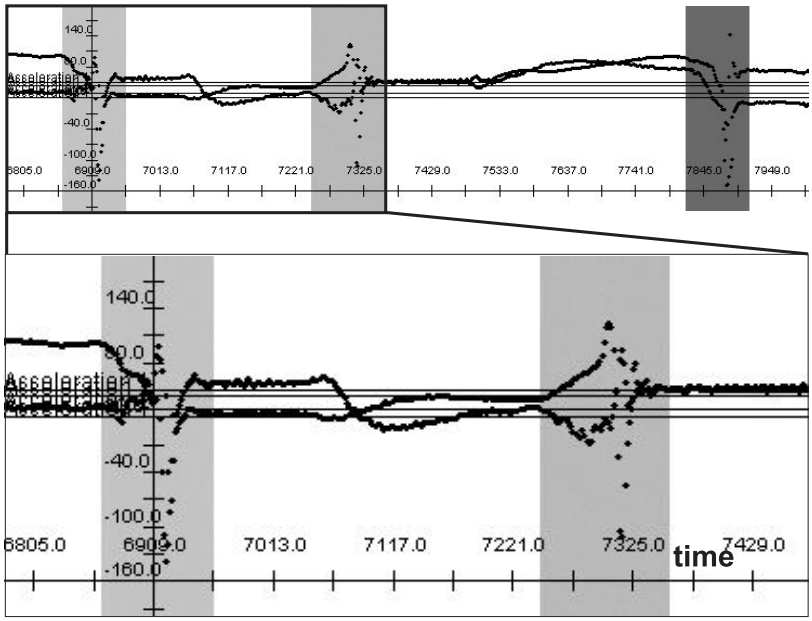


Figure 4.4: An example of gesture segmentation performed on the accelerometer signals. The dynamic chunk is marked with a gray color that shows when the gesture starts and ends. We consider the motionless parts before and after the marked area as start and end static chunks (postures).

4.4 DTW-based Gesture Modeling

Our framework uses DTW for two important functionalities of our gesture model: *gesture evaluation* to discover qualitative difference between two time series and *gesture recognition* to identify the type of input gesture for gesture recognition. DTW has been widely used to measure similarities between time series data. In this section, we present basic fundamentals of DTW (Section 4.4.1) and describe some variations of DTW for our gesture model (Section 4.4.2).

4.4.1 Fundamentals of Dynamic Time Warping

In general, gesture sensor signals are time series and a task in gesture recognition is comparing one sequence with another. Dynamic Time Warping (DTW) is one of the most popular template matching techniques considered as a straightforward method to recognize gestures. It computes the difference between two patterns (*input* and *template*). The major goal of the DTW is to match a given sequence of input values to a stored template. For instance, as one of the earliest approaches, DTW was used to isolated word speech recognition. A prototypical version of each word is stored as a template in the vocabulary, and incoming input speech is compared with each word. The closest match is taken as a result of recognition. Two problems are involved in this recognition process: what form do the templates

take and how are they compared to incoming signals.

DTW has been used to recognize gestures performed with variable speed [Cor01, DP]. The template is a single performance of the gesture and stored as a standard. The simplest form for a template is a sequence of feature vectors. Since gestures can be performed with different speeds, the matching process between template and input sequences needs to compensate for the non-linear nature of the length differences. DTW achieves this goal; it finds an optimal match between two sequences of feature vectors stretching and compressing sections of the sequence. We will describe fundamentals of DTW in this section.

The DTW Grid

Again, the main idea of DTW is to temporally align two sequences X and Y before computing a matching score such as distance. The alignment between an input sequence and a template sequence is done by finding a correct warping path and computing a local distance between these two sequences.

$$\begin{aligned} X &= x_1, x_2, \ldots, x_i, \ldots, x_{T_n} \\ Y &= y_1, y_2, \ldots, y_j, \ldots, y_{T_m} \end{aligned} \tag{4.3}$$

The time indexes (T_n and T_m) need not be identical. To find best alignment, a set of warping paths (ϕ) is computed and the one which has minimum warping cost is selected. A T_n-by-T_m distance matrix is created, and each element (i, j) contains the distance $d(x_i, y_j)$ (e.g. Euclidean distance) between the two points x_i and y_j given by

$$d(q_i, c_j) = (q_i - c_j)^2 \tag{4.4}$$

A warping path (ϕ) has to start and finish at the diagonally opposite corner cells $(1,1)$ and (m,n) of the distance matrix. Also, it may only move in direction of increasing i or increasing j, or both (diagonal). A visualization of the warping path and the step pattern is shown in Figure 4.5. Note that the indexes of the distance matrix in this representation rise from the lower left to the upper right corner. A warping path is formally defined as

$$\phi = \left\{ \phi_x(k), \phi_y(k) \,\middle|\, \begin{array}{l} k = 1, ..., K, \\ \max(T_x, T_y) \leqslant K \leqslant T_x + T_y - 1 \end{array} \right\} \tag{4.5}$$

where $\phi(k)$ is defined as the k^{th} step of the warping path, containing the position $(i, j)_k$ in the distance matrix.

Given a warping path ϕ, d_ϕ is defined as the normalized and accumulated distance along the warping path:

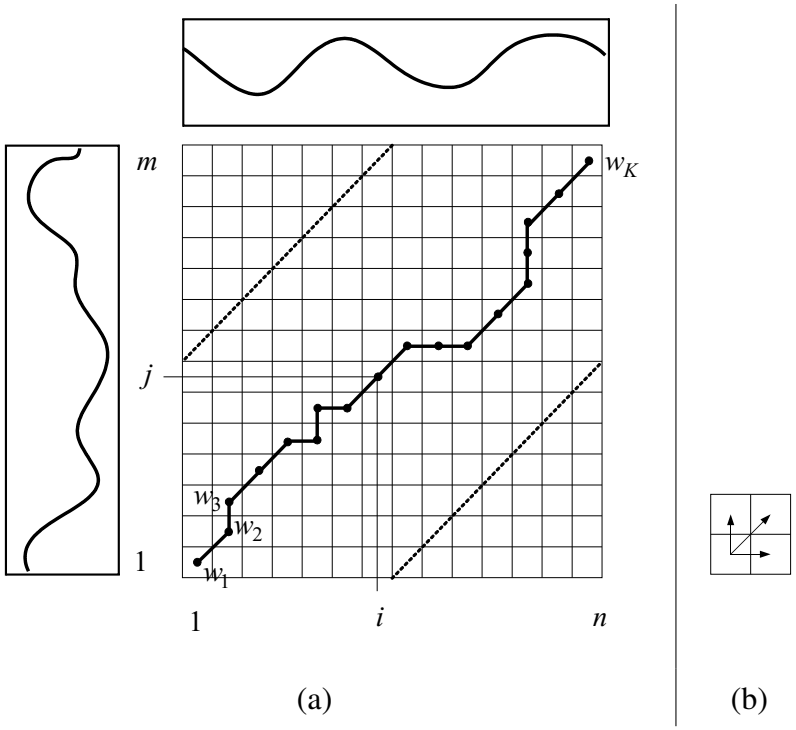


Figure 4.5: (a) An illustration of a warping path for a input (on the top-side of the grid), and a template (on the left-side of the grid). (b) The step pattern showing three possible steps of the warping path .

$$d_\phi(X,Y) = \frac{1}{K}\sum_{k=1}^{K} d(\phi_x(k),\phi_y(k)) \tag{4.6}$$

where K is a normalization factor to compensate for the fact that warping paths may have different length. There are exponentially many warping paths that satisfy the above equation. We are only interested in the path which minimizes the warping cost:

$$d(X,Y) = \min_{\phi} d_\phi(X,Y) \tag{4.7}$$

The minimum warping path is efficiently found using dynamic programming [Bel57]. Dynamic programming is an overall solution combing a set of sub solutions, which is combined by a set of another sub-solutions, and so on. Minimization of the cost in the warping path is simply determined by taking the minimum cost of all previous points within a valid step path. The costs of the previous points are again determined by selecting the minimum of their own predecessors.

This minimization problem is implemented by creating a $(n+1)$-by-$(m+1)$ cumulative distance matrix that contains all cumulative costs to reach element (i, j). Costs at the two borders $(1, j)$ and $(i, 1)$ are initially set to infinity for keeping the warping path within the bounds of the matrix. The cumulative distance at the

starting point $(1,1)$ is set to zero. The matrix is iteratively filled with accumulated distances along the directions of the step pattern, always choosing the minimum cumulative distance of its predecessors.

Optimizations

The DTW algorithm works by keeping track of the cost of the best path to each point in the grid. There are several optimizations to find a warping path through the DTW grid. A warping path ϕ is typically subject to several constraints:

- **Start-End point Constraint**: This constraint forces the first element of X to map to the first element of Y (i.e. $\phi(1) = 1$) and the last element of X to the last element of Y (i.e. $\phi(T_m) = T_n$). The path starts at the bottom left and ends at the top right.
- **Monotonic Constraint**: This constraint ensures that the warping maintains the temporal ordering of points in X, i.e. $\phi(t_i) \geq \phi(t_j)$ for $t_i \geq t_j$. The warping path will not turn back on itself, both the i and j indexes either stay the same or increase, They never decrease.
- **Slope Constraint**: The path should not be too steep or too shallow and a very short sequence should not be matched to a very long sequence. This constraint is expressed as a ratio n/m where m is the number of steps in the X direction and m is the number in the Y direction. In other words, after m steps in X you must make a step in Y and vice versa.

By applying these constraints, we restrict the moves that is made from any point in the path. For instance, with the slope constraint of $P = 1$, if a path has already moved one square up, it must next move either diagonally or to the right. Using these constraints, the umber of possible paths is then efficiently defined instead of using all.

4.4.2 Variations of DTW

We applied several variations to the standard DTW to fullfill the needs of our framework.

Multi-sensor Temporal Sequences

The standard DTW uses one-dimensional sequences. However, since we use different sensors in our framework, the output signal consists of multiple sensor data. The extension to multisensor temporal sequences is straightforward and needs only minor changes in the standard DTW algorithm. For instance, the input sequences $X(I \times N)$ and the template sequences $Y(J \times N)$ represent multisensor temporal sequences where N is the number of variables.

In the standard DTW algorithm, $N = 1$ and the local distance is measured by calculating only the difference between two values: X_i and Y_j. To extend to multiple sensors, only the local distance measure needs to be extended to calculate the difference between the two vectors: ${N_i}^X$ and ${N_j}^Y$. There are several ways to calculate distance: the Euclidean distance, Manhattan distance, Cosine correlation coefficient etc [Web02]. For instance, local measures with Euclidean distance for multisensor temporal sequences is defined as:

$$d(X_i^N, Y_j^N) = \sqrt{\sum_{n=1}^{N} WV(n) \left(X_{i(n)} - Y_{j(n)}\right)^2} \tag{4.8}$$

Derivative Dynamic Time Warping (DDTW)

The standard DTW often fails to find natural alignments in two sequences because a feature in one sequence is slightly higher or lower than its corresponding feature in the other sequence. This situation occurs in the sequence from the accelerometer body sensor when the gesture is performed with different speed and power. The reason is related to the characteristics of gesture sensor signals. Our body sensor signals from the accelerometer produce a signal in which a feature (peak, valley, inflection point, plateau, etc.) exists.

To solve this problem, we applied another variation called *Derivative Dynamic Time Warping* (DDTW) [KP01]. DDTW finds more correct alignments in two sequences in this situation. With DDTW the distance measure is not Euclidean but rather the square of the difference of the estimated deprivates of x_i and y_j . While there are sophisticated methods for estimating derivatives, we use the following estimates for simplicity and generality:

$$D(x_i) = ((x_i - x_{i-1}) + (x_{i+1} - x_{i-1})/2)/2 \tag{4.9}$$

where $D(x_i)$ is the estimated derivative of x $_i$is the average of the slope of the line through the point question and its left neighbor, and the slope of the line through the left neighbor and the right neighbor. We pre-process data with smoothing filter to reduce noise in data. Figure 4.6 shows the two different results of standard DTW alignment and DDTW alignment in our analysis.

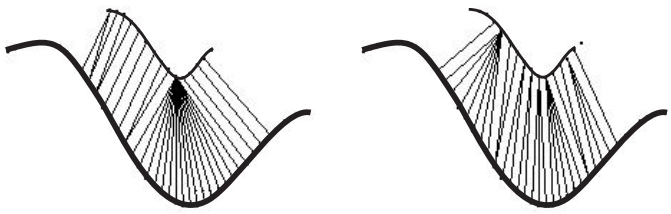


Figure 4.6: Signal alignment with DTW (Left) and DDTW (Right). Note that DDTW provides more natural alignment in regions of high curvature.

Relaxed Start-End Constraints

As described in the previous section, due to the start-end constraint, the warping path should start and finish in diagonally opposite corner cells in the matrix. However, it is difficult to find the precise start and end points in continuous gesture signals. Thus, the original DTW algorithm should be modified to consider this problem. Our framework uses a variant that allows the start and end point to fall in a defined region. Therefore, the warping path can start at the first point of the prototype gesture sequence and within the start region of the test sequence, and ends at the end region of the test template and last of point of the prototype template.

Mahalanobis Distance

The final variation is to use a Mahalanobis distance [BB01]. The Mahalanobis distance is a weighted Euclidean distance by the inverse of the covariance matrix. This provides a statical measurement how well the unknown sequence matches with the trained sequence set based on the distribution of the values.

$$D_i^{Mahalanobis} = \sqrt{(\mathrm{X}-\mu_i)^T \sum_i^{-1} (\mathrm{X}-\mu_i)} \tag{4.10}$$

where s_t is a point at a time step t and μ_t is the mean at the same time step and represents the inverse of the covariance matrix of prototype. Please note that the Mahalanobis distance is the same as the Euclidean distance if the covariance matrix is the identity matrix.

4.5 HMM-based Gesture Modeling

There are many uncertainties when identifying gesture using sensors because there are various errors in performing and measuring gestures. Stochastic modeling is a flexible general method for modeling such problems. Hidden Markov Modelings (HMMs) have been used to handle such a stochastic data problems associated with time series. The most successful example is an automatic speech recognition. In speech recognition, the states are determined indirectly from the speech signal observations and commonly represent word or phonemes. Word and phoneme models are then defined as a consecutive series of states. Our framework uses HMMs to model 3D spatial gesture.

This section explains the fundamentals of HMMs and the basic tools in hidden Markov modeling of gesture signals, (the forward-backward algorithm, the Baum-Welch algorithm, and the Viterbi algorithm). Then we describe our 3D spatial gesture modeling which is based on the structure of our gesture unit motion chunk.

4.5.1 Fundamentals of Hidden Markov Model

Markov Processes

There is often significant structure embodied in a 3D spatial gesture. The start posture often greatly influences subsequent gestures. That is, a certain gesture is followed by a certain posture and the probability of performing the gesture depends very much on the posture that occurs before it. Our framework uses the *Markov* process to describe some of the high probability structures preserved in 3D spatial gestures.

The Markov process is summarized by the Markov property - for any sequence of time domain events the conditional probability density of a current event given all the past and present events depends only on the most recent j events. This is represented as follows:

$$P(X_{t+1} = s_{j_{t+1}} | X_t = s_{j_t}, X_{t-1} = s_{j_{t-1}}, ..., X_0 = s_{j_0}) = \\ P(X_{t+1} = s_{j_{t+1}} | X_t = s_{j_t}, ..., X_{t-j+1} = s_{j_{t-j+1}}) \tag{4.11}$$

Figure 4.7 shows an example of a Markov process where each state is indicated by its circle. The graph visualizes a finite state machine that consists of states, transitions and actions. Each directed line is a transition from one state (i) to another state (j), whose probability is indicated by a_{ij} alongside the line. The transitions are normally represented by a simple probabilistic model while the states generally involve more complex stochastic models. In an observable Markov model, the states correspond to random processes whose outcomes are directly observable.

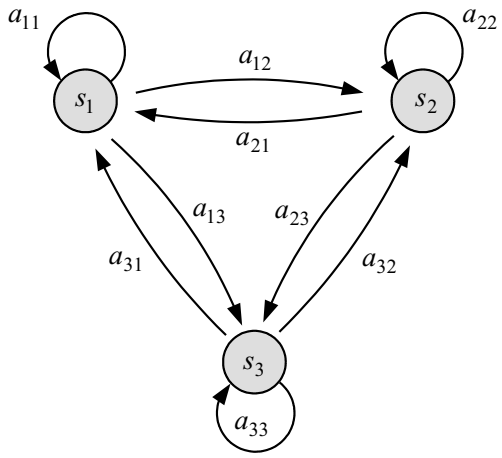


Figure 4.7: An example of a Markov process. At each time step the state is changed with a given transition probability a_{ij}.

Definition of the Hidden Markov Model

In the previous section, we presented that Markov models that are used for a study of observed state outputs. However, such a model is applicable to many problems of interest. In a *hidden* Markov model (HMM), the output for each state will be *hidden* (not observable) and can only be observed through another set of observable stochastic processes. Our framework uses HMM to model the changing statistical characteristics that exist in the actual observations of gesture signals.

The state transition is a stochastic process and a sequence of *hidden states* is inferred from the observed data. Each hidden state of the model is associated with a set of output probability distributions, which is characterized by either discrete probability distributions or continuous probability distribution functions.

As shown in Figure 4.8, HMMs is described in terms of two random processes. The first process is a discrete-time Markov chain, meaning that the system is in one of a finite number of states $(s_1, ...s_N)$ at each time $t = 1, 2, ...$ The process starts in one of these states and moves successively from one state to another. The state change is determined by a set of transition probabilities (a_{ij}) associated with the current state. The second random process generates an output symbol $o \in \{o_1, ..., o_k\}$ at every time step, subject to a probability distribution b_i, which depends on the current state.

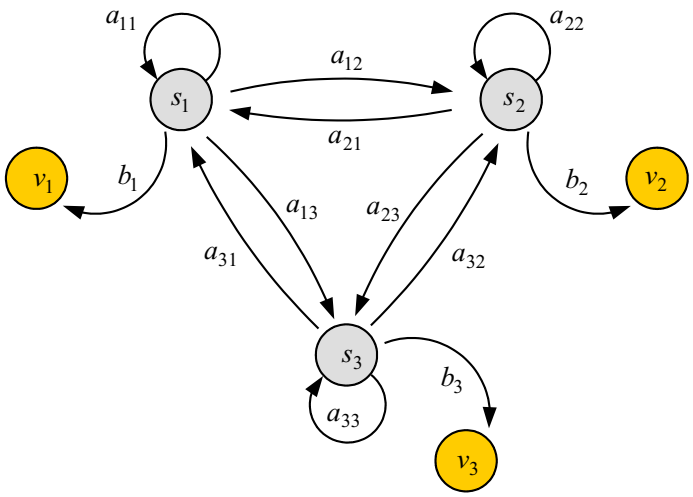


Figure 4.8: An example of Hidden Markov Modeling. A Hidden Markov Model consists of two random processes: the first process is a Markov chain with a state transition and the second process provides observable output symbols depending on these hidden states.

We can describe the HMM formally with a set of notations. An HMM (λ) is defined by the number of states N and three sets of probability distributions. The states (Q) are simply labeled $\{1, 2, ..., N\}$ and the state at time t is denoted as q_t. There are three different probability distributions. First, an HMM includes the

initial state distribution, $\pi = \{\pi_i\}$, where $\pi_i = P\{q_i = i\}$, $1 \leq i \leq N$. The second probability distribution is a set of state transition probabilities, $A = \{a_{ij}\}$, where $a_{ij} = p\{q_{t+1} = j | q_t = i\}$, $1 \leq i \leq N$. q_t denotes the current state. The third set of distribution is an output probability in each of the states, $B = \{b_j(k)\}$, $b_j(k) = p\{o_t = v_k | q_t = j\}$,$1 \leq j \leq N$, $1 \leq k \leq M$, where v_k denotes the k^{th} observation symbol in the alphabet, and o_t the current parameter vector.

If the observations are continuous then we use a continuous probability density function, instead of a set of discrete probabilities. In this case we specify the parameters of the probability density function. Usually the probability density is approximated by a weighted sum of M Gaussian distributions $\mathcal{N}$,

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\mu_{jm}, U_{jm}, o_t) \tag{4.12}$$

where c_{jm} is the weighting coefficients, μ_{jm} are mean vectors, and U_{jm} are covariance matrices. Figure 4.9 shows a one dimensional Gaussian mixture probability distribution function consisting of two single Gaussians (red and blue).

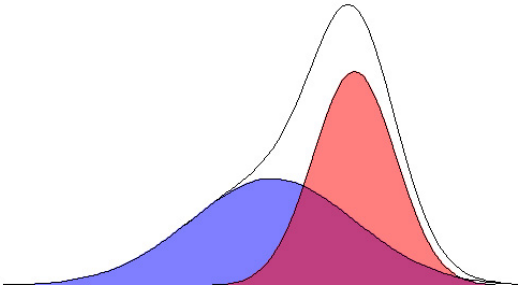


Figure 4.9: An example of two Gaussian mixtures. A Gaussian mixture probability distribution function is constructed from two Gaussian probability distributions (red and blue).

Based on the notations described above, we can define a complete notation of a HMM as follows: a HMM with discrete probability distributions,

$$\lambda = (\pi, A, B), \tag{4.13}$$

and a HMM with continuous densities,

$$\lambda = (\pi, A, c_{jm}, \mu_{jm}, U_{jm}). \tag{4.14}$$

Specification of an HMM involves the choice of the number of sates, N, the number of discrete symbols L, and specification of three probability densities with matrix form A, B, and π. A set of initial states Q_I and final states Q_F can also be defined. Thus, transitions must start from one of Q_I and and end at one of Q_F.

Each HMM is made up of a number of states; the number of states per model is another design decision, which needs to be made by the system designer. There are

several types of HMMs with different constraints on the transition probabilities. For instance, a *left-right HMM* has the property that limits state transitions only to consecutive states. Figure 4.10 shows an example of the left-right HMM that consists of four states without no backward transition. These left-right HMM models have been widely used for speech recognition because of the relatively short duration of phonemes and an underlying structure of language based on phonetic ordering. Another example is a topology in which the states are fully-connected, i.e. transitions may occur from any state. This type of model is known as an *ergodic HMM*. In this case, any state is reached from any other state in a finite number of steps.

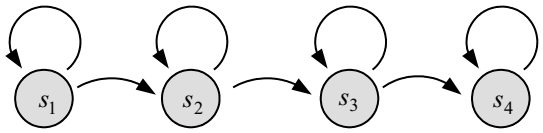


Figure 4.10: An example of a left-right HMM that consists of four states. State transition is limited to the current state and the next state without having the backward transition.

4.5.2 Motion Chunk based HMM

In our framework, HMM models general parameter changes over time through the changes in gestural states. The meaning of the states themselves is not easily defined from a perceptual standpoint, but they are common parameter combinations derived from the gesture signal relating to aspects of the physical state of the gesture at a given time. The state-to-state motion captures some of the enormous variety of physical configurations involved in gesture.

In HMM-based gesture modeling, a gesture signal is viewed as a piece-wise stationary signal or a short-time stationary signal. That is, one could assume gesture is approximated as a stationary process in a short-time. Here, the power of the HMM lies in the fact that the parameters that are used to model the gesture signal is well optimized resulting in lower computational complexity in the decoding procedure as well as improved recognition accuracy. Furthermore, other knowledge sources are incorporated by extending the structure of the HMM.

In an ideal case, the trained states need to be concentrated clusters of data in the parameter space. Then, the state path is a good approximation to the actual parameter trajectory, in which the estimated trajectory passes through the mean of each state.

We apply the structure of the motion chunk to design the structure of the HMM for 3D spatial gesture. To this end, five states left-right HMM so called *motion chunk based HMM* is designed as illustrated in Figure 4.11. Intuitively our HMM

corresponds to one state for the transition into the gesture, one for the middle part and one for the transition out of the gesture. The first sate and end state are equivalent to the start-static chunk and the end-static chunk respectively. The three in-between states are used for dynamic chunk features only. Each state in the model corresponds to some part of the input gesture signal. Given a set of input gesture sequences, we can train a gesture model on the data and then estimate the conditional probability of an unknown sequence given the model.

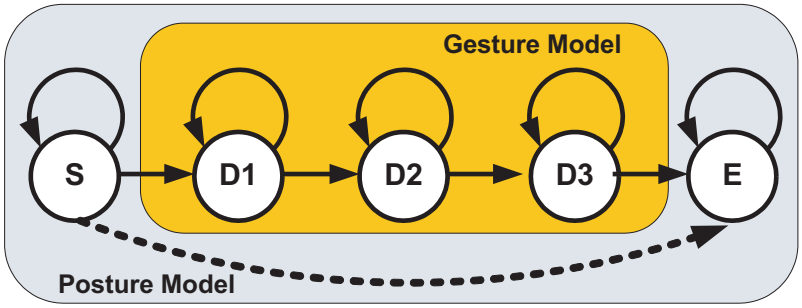


Figure 4.11: A five state left-to-right HMM with state transitions for 3D spatial gestures. The gesture model is designed based on the motion chunk structure chaining together postures and gestures. Optionally, the three internal states of the dynamic chunk are omitted for the posture model.

Static chunks are skipped by directly connecting a start state to an end state. We use the two state HMM as *posture model* and the complete five state HMM as *gesture model*. The five states might be linked in a chain where transitions are only allowed to higher numbered states or to themselves. Alternatively, each state might be all linked to all others, the so-called ergodic model. These two structures are common but many other combinations are also possible.

Following the previous notation of HMMs, the posture model is represented by the transition matrix $A = \{a_{S,S}, a_{S,E}, a_{E,E}\}$, the initial probability $\pi = \{\pi_S\}$, and $B = \{b_S(O), b_E(O)\}$ for the observation probability distributions for each state given the observation $O = \{O_S, O_E\}$. The main idea behind this design is that the posture model, with two states, can be trained with two posture sequences separately from the gesture model with five states. We apply this posture model to the gesture registration process to detect input training gestures automatically.

The observation vectors for the motion chunk are the gesture features from sensors. Since the observations are taken every window frame, each state will correspond to one signal window, and state transition will occur at this rate. The number of states was determined empirically by evaluating models with different number of states (e.g. 3, 7, and 11) in terms of sampling speed and resolution. 5-state models were chosen qualitatively as a compromise between the divergent goals of encompassing the wide variability of each motion chunk's observed features and limiting the computational complexity of the model.

We do not use the codebook approach used in speech recognition. The code-

book approach is generated by the quantization process are constructed using training data from all motion chunk templates. However, when a new motion chunk is added, we need to reconstruct the codebook and retrain all system modules. With continuous observation densities provided by motion chunk states, we do not need to train the system from the beginning, Since there is no codebook to be constructed, we only need to train the newly added motion chunk.

4.5.3 Three Basic Problems of HMM

There are three fundamental problems associated with the use of HMMs [Rab89]:

- *The evaluation problem for estimating conditional probability:* Given the observation sequence O $= O_1, O_2, ..., O_T$, and the model $\lambda = (A, B, \pi)$, the problem is how to compute $P(O|\lambda)$, the probability that this observed sequence is produced by the model. In other words, given a set of models and a sequence of observations, how do we choose the model which best matches the observations for the purpose of recognition.
- *The decoding problem for finding the most likely state sequence:* Calculate the most likely sequence of HMM states for new observations given a trained model, given the observation sequence $O = O_1 O_2 \ldots O_T$, and the model λ, how do we choose a corresponding state sequence $Q = q1q2 \ldots q_T$, which is optimal in some meaningful sense (best explains the observations). This problem relates to recovery of the hidden part of the model.
- *The learning problem for adjusting the model parameters:* Given the observation sequence $O = O_1 O_2 \ldots O_T$, how do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$. The problem concerns how to optimize the model parameters so as to describe how the observations came about.

There are well established solutions to each of these problems. The following sections will describe how we used an algorithm to solve each problem in our framework.

The Evaluation Problem using the Forward Algorithm

The evaluation problem is to find $P(O|\lambda)$ given a model $\lambda = (\pi, A, B)$ and a sequence of observations O $= O_1, O_2, ..., O_T$. We calculate the probability using simple probabilistic arguments. But this calculation involves a number of operations in the order of N^T for N states and T observations. This will be very large, even if the length is of considerably low complexity, T is moderate. An efficient algorithm for this problem is the *Forward-Algorithm* that has a considerably low complexity and makes use of an auxiliary variable, $\alpha_t(i)$ called the forward variable.

The forward variable is defined as the probability of the partial observation sequence $O = O_1, O_2, ..., O_T$, when it terminates at the state i. Mathematically,

$$\alpha_t(i) = p\{o_1, o_2, ..., o_t, q_t = i|\lambda\} \tag{4.15}$$

Then it is easy to see that the following recursive relationship holds.

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}, 1 \leq j \leq N, 1 \leq t \leq T-1 \tag{4.16}$$

where, $\alpha_T(i), 1 \leq j \leq N$. Using the recursion we can calculate $\alpha_1(j) = \pi_j b_j(o_1), 1 \leq j \leq N$ and then the required probability is given by,

$$p\{O|\lambda\} = \sum_{i=1}^{N} \alpha_T(i) \tag{4.17}$$

The complexity of this method, known as the forward algorithm, is proportional to N^2T, which is linear to T whereas the direct calculation mentioned earlier, had an exponential complexity.

In a similar way we can define the backward variable $\beta_t(i)$ as the probability of the partial observation sequence $O = O_{t+1}, O_{t+2}, ..., O_T$, given that the current state is i. Mathematically,

$$\beta_t(i) = p\{o_{t+1}, o_{t+2}, ..., o_T|q_t = i, \lambda\} \tag{4.18}$$

As in the case of $\alpha_t(i)$ there is also a recursive relationship which can be used to calculate $\beta_t(i)$ efficiently.

$$\beta_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1}), 1 \leq j \leq N, 1 \leq t \leq T-1 \tag{4.19}$$

where, $\beta_T(i), 1 \leq j \leq N$. Further, we see that,

$$\alpha_t(i)\beta_t(i) = p\{O, q_t = i|\lambda\}, 1 \leq j \leq N, 1 \leq t \leq T-1 \tag{4.20}$$

This gives another way to calculate, by using both forward and backward variables.

$$p\{O|\lambda\} = \sum_{i=1}^{N} p\{O, q_t = i|\lambda\} = \sum_{i=1}^{N} \alpha_T(i)\beta_T(i) \tag{4.21}$$

The Decoding Problem and the Viterbi Algorithm

The state-path representation is central to this framework, as it compactly represents the essential dynamics of the gesture observations. The variation of parameters over time is reflective of the expressive characteristics of the gesture performer. The hypothesis is that this representation will be rich enough to fully encode the time-varying aspects of the gesture.

The decoding problem is to find the most likely succession of states, given a sequence of observations $\mathrm{O} = O_1, O_2, ..., O_T$ and the model probabilities A, B, and π. There are multiple meaningful optimality criteria. The most popular is to find the state sequence which maximizes $P(Q|O,\lambda)$. This problem is solved with a dynamic programming algorithm called *Viterbi Algorithm*. In order to facilitate the computation, an auxiliary variable is defined as follows:

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} p\{q_1, q_2, ..., q_{t-1}, q_t = i, o_1, o_2, ..., o_{t-1}|\lambda\}, \tag{4.22}$$

which gives the highest probability that the partial observation sequence and state sequence up to t=t can have, when the current state is i. It is easy to observe that the following recursive relationship holds:

$$\delta_{\mathrm{t}+1}(j) = b_j(o_{t+1}) \left[\max_{1 \le i \le N} \delta_t(i) a_{ij} \right], 1 \le i \le N, 1 \le t \le T-1 \tag{4.23}$$

where $\delta_1(j) = \pi_j b_j(o_1), 1 \le j \le N$. So the procedure to find the most likely state sequence starts from calculation of $\delta_{\mathrm{T}}(j)$, $1 \le j \le N$ using recursion in the previous equation, while always keeping a pointer to the *winning state* in the maximum finding operation. Finally the state j^*, is found where

$$j^* = \arg \max_{1 \le j \le N} \delta_T(j), \tag{4.24}$$

and starting from this state, the sequence of states is back-tracked as the pointer in each state indicates.This gives the required set of states.

This whole algorithm is interpreted as a search on a graph whose nodes are formed by the states of the HMM in each of the time instant t, $1 \le j \le T$.

The Learning Problem

Generally, the learning problem is to adjust the HMM parameters to determine a HMM model for each gesture. We use the multiple observation sequences K (training data), where $P(O|\lambda) = \prod_{k=1}^{K} P(O^{(k)}|\lambda)$ is maximized and determine the model parameters A, B, and π.

The quantity we wish to optimize during the learning process is different from application to application. There may be several optimization criteria for learning, out of which a suitable one is selected, depending on the application. Two

main optimization criteria have been widely used in pattern recognition literature; *Maximum Likelihood* (ML) and *Maximum Mutual Information* (MMI). In our framework, we use ML which maximizes the probability of a given sequence of observations O^g, belonging to a given gesture class g, given the HMM λ_g of the class g. This probability is the total likelihood of the observations and can be expressed mathematically as

$$L_{tot} = p\{O^g|\lambda_g\}. \tag{4.25}$$

There is no known way to solve this problem analytically. The most common HMM training technique is an iterative method that finds a local maximum of L_{tot} such as the *Baum-Welch* or *Expectation-Maximization* (EM) algorithm. For these iterative algorithms, the choice of the initial parameters is important in order to find the global instead of a local maximum. In our framework, we use EM training algorithm to determine the model states according to the density of the observations in parameter space.

4.5.4 Comparison between HMM and DTW

In this section, we summarize the concept of the Hidden Markov Model (HMM) and the Dynamic Time Warping (DTW) used in our framework. As described earlier, HMM is a statistical model of a sequence of feature vector observations.

The concept of HMM can be viewed as the integration of statistical methods with the DTW technique which absorbs time variations of gesture signal patterns. The HMM is a parametric modeling technique in contrast to the non-parametric DTW algorithm. If the Viterbi Algorithm is used for decoding in HMM-based gesture modeling, it is actually similar to the DTW algorithm. The difference is that HMM uses the probability between the input and template model and DTW computes the distance measure between two gesture signals.

This can be viewed as the information theoretic expansion of the distance to a probability. For instance, DTW uses the frame distance $d(x_i, y_i)$ as described in the previous section. In HMM, instead of using the frame distance, it uses probability $Pr(x_i|s_j)$ which is a likelihood that the input frame x_i is produced from the template state s_j. In addition, the use of the weight function $w(k)$ to find a warping path in DTW is considered as the transition probability from one state to another possible state in the template.

In general, the HMM is used to build a probabilistic model of a sequence of observations. The gesture template results in a sequence of states which have a probability of emitting the input frame instead of a sequence of real data frames used in DTW. In practice, the number of states should be reduced efficiently but this is a trade off between obtaining the probabilistic model and losing the time information in the template. For example, if the number of the states is reduced to one, the final model considered is a single Gaussian model. Therefore, the HMM

can be seen as an extension of the simple Gaussian model which combines the probabilities of a sequence of observations into an overall probability score.

4.6 Gesture Registration

While there are a number of approaches to recognize gestures, the available gestures are mainly designed to show the performance of the proposed techniques for gesture acquisition and recognition. In our framework, the gesture model is designed to support users registering spatial gestures. Therefore, users can design gestures in a systematic way, and test many of them for target applications in a short period improving the usability.

In general, gestures available in the system are categorized into groups (*pre-designed* and *user-designed*). Pre-designed gestures are provided by a system developer. During the system development, gestures are designed with a set of gesture principles and standards that are proved by gesture experts. In this case, end-users are required to learn the pre-designed gestures. They have to spend time to practice and memorize the gestures before the gesture interface is really usable to them. This learning process is an explicit upfront burden when the required gestures are complex. On the other hand, end-users can design their own gestures. This user-centered approach minimizes the gesture learning time by asking them to register their own gestures before using the interface.

In our framework, there are two separate steps with which end-users can follow to register individual 3D spatial gestures: a *posture registration* step and a *gesture registration* step as illustrated in Figure 4.12. Following these two steps, the HMM for the registered gestures is trained using the recorded samples.

During posture registration, users are asked to provide the start posture and the end posture for a certain time period (2 or 3 seconds). The posture data is used to adjust the parameters of the posture HMM model as explained in the previous section.

Once the posture model is trained, the system employs it to automatically discriminate training gestures from arbitrary input gestures such as recovery gestures or rest gestures. The detection is accomplished if $P(O_S, O_E|\lambda)$ is above a certain threshold (typically 90%). This approach significantly simplifies the user's effort to manually segment and detect training gestures. The posture model is stored and activated whenever the user adds new training data as long as the posture model remains unchanged.

4.7 Gesture Recognition

Gesture recognition identifies a type of gesture performed by a user. Depending on the signal, the system chooses the gesture template that most closely matches

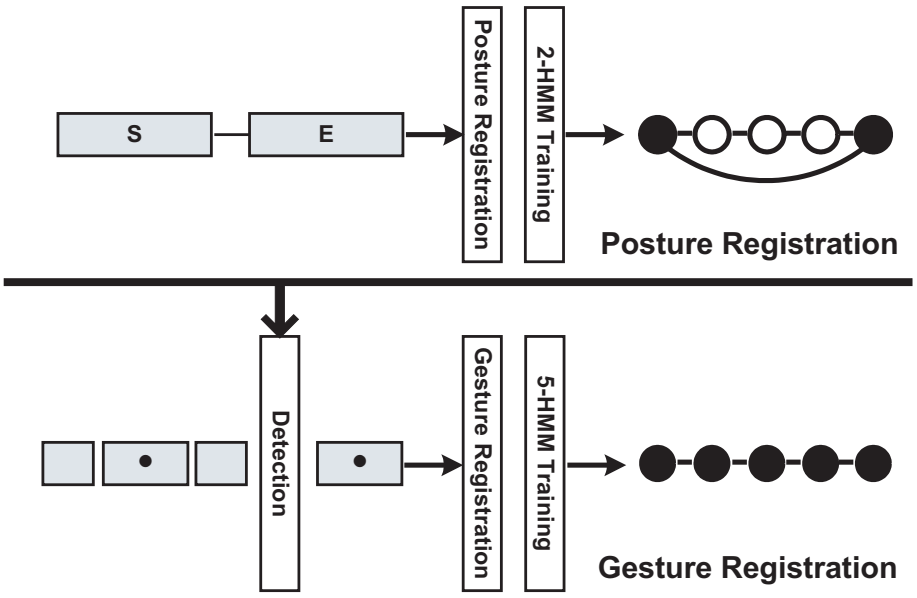


Figure 4.12: An overview of our gesture registration process which consists of posture registration (top) and gesture registration (bottom). In the posture registration, the two-state posture HMM model is trained using a start posture and an end posture. The gesture registration uses the trained posture model to detect meaningful gestures from arbitrary gestures and trains the full 5 state gesture HMM model.

the input gesture. Simple gestures are easily distinguished by the values of the sensors. However, complex 3D spatial gestures are harder to identify.

In this framework, we developed two different algorithms for gesture recognition: the *DTW recognizer* and the *HMM recognizer*. The main idea is to develop a gesture interface which is more accessible to end-users. For instance, the DTW recognizer allow us to use the system with only a single training data.

When more gestures are provided during usage, the HMM recognizer is trained according to the provided gestures. This way, the HMM recognizer is continually trained in an on-line manner, resulting in a user-specific recognizer without an explicit intensive pre-training phase. Using the HMM recognizer, gesture recognition is viewed as a problem of probabilistic inference. The system is more robust in different gesture performances and noisy sensor measurements.

4.7.1 The DTW recognizer

Once the gesture templates are prepared for the DTW recognizer, recognition is achieved by the DTW. DTW computes similarity between the multisensor temporal sequences in the database of gesture templates and the other multisensor temporal sequences segmented from the collected multisensor data. The type of input gesture is selected by the template that minimizes the overall distance to the input gesture.

The accuracy of the DTW recognizer greatly relies on the quality of gesture templates. In the presence of noise, DTW might fail to measure the correct similarity between two sequences, since DTW tries to match all elements between the two sequences. To eliminate this problem, the system should provide a suitable filter to reduce noise and outliers. In addition, we also have to normalize sensor data. Normalization is typically performed by integrating vector components of varying dynamic range. If some vector components have a variance that is significantly higher than the variance of other components, those components will dominate the results of the local distance measure. Some common normalization methods have been investigated to normalize the dynamic ranges of the vector components in the interval (0, 1) such as Variance normalization, Min-Max normalization and Softmax normalization.

The templates are set either by an end-user or a gesture designer. The created gesture templates by the end-user are more reliable than the pre-defined templates because the templates preserve some unique characteristics of the user. On the other hand, gestures can be designed for specific application domains (e.g. rehabilitation).

Assume we have N gestures, G_n, $1 \leq n \leq N$. Given our motion chunk representation we obtain the following three DTW distances from the three chunks for each template: $D_{n,S}$ for the start chunk, $D_{n,D}$ for the dynamic chunk, and $D_{n,E}$ for the end chunk. Our decision rule is simply:

$$\hat{n_{DTW}} = \arg\min_{n} [D_{n,S} D_{n,D} D_{n,E}] \tag{4.26}$$

We use a set of three full chunks for distance computation because gestures usually exhibit a higher variability than postures. Our framework actually supports two different types of DTW recognizers depending on the number of templates: a single template DTW (SDTW) and a multiple-template DTW (MDTW). The main idea behind the MDTW is to improve the recognition rate by accommodating the variations between multiple templates. Since the multiple-template approach is computationally more expensive, it is necessary to find an optimal number of templates. In practice, three templates are sufficient in our tests.

When multiple template datasets are available, there are three different ways to prepare templates. First, we can compute the intra-class DTW distance that is the sum of DTW distances between a template dataset to all other datasets within the same gesture type. The one with the minimum intra-class DTW distance is selected as the gesture template. Second, a gesture template is prepared by averaging the available multiple datasets. However, it is non-trivial to average a collection of time series that are not perfectly time-aligned. Finally, multiple datasets are used for each gesture. Then we need to compute the distance for each dataset and combine the results. This method usually achieves better recognition rates than when using only a single dataset. However, this method is computationally inefficient because it increases the number of distance computations.

4.7.2 The HMM recognizer

The HMM recognizer was designed as an alternative to find the best matching gesture among the gesture templates. During the training phase, an HMM λ_n is built for each gesture G_n. Then, for each unknown gesture, the model computes the likelihoods for all possible models $P(O|\lambda_n)$, $1 \leq n \leq N$ and selects the gesture $G_{\hat{n}}$ with the highest model likelihood i.e.,

$$\hat{n_H} = \arg\max_n [\Pr(O|\lambda_n)] \tag{4.27}$$

While we generally recommend to utilize the five state HMM for recognition given the observation sequence $O = \{O_S, O_D, O_E\}$, the posture model with two states can in principle be applied as well. This is useful in cases where the dynamic chunk observations O_D are highly variant.

4.8 Gesture Evaluation

The gesture evaluation measures the similarity between two gestures using the similar techniques used in gesture recognition. The main task is to compare an input gesture to a template gesture that is performed by the same or different user, and determine how both performances are different. The result is used to improve user performance or to correct wrong gestures.

The evaluation process in our framework consists of both *posture evaluation* and *gesture evaluation*. We make the gesture instruction process similar to the practical motion training where users learn postures then perform gestures by connecting individual postures. Since posture and gesture co-occur, the coordination between postures is important for improving the efficiency of the gesture. The separation of postures and gestures helps users learn a complicated 3D spatial gesture in a systematic way.

We compute three distinct scores for the start static chunk, the dynamic chunk, and the end static chunk each. The evaluation of two static chunks measures how the start and end postures have been performed respectively. The evaluation of the dynamic chunk expresses how the gesture is performed, with respect to power and speed. Currently, the result of evaluation is a simple numerical score. We compare the input gesture with multiple reference templates stored in our database and take the minimum distance as the final score. In practice, the minimum distance is better suited to measure quality than mean or median. The scores are normalized to a maximum value of 100 and displayed on the screen in real-time.

Chapter 5

Experimental Evaluations

We conducted experimental evaluations to validate our framework in terms of gesture recognition (Section 5.1) and gesture instruction (Section 5.2). In these experiments, we analyze several issues to robust 3D spatial gesture interfaces and minimize operational errors against *mis-recognition* and *mis-instruction*. Mis-recognition occurs various reasons such as poor performance of gestures, insufficient training data for the gesture model, and poor design of gestures. Because of mis-instruction, it is difficult for users to know which gesture is for a certain operation and how the gesture should be performed. The system should inform users a set of available gestures in the gesture interface and the operations that can be controlled by the gestures.

5.1 The Gesture Recognition Task

In this task, we analyze the performance of our recognition methods (DTW recognizer and HMM recognizer). We focus on the single hand gestures as shown in Figure 5.1. We designed 18 different spatial gestures for the experiment. We compare recognition rates of our DTW and HMM recognizers under various conditions such as different user positions and orientations. We analyze robustness and invariance of gesture features and issues of human variability in performing 3D spatial gestures. We compare recognition rates for the user-dependent (D) and the user-independent (I) model.

5.1.1 Process

Our gesture vocabulary consists of 18 different 3D spatial gestures categorized into three style groups: planar-style, curved-style, and twisted-style. Figure 5.2 illustrates the example gestures with different sequences from simple to complex and different lengths from short to long.

Two subjects (male and female) were hired for our experiments. Each subject wore the mWire on the right wrist in the wristwatch fashion and wore the LED ring

Figure 5.1: An example of the performance of a 3D spatial gesture. A user wears the 3D input device mWire on the wrist. Four images show the sequence of performing a 3D spatial gesture from start posture to end posture.

on the index finger as illustrated in Figure 5.1. 2-dimensional accelerometer data was used to compute invariant body sensor features, and the relative 3D positions (rx, ry, rz) of the index finger tip were used as visual feature. The overall gesture vocabulary is demanding and poses a significant challenge to any recognition method.

Each gesture in our gesture vocabulary is repeated twenty times by the two subjects. Each subject was shown the gesture diagrams and was instructed in terms of gesture direction and gesture volume relative to their body size. After some practice, each subject was asked to perform the gestures in the same position. In addition, two other independent test datasets for translated (shifted) position and rotated position were acquired and utilized to test the invariance of the recognition, as illustrated in Figure 5.3. Our camera setup provides the active volume and distance (about two meters) regarding shift, and to the maximum rotation angle (60°). We used leave-one-out (LOO) cross validation to compute the recognition rates.

5.1.2 Results

The results of our experiments are given in Table 5.1. Overall, the five-state HMM with combined visual and body features performs best and achieves the highest recognition rates in all three user positions. We also compared the recognition performance for visual-only (V) and body-only (B) with those for combined visual-body (VB) features. As expected, the body-only features outperformed significantly the visual-only features in the rotated-position, reaching in a 15.9% error rate reduction. Conversely, and as expected, the visual sensor features perform better than the body sensor featuresfor shifted positions. This confirms our design decision to achieve both rotational and translational invariance by combining both features.

It is interesting that the visual-only features also gave relatively good recogni-

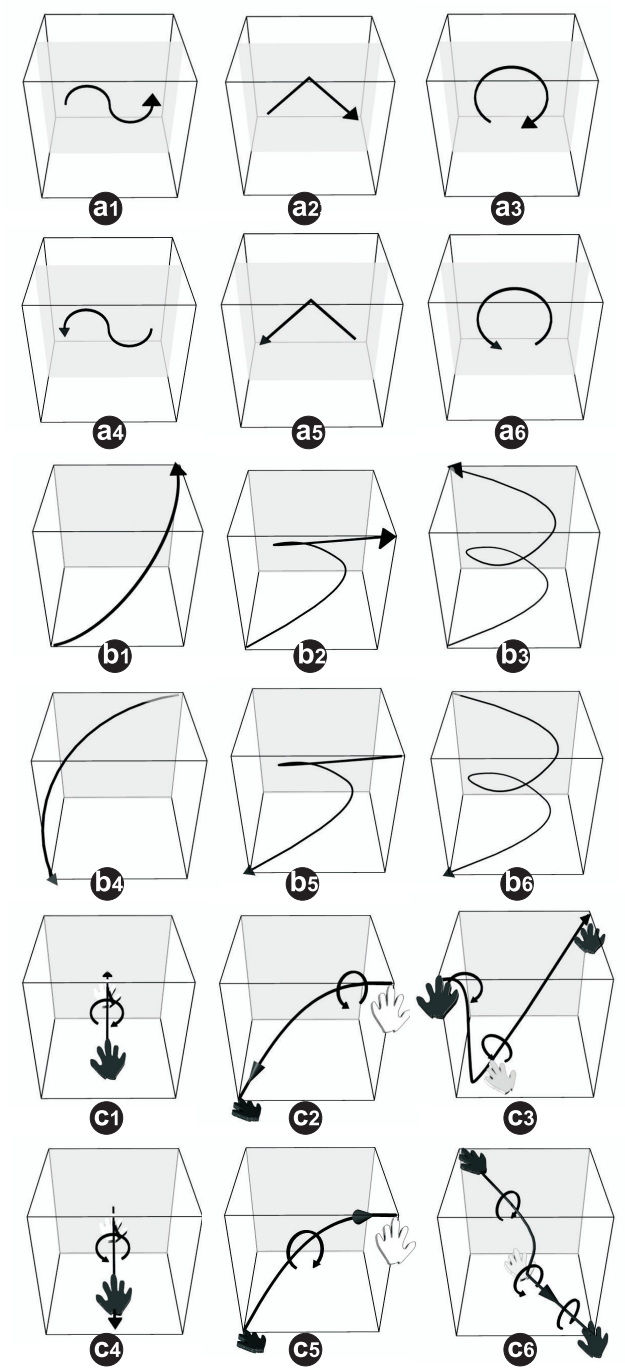


Figure 5.2: 18 gesture diagrams, with a box style 3D gesture volume, were used in our experiment. The 18 gestures are categorized into three style groups: (a) planar-style, (b) curved-style, and (c) twisted-style. The line indicates the trajectory of the gesture and the end of the gesture is illustrated by the arrow. The hand symbol uses black to indicate palm-down and white for palm-up. Note that some of the curved and twisted gestures are rather complex.

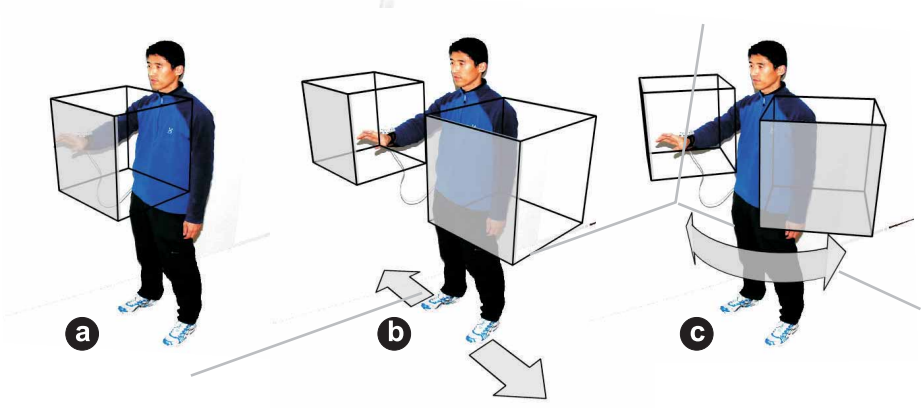


Figure 5.3: The three different user positions to acquire test data: (a) same (initial) position, (b) shifted position, and (c) rotated position. The gesture volume indicates the scale of the gestures relative to the body size.

tion results (60.0%) despite a 60° rotation. This can be attributed to the *rz* coordinate which is not very much affected by the rotations. By combining body and visual features, we achieved very good results in all three positions, given the size and complexity of our vocabulary. We can also see that the body features are not fully invariant with respect to shifted and rotated positions. During acquisition, subjects were requested to randomly change their positions in short time intervals to create more realistic situations. This added some additional variation to their gesture performances.

We carried out additional experiments to test the effects of gesture scale on recognition performance. A change of scale usually occurs when users become familiar with the the gestures. Users typically start with bigger gestures trying to conceive and learn hand-eye coordination. Later, gestures become more economical and smaller in size. Here the combined body and visual features are particulary useful because the visual features become somewhat less informative for smaller gesture volumes.

Even though the HMM recognizer is best, the result of the DTW recognizer is good considering the required number of training data. It is important to consider that we used twenty training datasets for the HMM recognizer, and only *a single* dataset for the single template DTW, and three training data for the multiple template DTW. The multiple template DTW, with three templates, outperformed the single template DTW. Thus, it is better to switch to a multiple template DTW when multiple training data is available. We tested different numbers of templates and found that three templates provide very good results and is a well-balanced trade-off between the computational costs and the recognition accuracy.

To analyze the performance variability between two subjects, we compared a user-dependent model (D) and a user-independent model (I) in terms of three different gesture styles. Table 5.2 illustrates the result of recognition accuracy in the "same position" data set. While the recognition rates of the user-dependent model

User Position	same	shifted	rotated	overall
V-5SHMM	96.0%	88.2%	60.0%	81.4%
B-5SHMM	94.5%	85.2%	75.9%	85.2%
VB-5SHMM	95.4%	93.1%	86.3%	91.6%
VB-SDTW	89.2%	86.7%	78.2%	84.7%
VB-MDTW	91.4%	89.3%	85.6%	88.7%

Table 5.1: A comparison of the user-dependent gesture recognition rate at three different user positions (same, shifted, and rotated) using an HMM with five states for visual-only features (V-5SHMM), and an HMM for body-only features (B-5SHMM), and an HMM for visual-body features (VB-5SHMM), the DTW with a single template for visual-body features (VB-SDTW), and the DTW with multiple templates (three) for visual-body features (VB-MDTW).

in both the HMM recognizer and the DTW recognizer are over 90%, the recognition rates of the user-independent model is below 50%. This is mainly due to the difference in the gesture performance between users which is caused by various factors such as different physical conditions and different intepretations of the gesture diagrams. Therefore, it is highly recommended to enable users to register 3D gestures in their own styles when developing 3D spatial gesture interfaces.

In addition, we found out that the illustration of 3D spatial gestures plays an important role in minimizing the human variability and improving the recognition rates. As shown in Table 5.2, the recognition rate of the planar-style gestures and twisted-style gestures are far superior to the curved-style gestures. The main reason is that our gesture diagrams for the curved-style (Figure 5.2-b) do not indicate the hand face (palm-down and palm-up) and the rotational direction of the hand in contrast to the twisted-style diagrams (Figure 5.2-c). Therefore, two subjects spontaneously turned their hand in different positions when performing the curved-style gestures. However, even though the planar-style diagrams (Figure 5.2-a) do not illustrate the hand face and its rotation either, both users performed the gestures with the palm of the hand in the correct position.

A trade-off exists between the generalization of the system to different users and the specialization to a certain user. Due to the different habits of the users, the differences in performance of the same gesture by different users (in-class difference) may overrun the differences between different gestures by the same user (between-class difference). In practice, this is often the main reason for misrecognition. Users either find two gestures indistinguishable to them, or they complain they cannot correctly recognize gestures. Thus, the system's performance is heavily dependent on the user's learning skills.

Gesture Type	planar	curved	twisted	overall
V-5SHMM(D)	92.6%	97.2%	98.2%	96.0%
B-5SHMM(D)	88.0%	97.2%	98.2%	94.5%
VB-5SHMM(D)	90.8%	97.2%	98.2%	95.4%
VB-MDTW(D)	88.5%	92.2%	93.7%	91.4%
V-5SHMM(I)	62.1%	18.3%	55.6%	45.3%
B-5SHMM(I)	55.0%	20.5%	53.1%	42.9%
VB-5SHMM(I)	69.8%	20.5%	60.7%	50.3%
VB-MDTW(I)	61.7%	25.7%	63.7%	50.3%

Table 5.2: A comparison of the gesture recognition rates at three different styles of 3D gestures (planar, curved and twisted) with a user-dependent model (D) and a user-independent model (I) with both DTW and HMM recognizers.

5.2 The Gesture Instruction Task

This section reports the experimental evaluation on the task of gesture instruction. We quantify the costs and benefits when using our gesture acquisition methods for learning new 3D spatial gestures.

5.2.1 Overview

Figure 5.4 shows the conceptual overview of the system that includes two types of users: *trainee* and *trainer*. Trainees create new gestures and register them to the system. Trainers learn the gestures by using the system. They observe the sensor data of the performed gestures and correct their gestures by comparing them with the trainer's data. The trainers' gestures can also be registered to the system. To this end, a 3D spatial gesture database can be constructed both from the trainers and the trainees.

Figure 5.4 shows how visual sensors and body sensors are used for gesture instruction. Our framework captures gestures wirelessly so that users can freely move and perform gestures within the interaction area. The body sensor precisely measures the tilt detection, movement, and vibration of the body parts. The accelerometer on a trainer's wrist provides precise tilt angles of the hand.

Visual sensors provide the images of the users, in real-time, like a mirror in conventional training places. The acquired images are recorded and combined with additional gesture information (visualization of body sensor data) to create a motion training video. Trainers and trainees can analyze their motions by watching

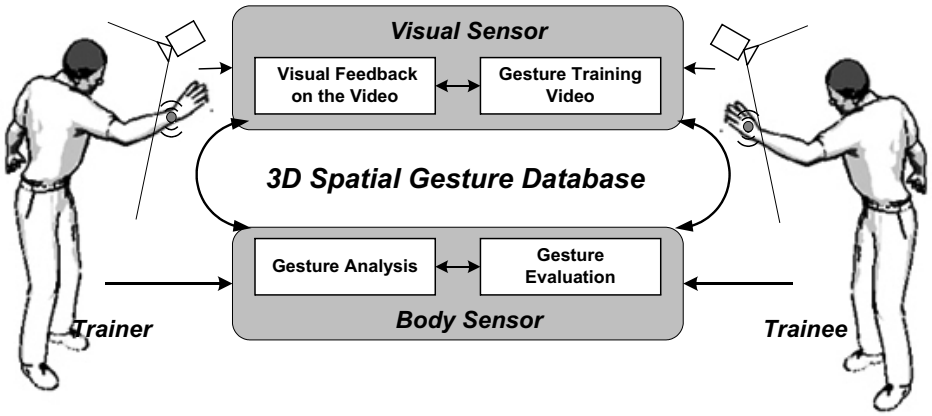


Figure 5.4: Conceptual overview of gesture instruction scenario. During the gesture instruction, two types of users (trainers and trainees) construct a 3D spatial gesture database. Visual and body sensors are used for gesture instruction (gesture analysis and generation of motion training video).

a hybrid representation of visual and body sensor data. In addition, the system automatically generates a motion training video using the captured image frames by cameras.

5.2.2 Motion Training Video

A motion training video is necessary for trainees and trainers as a reference to observe and analyze motions. However, producing such a video takes a lot of time. It requires simultaneous video recording during the trainer's performance. In addition, the captured videos need editing for motion training (selecting video frames and adding explanatory information).

Using the framework, we developed a method for the automatic generation of motion training videos. Figure 5.5 shows the pipeline for generating a motion training video using the visual and body sensor data. The training video is generated along with the gesture processing pipeline in the framework.

In a real time gesture instruction system, one challenge is to continuously capture and process human gestures. The generation of a training video uses the gesture representation and recognition techniques of our gesture processing framework. The gesture is segmented and represented using a flexible segment unit, motion chunk. Motion chunk allows us to store structured gesture information and manage it in a systematic way for gesture instruction.

The position of the body sensor is tracked on the acquired video frames in real time. We use colored LED markers whose brightness provides easy tracking in indoor environments and a simple vision tracking algorithm to find the pixel positions within a certain color and brightness range. While in the previous experiment

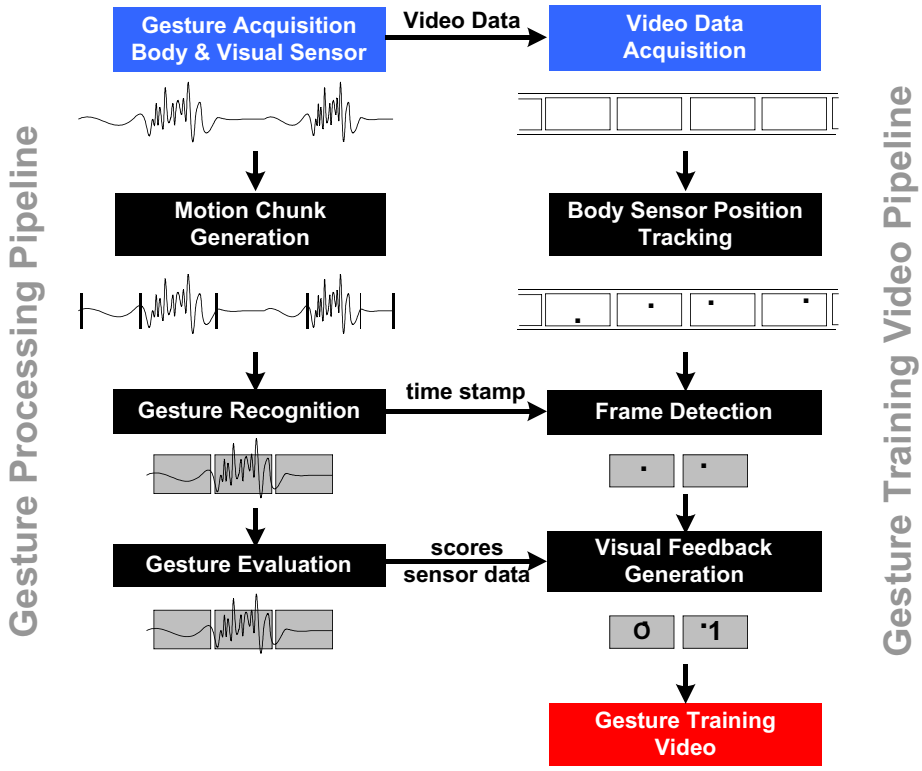


Figure 5.5: A pipeline for generating gesture training video using visual and body sensors. The training video is generated along the gesture processing pipeline, and additional information such as body sensor data is visualized on the images along the gestural path.

one LED is attached on the finger tip, in this experiment four LEDs are used at the four sides of the wrist bend. We can detect at least one point reliably even when the users perform gestures with their fist and when the hand is rotated in different directions. Figure 5.6 illustrates four cases where one, two, and three points are detected respectively. We use the center of the detected point as the position of the body sensor.

As soon as the input gesture is recognized, we save both the relevant video frames and the body sensor data. Then we generate a gesture training video that includes the visualization of body sensor data along the tracked sensor positions as illustrated in Figure 5.7. We used the tracked sensor positions and designed a simple template to display a moving circle along the gestural path. The circle size is changing as a function of the magnitude of the acceleration.

There are various design alternatives by using various shapes and the rules for shape transformation. Visual feedback helps users explain and improve their gesture performance. Users see how the body sensor data is changing with the appearance of their posture. Especially for the trainees, the body sensor data helps significantly in understanding a dynamic gesture between two static postures.

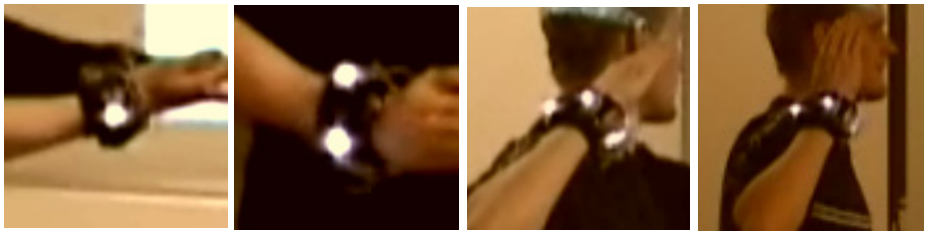


Figure 5.6: Four different situations occurred during body sensor tracking with visual sensors. Note that even when the hand is rotated, at lest one of the LEDs is visible on the camera images.

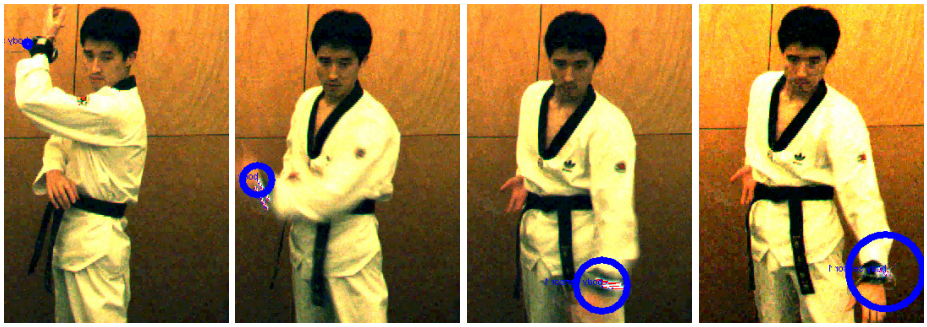


Figure 5.7: A visual feedback of accelerometer sensor data in video images. A circle is drawn on the position of the body sensor - its scale varies depending on the magnitude of the acceleration.

5.2.3 Process

We used a martial arts training scenario. Martial arts training is well suited to our experiment because it includes highly complex, precise motions which contain both postures and gestures. For this experiment, we hired a trainer who is a master of Taekwondo and six additional subjects as trainees, three male and three female, all of them having no experience in martial arts training.

We designed separate tasks for the trainers and for the trainees. The task of the trainer was to design and produce the referenced gesture data model for 10 sets of five gestures each (punch, outside block, upper block, inside block, and down block). This model was later used by the trainee. Subsequently, each trainer was asked to perform 5 sets of 10 outside-blocks for testing gesture evaluation methods. The rest time between each set was two hours, and each set was repeatedly performed 10 times without resting.

For the trainees, we designed two basic learning conditions: *posture learning* and *gesture learning*. The task of posture learning was to learn the start and end postures of the five motions. The middle-gesture learning allowed for the practicing of individual gestures between the start posture and the end posture. In posture learning, the trainees were told to perform four postures three times each while watching a reference image without resting. We measured how long it

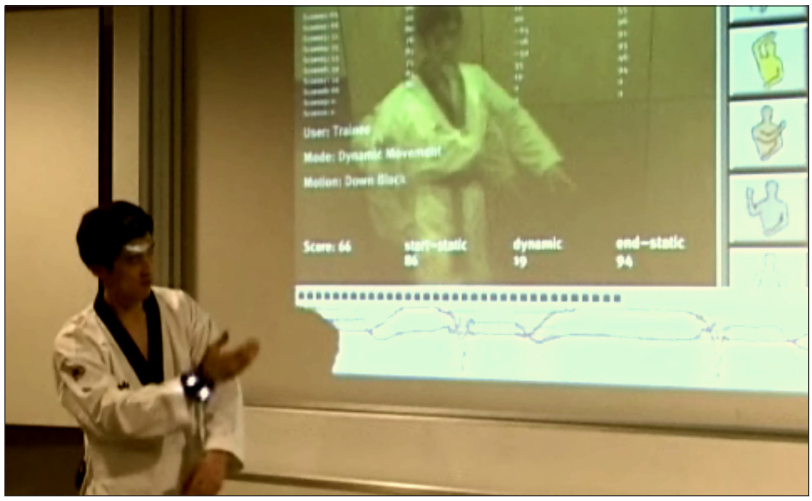


Figure 5.8: Gesture instruction system in action. The trainee wears body sensors on the wrist and performs a gesture watching the visual feedback on the wall display.

took to learn to match their postures to the trainer's average roll and pitch values. Among the five motions, we selected the punch which is relatively easy for teaching novices to use the system. The end posture of the punch is also employed as the initial posture for the trainee. We designed a simple progress bar which provides Boolean feedback to indicate whether the current posture is correct.

In gesture learning, we evaluate how trainees perform a single gesture in rapid succession of the start posture, the middle-gesture, and the end posture. Each trainee was told to do an outside block gesture. Before this, they had to learn the start and end postures of the block gesture. In our training scenario, we utilized the trainer's reference data to train the HMM. Thus, trainees first had to learn the required start and end postures so that their gesture could be detected. This usually took several minutes. This process is similar to the practical training and thus, it is not considered as an additional, unnecessary step to prepare the system. We evaluated each gesture based on the trainers' data from the previous experiment. From the first posture training experiment, we found that the outside block is the most difficult movement and is appropriate for using in the gesture training. Again, trainees were asked to perform 3 sets of 10 outside-blocks with approximately two hours between each set. All subjects were give time to become familiar with the new training devices, the wireless accelerometer worn on the right wrist, and the video projection.

5.2.4 Results

We collected the trainees' performance data for each of the tasks. For the trainer's task, we found that the trainer completed nearly all the tasks correctly. Thus, we could use the time to complete the task as an overall performance measure for

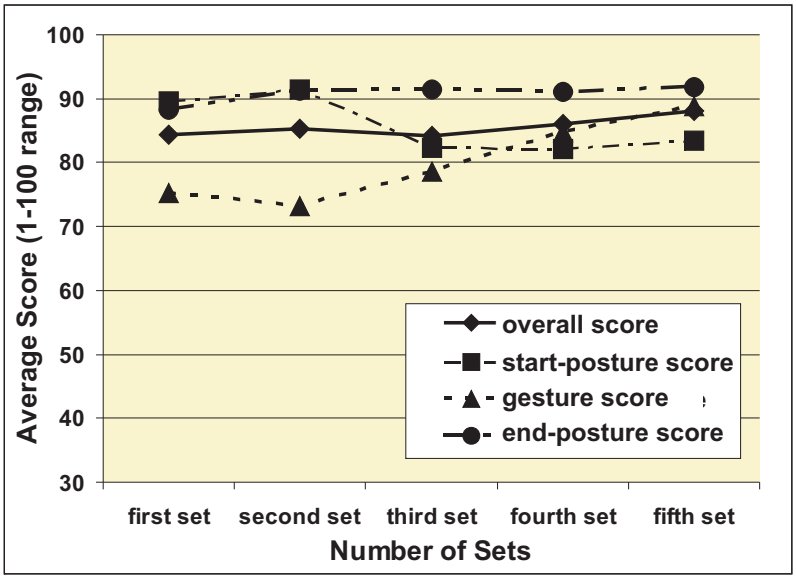


Figure 5.9: Experimental results of a trainer subject during gesture learning: average scores of two postures and one gesture and their overall score after five sets of 10 outside blocks.

potential trainers. It only took the trainer 10 minutes to create the 10 reference sets of the five motion data. From this, we obtained 50 motion training videos (10 for the five motions) containing both body and visual sensor data as well as visual feedback. The motion training video generation was performed very well with the help of automatic motion detection.

Figure 5.9 shows the average scores after the trainer's gesture learning. Each gesture performed was scored against three parts: start posture, gesture, and end posture. The overall scores were calculated by averaging the three scores. We can see how the trainer improved after 5 sets of 10 outside blocks each. While the scores of the end postures are constant, the scores of the start postures and the gestures change slightly over time, indicating the adaptation to the reference. After interviewing the trainer, we realized that the end posture scores slightly decrease, because the trainer focused more on the gesture and spent less effort on the postures. We also inferred that gesture learning bears more potential for further improvement than static posture. Even though the trainer masters the postures, it is very difficult to keep the right postures during the dynamic performance - a skill that distinguishes masters on the highest level. This shows that the resolution of our gesture evaluation is sufficiently high for evaluations on the highest level and that it can be used to practice gestures with self-created reference data.

The results of the posture and gesture learning clearly demonstrates that the system helps trainees learn complex martial art postures in a short time period. As illustrated in Figure 5.10, watching the body sensor signals helps trainees or trainers to find the correct postures. After this experiment, we compared the

individual postures and realized that some postures are relatively difficult to learn. We found that if the further away the sensors are from the trainee's body, the more difficult it is to repeat a constant posture.

The gesture learning experiment of the six subjects yielded interesting results as shown in Figure 5.11. Similar to the trainer's experiment, the static posture scores are higher on average than the gesture scores. Interestingly, we found that there were three different styles. Trainee 1 and trainee 2 focused on improving gestures. As a result their start postures got worse over time. On the other hand, trainee 4, trainee5 and trainee6 focused their attention on improving their static postures rather than on their dynamic gestures. Finally, trainee 3 improved both postures at the same time - which is clearly the desirable case. Even though the trainees knew the start and end postures, it was difficult for them to correctly perform motions in the dynamic setting. We also felt that the male subjects used more power in their movements, whereas the female subjects focused more on technique. However, this finding did not influence the results significantly.

During the experiments, we collected user reactions and received many comments. Some users felt that our evaluation methods could be useful for computer games related to sports and martial art sparring. They suggested that using real motions would make the interactions in playing computer games more appealing. Some participants got very involved in the training and took their performance seriously. As one participant commented: "The system helps me to focus my attention on precise my body movement." Most of people asked to use the system on a regular basis. Since we used low cost technologies, the system can easily be tailored to a personal training system. We also let some participants play with

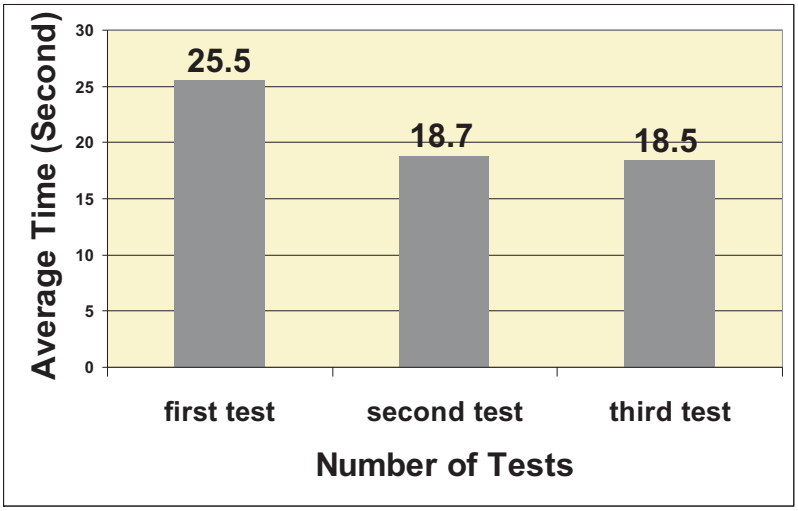


Figure 5.10: Experimental results of the trainee subjects in posture learning: average time in seconds to complete the posture learning task for four different motions. Note that the first and the last experiment use only a Boolean feedback to indicate whether the current posture is correct or not.

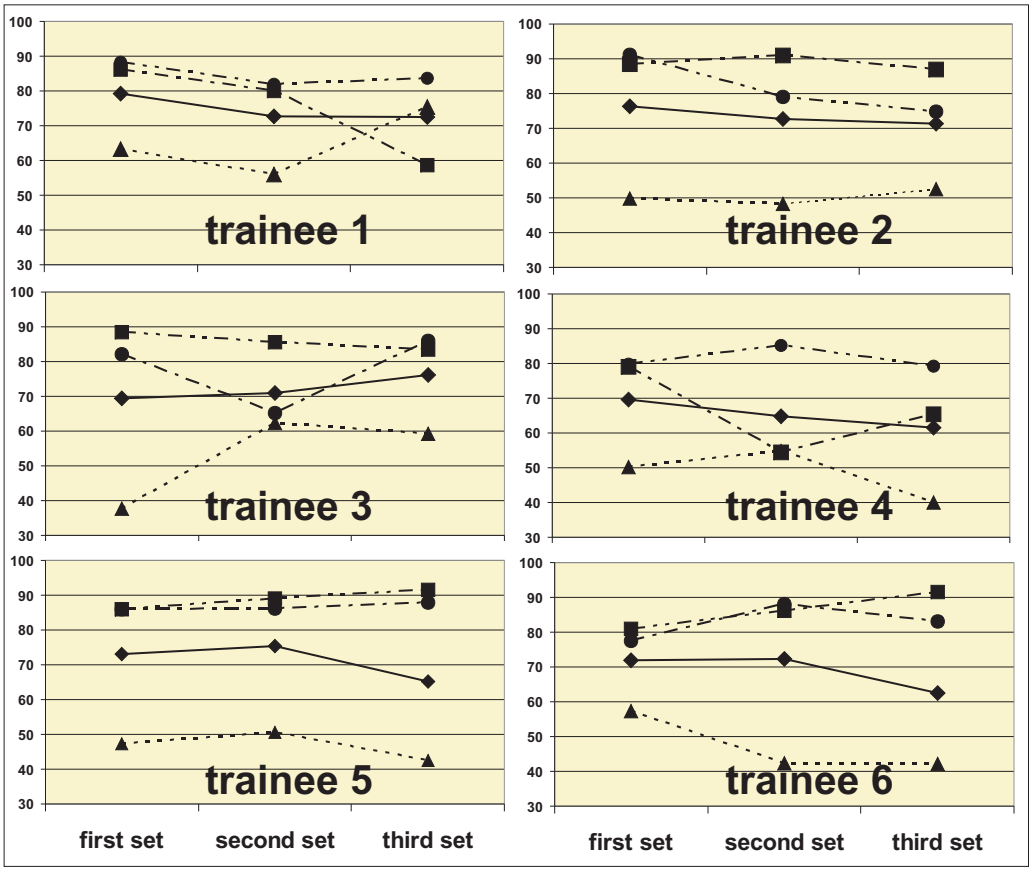


Figure 5.11: Experimental results of the trainee subjects for gesture learning: average scores of two postures and one gesture and their overall score after doing three sets of 10 outside blocks. The legend is the same as in Figure 5.9

the system due to their interests. In order to test long term training progress, we continued some experiments for a while. Although some users had lower initial scores compared to last time, they quickly caught up and made progress.

Chapter 6

A Spatial Context Aware Gesture Interface

In the previous chapters, we described the gesture acquisition and modeling used in our framework and presented the results of experimental evaluations. We show how the proposed framework can be used to develop gesture interfaces. This chapter presents a spatial context-aware gesture interface that improves the usability of gesture-based inputs by combining gesture information with additional context information.

There is a large amount of research dedicated to studing the use of context information in human computer interaction. Obviously, it is a far-reaching and difficult goal to define the general application of the context information to the design of gesture interfaces. Here, we focus on specific context information which is related to spatial objects of gestures (locations and objects of a gesture) easily featured together with 3D spatial gestures. We call this particular context information *spatial context*. Using spatial contexts, we aim to reduce the recognition error and improve the usability of gestures.

This chapter is structured as follows: after a short overview of the architecture of context model (Section 6.1), we present the spatial context model which supports registration and selection of spatial contexts (Section 6.2). Then Section 6.4.2 explains how a gestural command is recognized using the selected spatial contexts. Dynamic Bayesian Network (DBN) is developed to integrate the result of the gesture recognizer with other context information for command recognition. We developed two prototype applications which use 3D spatial gestures to control functions embedded with physical objects (Section 6.4).

6.1 Overview

Figure 6.1 shows the overall architecture of our spatial context-aware gesture interface for two major tasks: spatial context modeling and command interpretation. Acquired sensor data is forwarded from the gesture acquisition module.

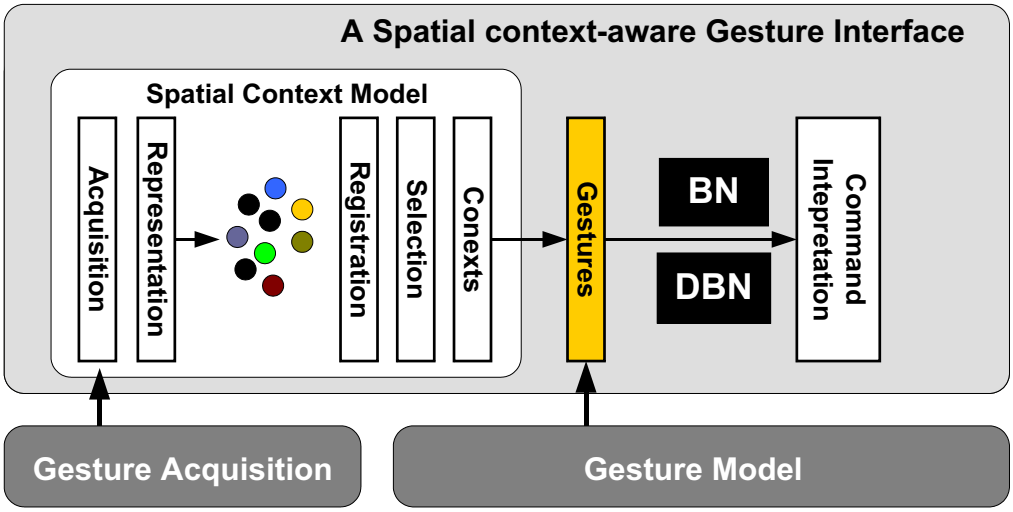


Figure 6.1: An overview of a context-aware 3D spatial gesture interface. Using the acquired sensors data, we represent the spatial context model with spatial objects such as gesture target and volume. Next, the spatial context is registered and selected based on this representation. To interpret gestural commands, the output of gesture recognition and the context selection are combined within the pre-defined BN or DBN.

The spatial context model handles the abstraction and registration of the context data using two types of spatial objects: gesture volume and target. The current status of spatial objects is traced with the performed gestures. The spatial context model provides an efficient way of extracting and retrieving the gesture volume and target. Therefore, system developers can register their own spatial objects and test different configurations in their applications. The command interpretation is accomplished by combining a list of the current context values and the recognized gestures using the gesture model.

6.2 Spatial Context Objects: Gesture Targets and Volumes

In general, the collection of implicit contextual information through available resources is a major challenge in the development of context-aware applications. For the application designer, it is important to decide what context information is relevant to the applications and then test it. The ultimate goal is to improve the usability of the application by optimizing the context information. We focus on the problem of defining the contextual information as it relats to specific gesture

information called the spatial information.

As mentioned in Section 2.1.2, spatial information is one of the four aspects of gesture as proposed by Hummels and Stappers [HS98a]. 3D spatial gestures usually relate to objects and locations. The relevance of objects and locations is a key idea in using context information for 3D spatial gesture interfaces. Based on this observation, we defined two types of spatial objects for 3D spatial gesture: *gesture targets* and *gesture volumes*.

Gesture targets are physical objects which users can select by moving a gesture input device close to an object or pointing it at an object. They can be physical or virtual objects. Gesture volumes are interaction zones where a set of gestures are planned for the application. Using gesture volumes and targets, we can define gesture candidates during gesture recognition. Therefore, we can minimize the computational complexity by only processing associated gestures within the current contexts.

For instance, kitchen space can be defined as gesture volume and the individual components in the kitchen, like a microwave and a refrigerator, can be defined as the gesture target. During the system design, application developers define gesture targets and volumes while considering their design policy about using gestures with the related locations and objects.

6.2.1 Registration

In our framework, spatial context objects are represented as a three-dimensional Gaussian blob with mean μ and covariance Σ as the center and shape of the object as shown in Figure 6.2. These Gaussian blobs are modeled and registered based on a series of 3D positions that users provide during the system setup.

In Figure 6.3, a user is registering the 3D position of an object. A user simply locates the device at the position of the object, and registers it by pressing one of the pressure buttons while holding the device on the object's position. The bright color LED of the device allows accurate tracking in various background conditions. We call this direct registration *touching*. Touching enables users to register the position of the object from the actual position of the object. However, this technique is only available when the object can be reached by the hand inside of the camera volume (i.e. the object should be visible on both camera images).

We use another technique called *pointing* as proposed in Wilson [WS03]. This pointing technique allows registration when the target objects are outside of the camera volume or when the users cannot reach the position of the target object. For instance, as shown in Figure 6.4, when a user wants to register objects on the wall that can't be captured with the current camera setup, then the user can use this pointing technique.

The main purpose is to find the object location by computing the intersection of the multiple pointing rays $\mathbf{w_i}$ from different locations $\mathbf{p_i}$. We can find the covariance of the object μ by solving the linear system of equations via least squares

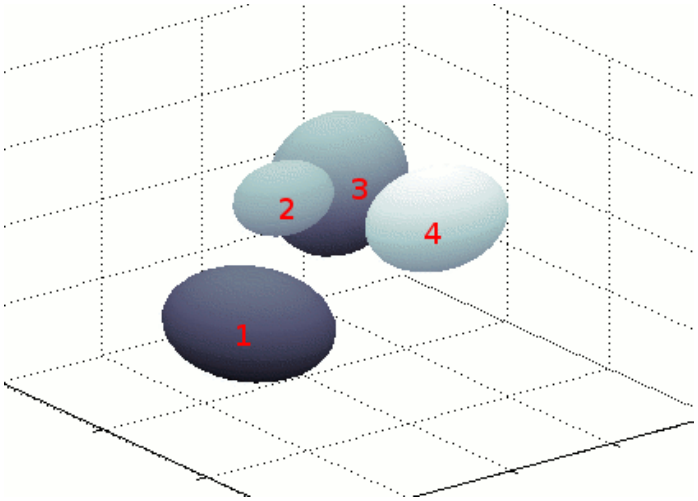


Figure 6.2: Representation of spatial objects. Each object is modeled as a 3D Gaussian distribution with mean μ and covariance matrix Σ with a center point c_j, a distance d_j and a variance σ_j^2.

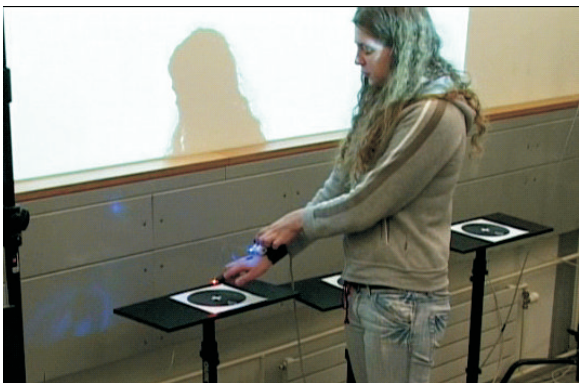


Figure 6.3: Registering an object using touching. A user is locating a gesture input device (mWire) at the location of the object and registering the positional data to the system by pressing the button of the device.

which can be represented as:

$$\mathbf{p_i} + s_i \mathbf{w_i} = \mu \tag{6.1}$$

where $\mathbf{p_i}$ is the device position and $\mathbf{w_i}$ is the pointing ray for the ith pointing observation, and the distances s_i to the object are unknown.

The covariance matrix Σ can be computed by summing up the spread of the differences between calculated target location μ and its estimates $\mathbf{p_i} + s_i \mathbf{d_i}$ and adding some minimal covariance Σ_0 i.e.,

$$\Sigma = \Sigma_0 + (\mathbf{p_i} + s_i \mathbf{d_i} - \mu)(\mathbf{p_i} + s_i \mathbf{d_i} - \mu)^T \tag{6.2}$$

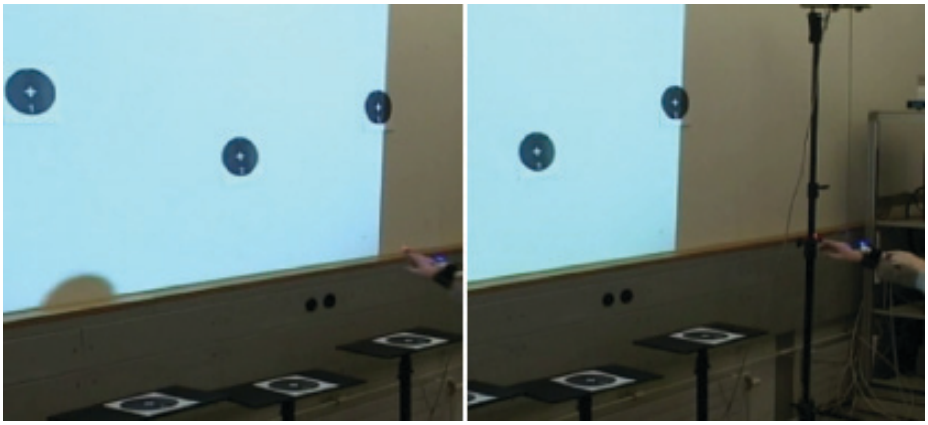


Figure 6.4: Registering a spatial object using pointing. A user is performing multiple pointing gestures to the same object from different locations.

The pointing direction should be accurate for increased pointing accuracy. Currently, we use the prototype input device mWire by connecting two LEDs to the mWire (one n the index finger and another on the wrist). Given these two 3D positions, we compute a pointing direction. The accuracy in pointing highly depends on the accuracy of the tracking of the two LED positions.

This method is improved using different input mechanisms. For instance, we can use a 6-DOF input device like a 3D mouse that can provide more accurate orientation information even though it is not a wireless device. In addition, as described in Section 3.2.2, we can derive the orientation of the device by combining the digital compass and accelerometers. This approach was used to develop mCube, a versatile gesture input device that will be explained in Section 7.2.

6.2.2 Selection

When the spatial context objects are registered to the system, we then consider the task of object selection. The basic idea is to find objects which are close to the place where users perform 3D spatial gestures. Since each gesture object is modeled as a 3D Gaussian blob as explained earlier, the result of selection is the probability computed from the geometrical relationship between an object and a 3D spatial gesture.

Similar to the registration, there are two different selection techniques: *touching* and *pointing* depending on whether the object is reached by the input device. As the name suggests, touching selects an object by bringing the input device near the object that the user wishes to interact with.

The system determines which object is closest to the position where the 3D spatial gesture is performed. Each object needs to be modeled as a 3D Gaussian blob with mean $\mu_{\mathbf{i}}$ and covariance Σ_i. A simple technique is to compute the likelihood l_i of selecting object i and evaluating the Gaussian distribution at the point. The

likelihood of pointing at target i is

$$l_i = g(\mathbf{p}, \mu_\mathbf{i}, \Sigma_i) \tag{6.3}$$

where $g(x, \mu, \Sigma)$ is the probability density function of the multivariate normal distribution, $\mathbf{p}$ is the 3D position of the device.

For pointing selection, we evaluate the Gaussian distribution at the point that is the same distance away as the hand is from the target and along the ray cast by the hand. The likelihood of pointing at an object i is given by:

$$l_i = g(\mathbf{p} + |\mu_i - \mathbf{p}|\mathbf{d}, \mu_\mathbf{i}, \Sigma_i) \tag{6.4}$$

where $\mathbf{p}$ is the position of the hand and $\mathbf{w}$ is the ray along the hand.

6.3 System Action Interpretation using BN

In the previous section, we defined spatial context objects (gesture volumes and targets) and described how they can be registered and selected based on the geometrical relationships between the gesture and various objects over various distances. Using this spatial information, we make a better interpretation of the gesture meaning in case it is ambiguous. Now, we explain our system action interpretation technique by combining the selected spatial objects with the gesture recognition results.

To integrate different information statistically, we can create Bayesian network (BNs) that provides a powerful tool for dealing with uncertainty. They enable efficient representation and manipulation of conditional probability distributions for a large number of variables [HGC94].

A Bayesian network is a directed acyclic graph that describes the dependencies among random variables. Each node of the graph represents a random variable. The directed edges define the conditional dependency relations among these variables. This graph structure allows modular representation of knowledge, as well as local, distributed algorithms for inference and learning and facilitates modeling using the intuitive and possibly causal interpretation.

Figure 6.5 illustrates the Bayesian network used to interpret system actions. The dependencies are modeled with discrete nodes between a system action, a gesture, spatial contexts, and additional contexts. Arrows between the nodes indicate the causal relationship between the action and attributes. The two main attributes are a type of the recognized gesture, and selected targets. Additional context information, time and system functions, can be integrated for minimizing misinterpretations.

In particular, our network uses temporal integration based on the concept of dynamic Bayes network (DBN), a special type of Bayesian networks for time-series events. Depending on the application scenario and user, there are a certain sequence of commands that the user must follow to complete tasks. Therefore, the

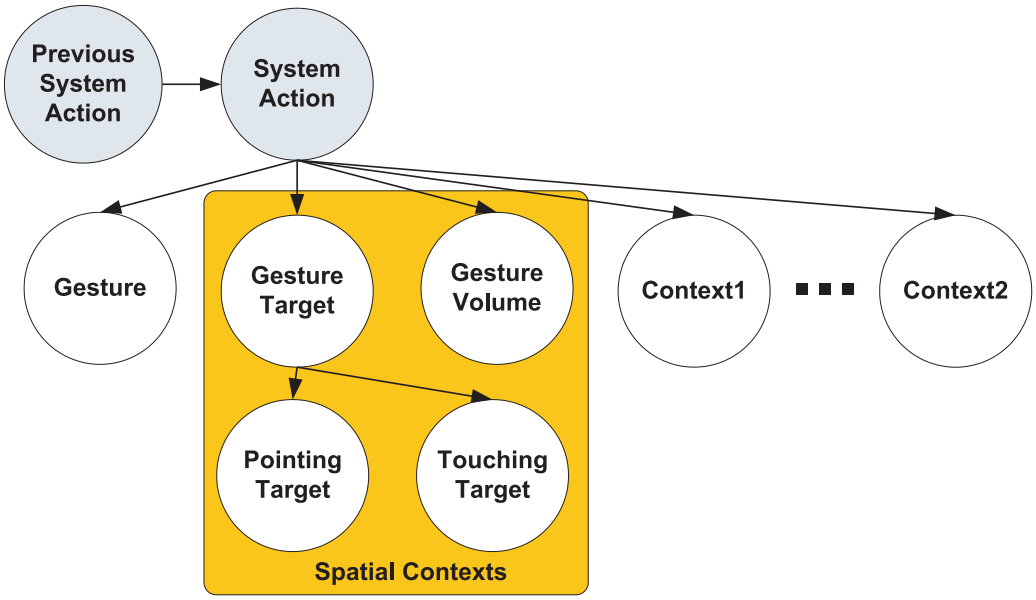


Figure 6.5: Topology of the DBN for command recognition. A topology is the predefined connecting of different nodes namely gesture classes, gesture targets and gesture volumes.

next command is partly predictable by the previous command history. For example, in a file-operation scenario, the probability of doing the past action becomes much higher when copy or cut is executed recently. Our recognition engine incorporates this prediction information to recognize the final command. The reliability of the gesture recognition engine is improved with this additional information.

For better understanding the use of network in gesture interface, we created make a simple example in a smart environment application. In the application scenario, the interpreted system action is equivalent to a certain command that runs on the system. For example the action "show weather forecast" can be executed as a command when a user is pointing at a window target and performing the *show* gesture. The result is used to execute the command that shows the weather information on the system's display screen.

When the user is pointing at the the light, the PointingTarget variable in the Bayes net is set to Light, for example. This causes the Command node to assign equal probability to the "TurnOnLight" and "TurnOffLight" variable settings, since these are the only admissible commands on lights. When the user performs "turn on", the gesture node is set to "TurnOn" and the distribution over the A node collapses to "TurnOnLight". The system then uses the appropriate command to turn on the light.

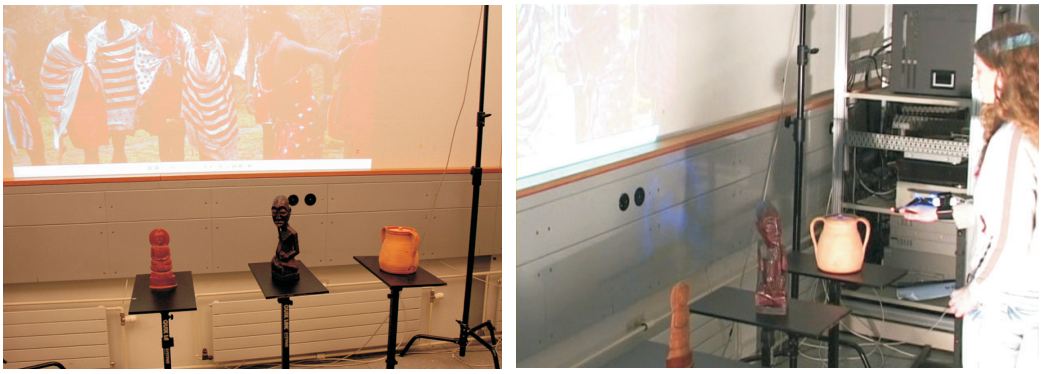


Figure 6.6: Gesture-based interactions in a smart museum environment. (Left) Exhibition items and visual information projected on the wall. (Right) Location registration of an exhibition item using mWire.

6.4 Prototype Applications

We developed two prototype applications based on the concept of smart environments [CD04]. We demonstrate the effectiveness of using spatial context information in reducing recognition error and disambiguating 3D spatial gesture performances.

6.4.1 A Smart Museum Environment

In this application, we developed a smart museum environment where users can interact with physical exhibition items to get related additional audio and visual digital information (Figure 6.6). For example, users can point at a certain item and listen to an audio description by performing a "turning on" gesture.

For gesture acquisition, we used two cameras installed in positions that efficiently cover the location and orientation of users. During the system setup, the location of each exhibit item is modeled as a 3D Gaussian blob. To train the Gaussian distribution of an item, a series of 3D positions of the mWire was collected when the pressure button was pressed at the items's location.

The system selects an exhibit item randomly when the hand is close to the item or is pointing at it. After successful gesture recognition, the system provides audio-visual feedbacks such as music or photos associated with the selected target item.

Figure 6.7 shows the prototype setup of the smart museum. There are four gesture targets: three (number 1, 2, 3) for exhibit items and and one for wall objects (number 4). There are two gesture volumes: one around the entrance area and one around the exhibition area.

The typical user audience is the public. It means that this application is used for a relatively short period. Therefore, we designed the gestures so that users can easily learn and perform them without errors. We used a total of 10 system

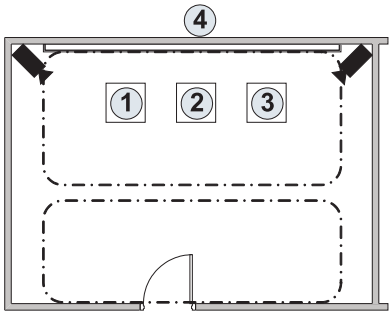


Figure 6.7: The setup of the smart museum environment. Number 1 -3 are th e targets for exhibit items and the target number 4 is on the wall. Two gesture volumes exist, one for the entrance area and one for the exhibit items respectively.

commands (turning on/off audio and video information), 4 gestures (on/off and up/down), and 4 gesture targets for the exhibit items, and 2 gesture volumes for the entrance and exhibition areas.

6.4.2 A Smart Home Environment

We designed a smart home environment where users can execute system commands with 3D spatial gestures. We show how the proposed context-aware gesture interface is applied to control home electronics (such as TV, audio system, and lights) in a living room environment.

Figure 6.8 shows the living room layout used in our application scenario. There are three gesture volumes, *entrance*, *sofa*, and *bed*, designed with the main functional areas of the living room environment. When the user enters the room then the entrance volume is activated, the sofa volume is used when the user is listening to music on the sofa. The bed volume is mainly used for sleeping at night. These examples clearly show how gesture volume is related to time and action.

We used five gesture targets: (1) ceiling light, (2) alarm clock by the bed, (3) coffee maker, (4) audio system, and (5) a bookshelf. They are located in the appropriate position for their use in the designed architectural plan. Since we do not have the actual devices, tagged paper boxes represented the devices and their operations were simulated on the computer.

Currently, there are 10 gesture based commands mainly for controlling the gesture targets such as switching on the lights and alarm clocks, and operating the audio system. These are registered as a gesture command which is a combination of a gesture and a target. These commands can be activated by pointing at or touching the gesture targets.

Different from the previous museum application, the typical user of this application is a private user and the system can be customized to a particular user. Therefore, we use gesture registration of our framework to prepare user-designed gesture sets. Typically when the system starts, users are asked to perform a *single*

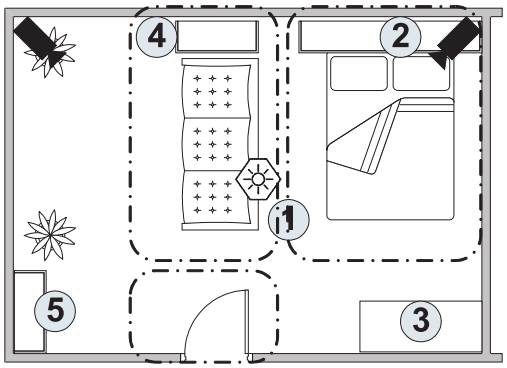


Figure 6.8: An experimental setup for a smart home environment. There are four gesture volumes (entrance volume, sofa volume, and bed volume) and five gesture targets (ceiling light, alarm clock, coffee maker, audio system, and a bookshelf).

gesture for each command to setup their own gesture templates.

Once this short initialization process is finished, the application recognizes the new input gestures by comparing them to the same-user gesture sets, and evaluate the performance in relation to reference gesture sets trained by another user for instruction. When the gesture is recognized, its associated command is executed. Obviously, if the user knows how to perform the required 3D spatial gestures, the gesture registration and evaluation process can be skipped. The DTW recognizer can be switched to the HMM recognizer if enough training data is available.

In addition to these spatial contexts, we used two other context information: typical using time and previous commands. For instance, the user is more likely to turn on the ceiling light at night, and the command to turn on the TV will logically be followed by turning off the TV.

Table 6.1 shows the suggested sequence of gesture commands using the time and gesture volume information. An example is when a person enters the room, first the user switches on the light followed the audio system. While various different cases could be be acquired, we applied a specific architectural knowledge to the various target users. Based on this information, conditional probability for the context information is estimated with the context usages and commands.

During the development of this application, we asked two subjects to use 3D spatial gestures to control the devices in the living room environment described above. They were asked to perform all tasks using the 3D spatial gestures with gesture targets and volumes. During these tests, every 3D spatial gesture performed was transcribed together with the recognition hypotheses, context information at the time of gesture performance and the user's intended command.

We evaluated the accuracy of the baseline system and compared with the one that does not use context information at all. Initially, there were eight gesture recognition errors, two target selection errors and four commands errors. By associating the gestures with spatial objects (gesture targets and volumes) and addi-

ID	Command Sequence	Used Time	Used Gesture Volume
1	4,1,6,8,10	morning (1)	bed (2)
2	7,2,5	morning (1)	door(1)
3	1,6,10	evening(2)	door(1)
4	9	evening(2)	bed (2)
5	7,3,2	evening(2)	sofa(3)

Table 6.1: A list of command sequences with the used time and the used gesture volume.

tional contexts (daytime and previous commands), we resolved gesture recognition errors.

The most prominent phenomenon was that the type of 3D spatial gestures could be misclassified in the gesture recognizer. However, we could eliminate these errors using the spatial context information that defines the relationship between the gestures and the gesture volumes and targets. For example, once the gesture target is selected, we could filter out some of gestures that are not related to the selected gesture target. First the evidences of all other nodes but the gesture node are computed and entered into the network. By inferring the gesture node, we eliminated the gestures with zero or very small probabilities and only compute the likelihood for the other gestures. We also checked the effects of each of the context used. In particular, we analyzed how command recognition errors are reduced using a particular set of contexts. Even though our experiments were for specific tasks, it shows how command interpretation improves by incorporating additional contexts.

Chapter 7

A Versatile Gesture Interface

While most gesture interfaces are created for specific tasks and displays, it is still an important issue to make a gesture interface more versatile. In this chapter, we present a versatile 3D spatial gesture interface system where users can use gestural interactions switching different applications and displays in a ubiquitous manner. We developed a unique gesture input device called the mCube. Users can use the device for desktop interactions by moving it on a planar surface, like a computer mouse. By lifting the device from the surface, users can seamlessly continue handheld interactions in the same application.

This chapter is organized as follows. Section 7.1 gives an overview of the proposed system. In Section 7.2, we propose the mCube device with a set of design principles for a versatile gesture input device. We also explain the system implementation and its core algorithms for computing orientation and the device's position. Section 7.3 demonstrates the device's interaction techniques which support a wide range of tasks, namely gesture commands, multi-dimensional manipulation and navigation, and tool selections on a pie-menu. Section 7.4 describes the experimental evaluation to test mCube's performance and its interaction techniques.

7.1 Overview

Figure 7.1 shows the architecture of a versatile gesture interface. There are two main components: a versatile input device and gesture inputs.

We developed a versatile input device called mCube using the concept of combining visual and body sensors. In particular, this device is designed for both desktop and hand-held interactions. The current prototype uses a cube shape, and includes accelerometers, digital compasses, buttons, an IR sensor, and a potentiometer. Sensor data is collected and processed with a microcontroller and transferred via Bluetooth wireless communication.

3D positional information of the mCube is obtained by means of an attached LED, which is captured by multiple cameras and processed using computer vision algorithms. The acquired gesture data from visual and body sensors are used for

robust recognition of a wide range of gestures as well as for multi-dimensional manipulation and navigation. The type of a gesture is identified by the gesture model and forwarded to the a versatile gesture interface for supporting various gesture based inputs.

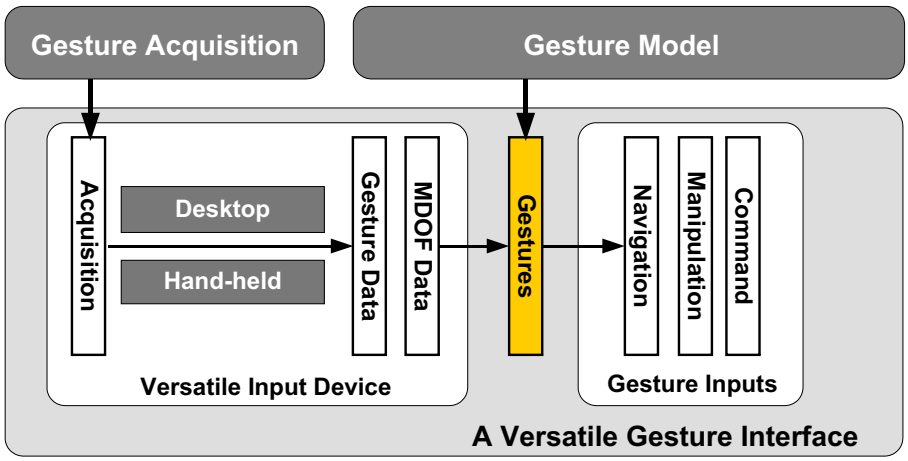


Figure 7.1: An overview of a versatile gesture interface. A versatile input device called mCube is used for gesture acquisition by providing gesture data and MDOF data. Taking the outputs of gesture model, the system provides various gesture inputs: navigation, manipulation, and commands.

Figure 7.2 shows the current prototype setup. During the acquisition, two firewire cameras are used to acquire the range of operation at 30 frames per second. The acquired images are processed to determine the 3D position of the mCube LED. Synchronously, the mCube acquires the data of the embedded sensors at a sample rate of 60 Hz and transfers the data to the host computer over the Bluetooth network. On the host, the sensor data is directly read from a communication port using the *java.comm* library. In the preprocessing stage, the acquired data amplitude is scaled using linear min-max scaling. The visual features are derived from the sensor data with appropriate up-sampling in order to match the frequency of the embedded sensor data and low-pass filtering to remove noise.

The 3D test application is developed based on the java3D API and provides dynamic 3D object creation, manipulation, and navigation using the mCube sensor data. We extract 2D position data by projecting the 3D data in a predefined plane and use it to control the low-level 2D cursor of the Windows operating system.

Figure 7.2: A prototype setup with a pair of video cameras, the mCube device, a computer with active Bluetooth wireless network, a desktop monitor, and a large display wall for wall projections.

7.2 mCube: An Input Device for a Versatile 3D Spatial Gesture Interface

This section describes a versatile gesture input device called mCube. The device is designed to acquire gestures combining visual and body sensor data. First, a set of design principles and solutions is presented used for the development of the device. Then we introduce our prototype device, called mCube with its hardware design and sensor configuration, and explain the computation of the rotational and positional information of the device.

7.2.1 Design Principles and Solutions

Support for Combined Desktop and Hand-held Interactions.

Ubiquitous computing environments can consist of computer controlled systems and multiple displays namely desktop monitors, table top displays, and wall displays. Considering potential spatial configurations of the user and displays, we categorize interactions into two groups (*desktop interaction* and *hand-held interaction*) based on whether the interaction takes place on a flat surface or in free space.

As one of the main principles, a versatile gesture input device should support both desktop and hand-held interactions. Moreover, the transition between them should be possible without additional operation requirements. Thus, users can instantly choose the best interaction for the current applications and displays. For this purpose, we use a cube form which has been widely used for the development of input devices in both interaction groups [PKE+06, RS00, FP00]. A cube form affords users to intuitively grab, move, and rotate both on a flat surface and in free space [SSL+03] as shown in Figure 7.3. For an automatic transition, the device

exhibits a IR reflective light sensor on the bottom so that it can automatically reconfigure with respect to contact with a surface.

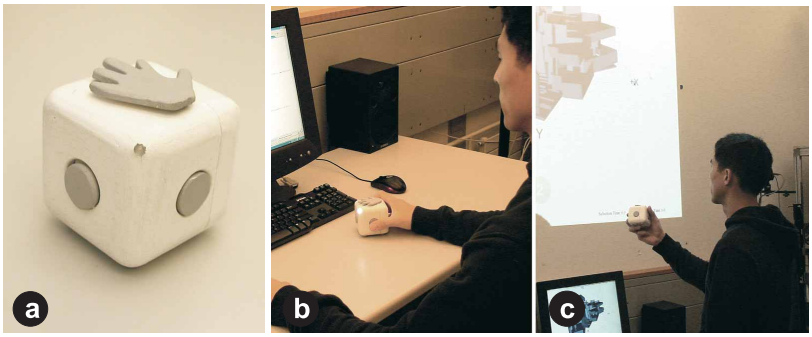


Figure 7.3: (a) The mCube prototype device for combined desktop and hand-held interaction. (b) Desktop interaction on a planar surface. (c) Hand-held interaction in the free-space.

Support for Wireless Operation.

A versatile gesture input device needs to be completely wireless so that the user can carry the device to another location and operate it freely in space. To achieve this goal, we track the 3D position of the device using cameras and computer vision technology.

The visual sensor approach relies on the accurate detection of relevant features which is a challenging task under varying illumination and environmental conditions. Moreover, it is not a-priori clear if the required tasks for a versatile gesture input device can be accomplished given their inherent level of inaccuracy. Due to these reasons, we use a bright color LED which provides focal brightness on the captured images, thereby achieving easier and more robust tracking.

Support for Multifunctionality.

A generic and versatile input device should support multi-functional operations so that the user does not have to change the input device between different tasks. We identify a set of required operations for a versatile gesture input device: gesture commands, navigation, manipulation of virtual objects, and tool selection.

The use of gesture commands has been motivated for various purposes such as appliance control in smart environments [WS03] and game control [PKE+06, Wii]. A gesture input device is required to provide enough features to discriminate between different gesture commands. For this, we combine visual sensors and embedded sensors with a proper feature extraction methods.

We compute 6-DOF information combining the visual and embedded sensor data so that the device can be used to manipulate and navigate virtual objects.

Manipulating virtual objects requires a certain degree of precision, which can be challenging to achieve with a vision-based approach. We facilitate this problem by using a bright color LED. Finally, for efficient tool selection, we allow the user to control a pie-menu by rotating a top handle attached to the cube.

Support for Design Variations and User Definable Ergonomics.

These days, many commercial electronics such as MP3 players and mobile phones are designed considering the user's desire to change the appearance and ergonomics of the device. We also consider these issues as a principle of a versatile gesture input device.

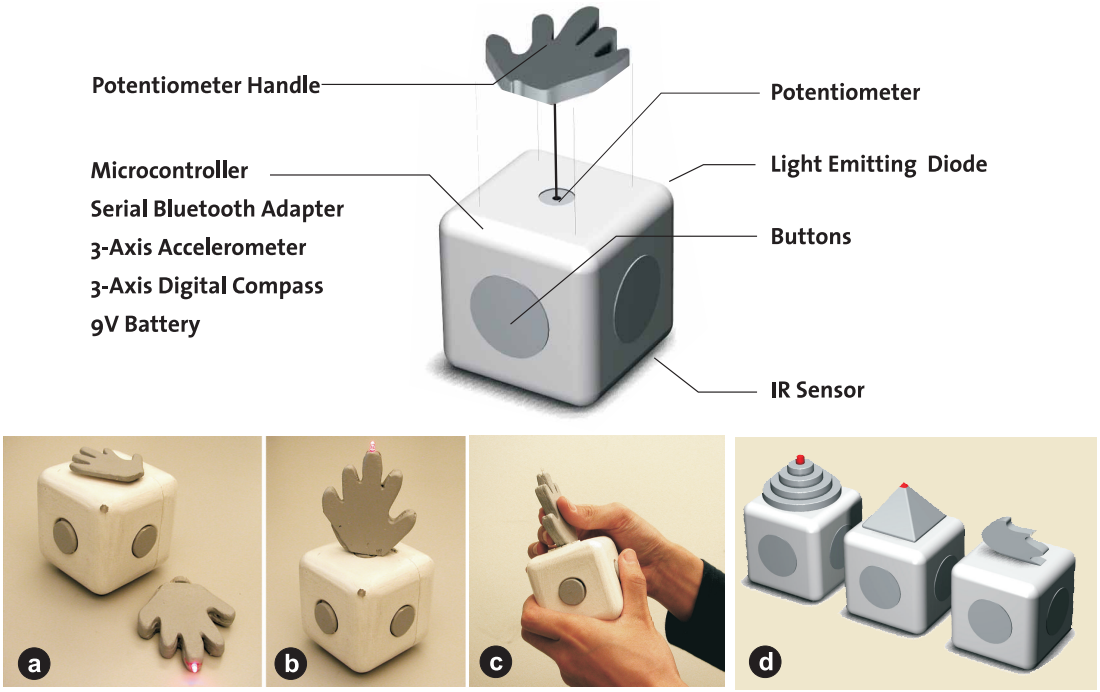


Figure 7.4: (Top) Hardware configuration of the mCube. The device includes multiple sensors such as accelerometers, digital compasses, four buttons, a rotary potentiometer, an infrared (IR) distance sensor, and one color LED. In addition, it contains a micro-controller for acquisition, a Bluetooth transmitter, and a 9V battery power supply. (Bottom) The top handle can be easily replaced with other designs according to the user's preference. (a) A flat style top handle using the hand metaphor. (b) A vertical style top handle using hand metaphor. (c) The operation of the vertical style top handle with both hands. (d) Alternatives top handle designs.

7.2.2 mCube Hardware Design and Sensor Configuration

Based on the proposed design principles and solutions, we developed the prototype device mCube. The current version is a prototype that can easily be miniaturized for production. The mCube consists of two units: a *body* and a *top-handle*. The body is a cube with an edge length of 6 centimeters as shown in Figure 7.4. For the top handle, various designs are provided so that users can choose according to their preference for design, application, and ergonomics. For instance, by attaching another top handle in an upright position (Figure 7.4b and c), the mCube can be easily re-configured for two-handed interaction.

The mCube is designed to sense various input events (button clicks, handle rotation, and surface contact) and the physical manipulation of the device (translation, rotation and tilt). We labeled the six sides of the body with top, bottom, front, back, left, and right. In summary, the mCube body is equipped with the following sensors and features. The images of some components are shown in Figure 7.5.

- *Buttons.* We attach a button to each side face of the body so that the user can intuitively use them, mapping the button/handle direction to the user direction.
- *Potentiometer.* A potentiometer is located under the top side, and senses the rotation angle of the top handle which is connected to the mCube body through a rotary shaft.
- *IR Distance Sensor.* On the bottom side, an IR reflective light sensor is located to sense the distance to the planar surface from 0 to about 3 centimeters. It emits a small beam of invisible infrared light and measures the amount of light that is reflected back to the sensor. Since the sensor is sensitive to ambient light, the sensor is shielded.
- *Light Emitting Diode.* One color LED is attached to a corner of the mCube body to provide a bright color spot in the acquired camera images at a wide range. As illustrated in Figure 7.4, a LED can be installed to top-handles. In this case, the LED on the cube can be disabled using a LED switch located on the bottom side.
- *3-Axis Accelerometer.* Acceleration is measured with two Memsic 2125 2-axis accelerometers attached in orthogonal configuration measuring dynamic acceleration (vibration) and static acceleration (gravity) with a range of $\pm 2g$ at a resolution higher than $0.001g$.
- *3-Axis Digital Compass.* Two Hitachi HM55B 2-axis digital compasses are integrated perpendicularly to provide information on the Earth's magnetic field in three dimensions at 6-bit (64-direction) resolution.
- *Micro-controller.* A Parallax Javelin Stamp micro-controller with 32k of RAM/program memory is used to read sensor data using delta-sigma A/D conversion. A Javelin Stamp is programmed in the Java programming language and it is based in the Ubicom micro-controllers.

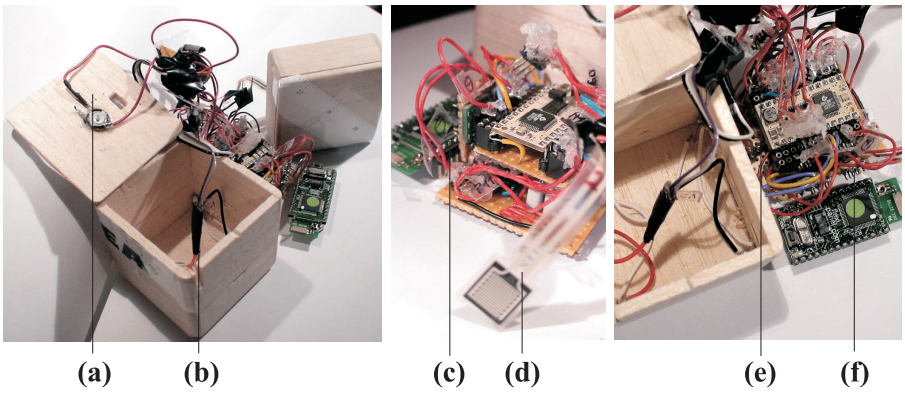


Figure 7.5: Hardware components of the mCube. (a) potentiometer, (b) battery, (c) bluetooth (d) micro-controller (e) accelerometers, and (f) digital compasses.

- *Bluetooh.* The raw sensor values are then transmitted wirelessly from the device to the host computer using a F2M01C1 Bluetooth module offering a nominal range of approximately 100m.
- *Battery.* The system operates on 9V.

7.3 Interaction Techniques

In this section, we describe an exploratory set of techniques intended to investigate the design space of mCube interactions. Some of these techniques can be selected and optimized for any application that seeks to use the mCube.

A versatile gesture interface is not intended as a universally best interface for any application but a more efficient interface that supports other necessary functionalities as well as efficient gesture interactions. It is intended to improve the work flow of interactions minimizing the use and learning of other input devices. mCube users can use 3D spatial gestures for various interaction techniques (e.g. gesture commands, virtual object manipulation and navigation, and tool selections). Particularly, users can quickly switch between desktop positioning and hand-held positioning, yielding a high degree of freedom for appropriate interactions in different display platforms.

7.3.1 Switching Between Desktop and Hand-held Interaction

As described earlier, the mCube uses the affordances of cube shape which can be operated on a flat surface and in free space. To facilitate this operation, we program the mCube to automatically recognize the current interaction mode based on

the contact between the device and the surface of the table using the embedded IR sensor. If the value is lower than a desktop threshold (1 cm) for a certain amount time (two seconds), the interaction mode is changed to desktop interaction.

Using this feature, users can optimize the interaction for the required task in the current application and display setup. For instance, for manipulation tasks requiring more precise control, users can use desktop interactions minimizing the hand tremor which can be caused by the lack of fixed support in hand-held interaction. Alternatively, users can choose hand-held interactions to perform direct 3D inputs and operations in free space.

7.3.2 Top Handle-based Mode/Tool Selection

The mCube allows users to switch between different modes or select a tool on a pie-menu by simply rotating the top handle. This movement is similar to the highly intuitive operation of using a pepper grinder. During this operation, we hold the side and rotate the top part without any substantial previous learning. The main benefit is that users can use this handle during other operations. For instance, during navigation or manipulation of virtual objects, users can choose different tools such as rendering modes (wireframe or shaded rendering) or texture and color styles of the model without interfering with the positional control of the device.

We use the top-handle to define the device mode. We defined three necessary modes: an *idle* mode for power saving, a *recognition* mode for gesture recognition and a *pie-menu* mode for selecting different tools. Figure 7.6-b illustrates an example of a pie-menu with twelve icons. During rotation of the top handle, the widget is displayed and the designated sub menu is highlighted depending on the direction of the handle as shown in Figure 7.6. Different numbers of icons can be used by dividing the circular region of the top handle into a corresponding number

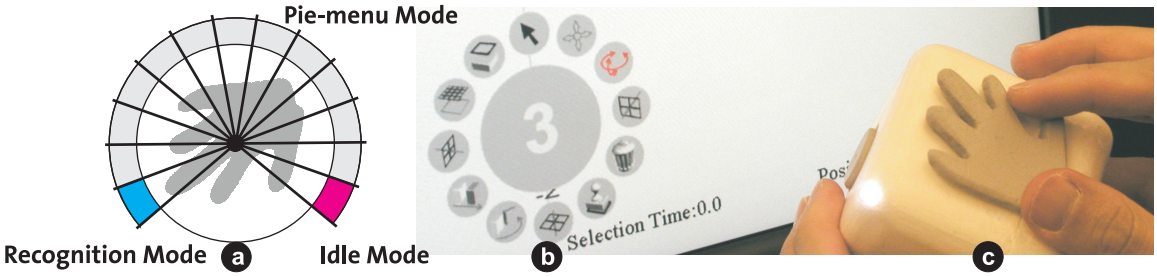


Figure 7.6: (a) Three modes of the device controlled with the top handle: recognition mode, pie-menu mode, and idle mode. (b, c) In the pie-menu mode, users can select a tool with a modified pie menu by rotating the top-handle. The circular region of the pie-menu is divided into 12 regions for 12 icons of the pie-menu. The number 3 in the center of the pie menu indicates that hand-held interaction is currently activated.

of sections.

7.3.3 Examples of mCube Gestures for Command Inputs

We designed a multipurpose set of gestures with the mCube exploring the intuitiveness and interoperability of the mCube in desktop and hand-held applications.

Gestures for Desktop Interactions.

For desktop interactions, we designed a set of gestures which are suitable using a *glass metaphor*: pouring water in three directions (front, left, and right), twirling and rotating in Counter-Clockwise (CCW) and Clockwise (CW) as illustrated in Figure 7.7-a. These gestures can be easily learned because of their familiarity and intuitiveness from the operation of a real glass. To perform the glass gestures, users place the mCube on a surface and then perform the gestures by lifting the device from the surface. This simple initialization motion greatly disambiguates the recognition based on the IR-distance value for each gesture.

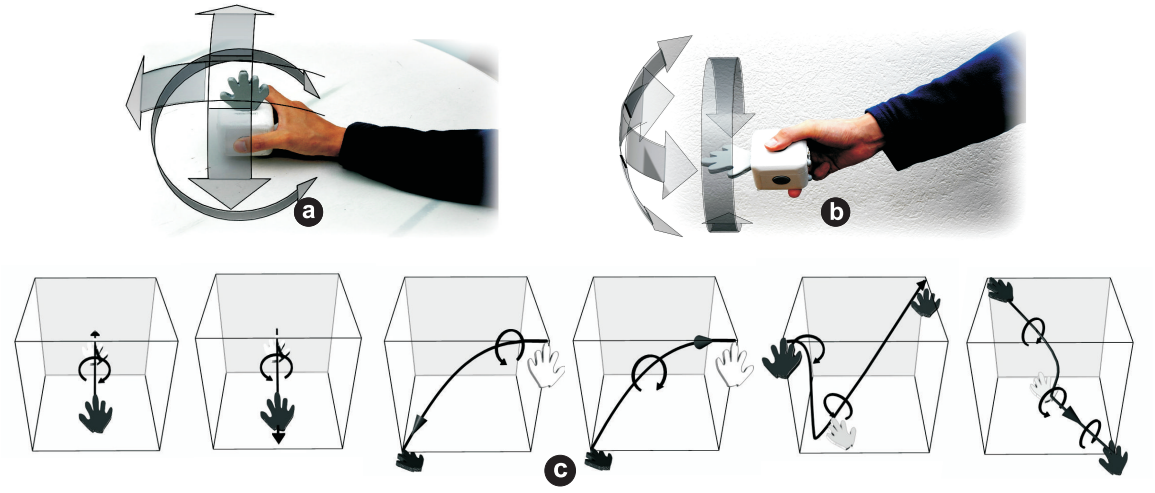


Figure 7.7: (a) Examples of desktop gestures. A user is performing gestures on the desktop surface: pouring-left, -right, -front, and rotating-CW, -CCW. (b) Examples of hand-held gestures. A user is holding the device in the air and performing gestures: pointing-up, -down, -right, -left, and rotating-CW, -CCW. (c) The 3D spatial gesture examples with a box style 3D gesture volume. The line indicates the trajectory of the gesture and the end of the gesture is presented as an arrow. The hand symbol indicates the direction and rotation of the mCube using black for a palm-down position and white for a palm-up position. The 3D gestures are increasing in complexity from left to right.

Gestures for Hand-held Interactions.

One of the common tasks of hand-held interactions is selecting a physical or virtual object using a pointing gesture and controlling it with subsequent gestures [WS03]. For this purpose, we designed a set of hand-held gestures targeting the commonly used actions on appliances (rotating the handle to turn an item on/off and changing the volume up/down.

In addition to the previous simple gestures, more complex 3D spatial gestures can be performed using the mCube. While simple gestures can be recognized with a heuristic approach, which looks for simple trends or peaks in one or more of the sensor values [WS03], for complex gestures we need advanced pattern matching techniques. Various recognition algorithms are proposed using statistical pattern matching techniques [CB96], and multiple sensors are combined to provide better discriminating gesture features. Even though their explanation is considered outside the scope of this paper, it is assumed that the combined gesture features of mCube will improve the recognition and enable a larger gesture space. It is also important to consider human variability exhibited in 3D spatial gestures due to the difference in user performance [PKE+06]. During our developments, we found that the illustration of 3D spatial gestures plays an important role in minimizing the human variability and improving the recognition rates. Figure 7.7c illustrates examples of our 3D gesture diagrams for the mCube device.

7.3.4 Multi-dimensional Manipulation and Navigation

Virtual Object Manipulation.

In desktop interaction, the object is controlled in the same way as a computer mouse. We implemented a similar implicit clutching using the IR distance sensor. For instance, when the user lifts up the device, the movements of the device do not affect manipulation. Users can also rotate the virtual object by physically rotating the device which is not possible using a conventional computer mouse as shown in Figure 7.8.

During hand-held interaction, the device is operated in space mapping the horizontal and vertical movement of the device to the corresponding object position. Therefore, users can directly move, rotate and tilt the object in any direction for 3D manipulation (Figure 7.8d). An explicit clutching technique is used with the right button of the mCube similar to other commercial 3D input devices.

Virtual Space Navigation.

Navigation is an important aspect of interaction with a Virtual Environment (VE). Users of VEs need to understand the space that surrounds them and find their way around. Unfortunately, the interaction methods to use a computer mouse is not optimal because users have to understand the mapping between the 2D

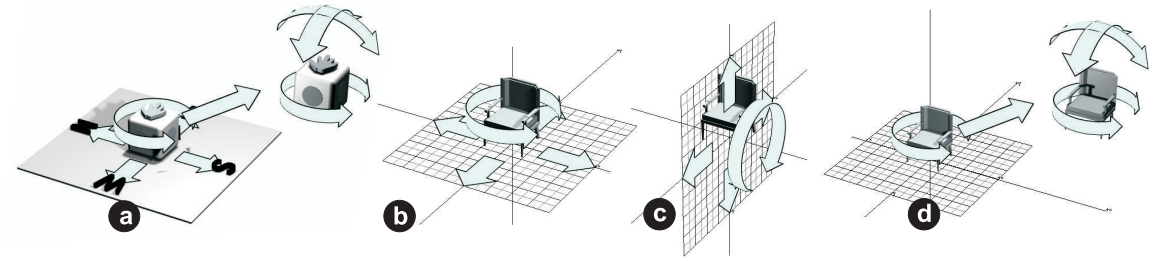


Figure 7.8: Examples of virtual object manipulation: (a) The mCube is operated on the desktop surface and in space. (b, c) In desktop interaction, the object is translated and rotated on a working plane e.g., $x-y$ plane and $x-z$ plane. (d) In hand-held interaction, the object is operated with additional dimensions in space.

control of a computer and navigation techniques (e.g. changing view orientation and directional movements). Navigation problems become even more serious in large scale VEs like a CAVE that does not support the use of a computer mouse.

To facilitate more efficient navigation, we use the four buttons as well as the rotational control of the device. The front, back, left and right buttons are used to control the movement of cameras. This idea is inspired by the *Cubic Mouse* which successfully utilizes a cube shape as an intuitive physical proxy of virtual objects in navigation purposes [FP00].

As illustrated in Figure 7.9, we implemented two navigation schemes: *examine viewer* to control a 3D virtual object like a 3D trackball and *walk viewer* to navigate through 3D virtual space with a walking metaphor. In the examine viewer, the rotational control of the mCube is used to rotate a virtual trackball. The virtual camera is located outside of the model pointing to the center of the model. The rotational movement of the device is mapped to a virtual camera angle. The front and back buttons are used for zooming in and out, respectively. In the walk viewer, users can look in a certain direction while moving in another direction similar to the walking navigation in real world.

3D Pointing Interaction

With the position and orientation of the mCube, we now consider the pointing interaction. Pointing interaction is an important technique used in various gesture-based applications. For instance, VR applications use the pointing interaction to navigate virtual space and manipulate objects. Smart environment applications use the 3D pointing to select a reference object to be controlled by a gesture. The mCube supports 3D pointing based on the 6-DOF information computed from the combination of visual and body sensor data.

During our development, we tested the performance of mCube for selecting objects located in a physical environment. To observe the performance of pointing in real time, we developed a test-bed application. When the user is pointing a

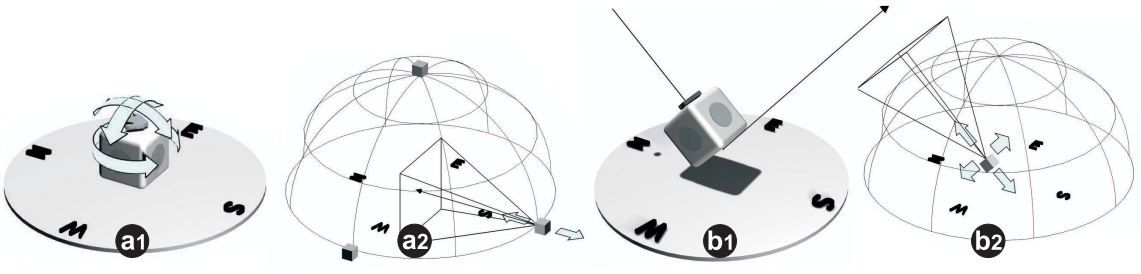


Figure 7.9: Examples of virtual space navigation: examine viewer and walk viewer. (a) In the examine viewer, the virtual camera is rotated around the scene pointing to the center while manipulating the mCube. To zoom in and out, the front and back buttons are used respectively. (b) In the walk viewer, the virtual camera is moved and oriented based on button clicks and the orientation of the mCube.

certain direction in the physical space, the system shows the virtual scene of the physical environment as viewed from the pointing direction. Figure 7.10 shows the four main objects labeled with window, monitor, book-shelf, and entrance on the virtual model used in the test. It also shows the virtual and real images of the environment from approximately the same view point.

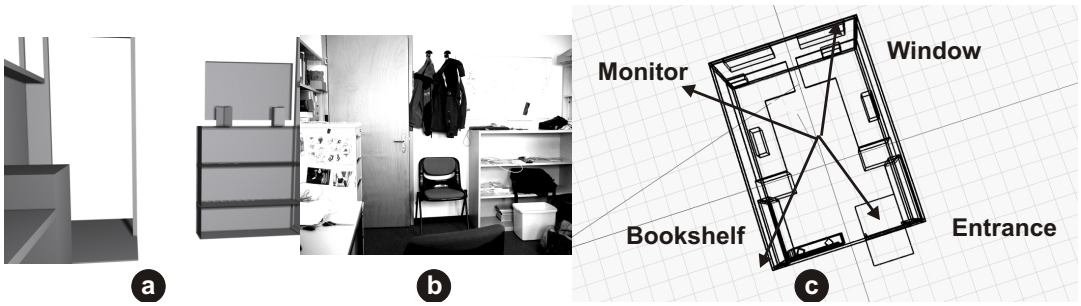


Figure 7.10: A digital representation of a physical space. (a) a 3-d graphic image of a virtual office and (b) a photo image of a real office. (c) Four main objects used for testing the pointing interaction: window, monitor, bookshelf, and entrance.

The coordinate system of the virtual model and physical model should be matched so that we can see the right rendering scene when pointing in a certain direction in the physical environment. Figure 7.11 shows the user performing 3D pointing standing in the middle of the physical environment. As the user changes the 3D pointing direction, the system shows the relevant virtual scene on the monitor.

7.4 Experimental Evaluation

The goal of this evaluation was to assess the performance of the gesture input device mCube and its interaction techniques. We analyzed design factors that

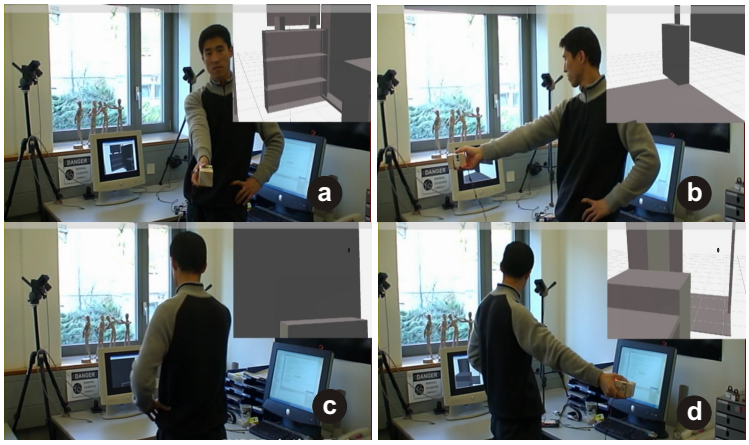


Figure 7.11: Demonstration of spatial pointing interaction. A user is performing pointing gestures and each image shows the four main objects of the office : (a) bookshelf (b) monitor (c) window (d) entrance. Each image includes the relevant virtual office rendered image.

influence human performance including perceived exertion and movement characteristics of the input device.

To explicitly test the user capabilities of desktop and hand-held interactions and switching between them, we performed a simple 3D docking task. We observed completion times and examined how the performance of the subjects improved with repeated experiments and experience. We expected lower performance over time due to user fatigue.

7.4.1 Process

We had six participants with strong backgrounds in computer science and computer graphics taking part in our experiment. All subjects were experienced in controlling virtual objects in 3D space with a computer mouse. Half of them already had experience with virtual reality systems and 3D interaction devices. Each participant received a demo of the system and interaction techniques. They then practice all the system functions. All users understood the system after a short practice (about 10 minutes).

As illustrated in Figure 7.12, the subjects were seated in front of a 21 inch monitor holding the mCube. The testing application consisted of two virtual cubes and two grid planes. The solid-rendered cube is an object to be manipulated. The wire-frame cube is the target object placed at the diagonally opposite corner. There are two tasks: selection and positioning. In the selection task, users have to select the object by manipulating the screen cursor. In the positioning task, users transform the selected (rendered) object to the wire-frame object using the 3D interaction mode of the mCube. When the object is transformed, shadows of both

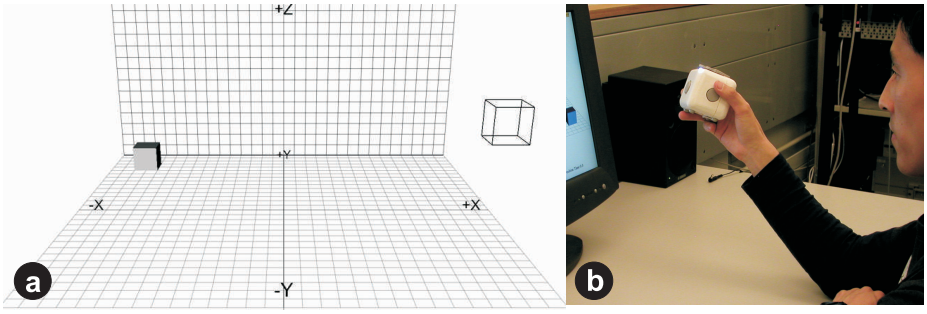


Figure 7.12: Visual stimuli for the usability testing and the experiment setup. (a) Visual stimuli consisting of two virtual cubes and two grid planes, (b) a subject is performing a task with the mCube sitting in front of a 21 inch monitor.

objects are projected onto the two grid planes to provide additional depth cues. Grid snapping is applied to the movement.

Before starting the test run, the users are introduced to the functionality of the mCube and the 3D application. Each subject was asked to repeatedly perform several recurrences of the same task to locate one cube with another cube located at different positions in 3D space. Users were asked to alternate between desktop and hand-held interaction modes by either putting the mCube on the table (desktop mode) or holding it in the users hand (hand-held mode).

7.4.2 Results

As illustrated in Figure 7.13, the subjects significantly improved over the experimental recurrences for selecting the object. Since the mCube supports desktop and hand-held interaction, the users were able to choose the one at will in their tasks. This means that users can easily return to desktop interaction by putting the mCube onto the table and lifting it up for further hand-held interaction.

In the positioning task, in the hand-held interaction, we clearly noticed that fine positioning was hard to accomplish even with the intuitive 3D operation in the air. Almost 50% of the task completion time was taken for the final placement. These problems are mainly due to trembling of the user's hand and further caused by the well-known fatigue problem in hand-held input devices [Zha98]. As compared to the experments with the desktop interaction, all participants were tired because they had to keep their arm extended in front of them.

Some users pointed out that moving the 2D screen cursor in hand-held interaction is even more intuitive than desktop interaction because the horizontal and vertical cursor movements are mapped to the same directional movements as the physical device. We think that hand-held interaction has some potential for novice users to perform screen cursor control easily. This might also be an interesting issue in the extended user testing.

After the 3D positioning experiment, we asked the subjects to try other mCube

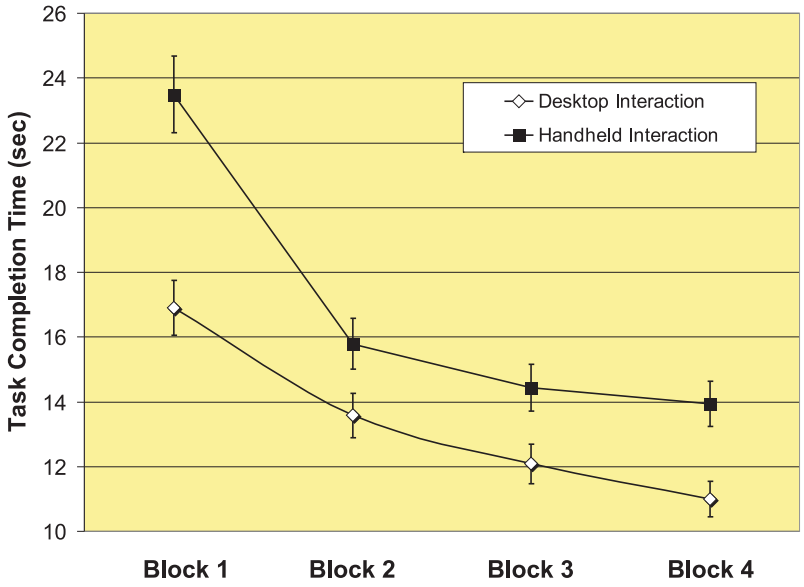


Figure 7.13: Mean task completion time for the selection task over the course of four experimental repetitions (with 95% confidence error bars).

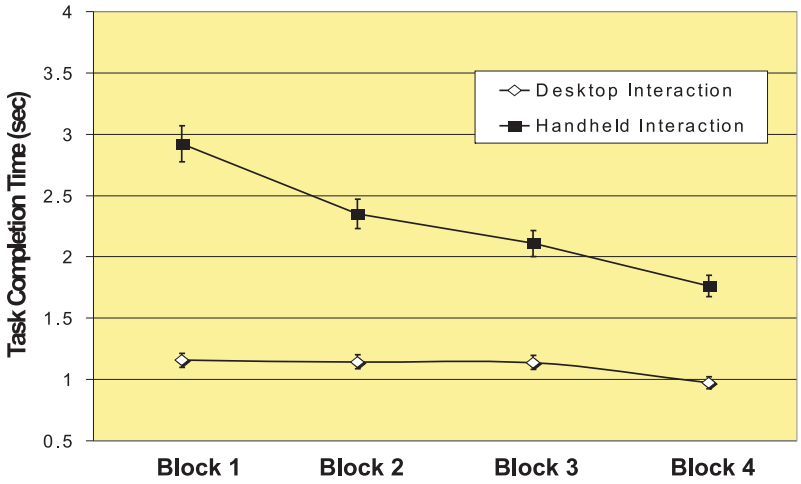


Figure 7.14: Mean task completion time for the positioning task over the course of four experimental repetitions (with 95% confidence error bars).

interaction techniques for informal evaluation. In the previous experiment, we mainly assessed the device in terms of speed and accuracy for screen cursor controlling and 3D object positioning. In this evaluation, the main purpose is to

evaluate the performance of gesture commands, the handle-based tool selection and MDOF control including virtual object and camera controlling. Through the whole evaluation processes, we received valuable feedback which we will consider for the next prototype.

For the gesture commands, all users had no difficulty in learning the pre-defined simple gestures (glass gestures and pointing gestures) within after few tries. For the user-definable 3D gestures, we used the DTW-based gesture recognizer which does not require much training data due to the time limitation. First, users were asked to perform a *single* gesture for each command to setup their own gesture templates. Once this short initialization process is finished, new input gestures are recognized by comparing the same-user gesture sets.

All pre-defined gestures were well recognized. When performing 3D spatial gestures, careful attention is required to perform the same gestures as the registered gesture templates. After a few attempts, the users began to remember the movement characteristics of the gesture and consequently perform it more consistently. We also found out that each user has different approaches and preferences to creating and performing 3D spatial gestures. For instance, others prefer short and simple gestures and some users prefer longer and more complex gestures with their own style.

The MDOF control used in the manipulation of virtual objects and cameras was well received due to the adaptation of similar physical manipulation skills. Users stated that the 3D rotational operation of the mCube is more intuitive and considered to be more efficient than the corresponding computer mouse functionality. We believe one of the reasons for this is that the current rotation technique with the mouse always requires an unintuitive transformation of 2D interaction to 3D information. If a user rotates along arbitrary axes, it is even more difficult to understand the rotation axis and angle. Using the mCube they were able to rotate the objects by physically rotating the device along arbitrary axes.

The top handle was easily understood and performed well. All participants felt comfortable rotating the top handle in both desktop and hand-held interactions. They could select a tool on the pie-menu during other operations such as navigation and manipulation. Specially in a large screen display, using the pie-menu with the top handle, users can minimize the use of cursors which might be difficult to precisely control therein. Some users stated that they could remember the item locations on the pie-menu and rotate the top handle even without looking at the device. We expect that this can be a powerful mechanism to minimize the cognitive loads during interaction and may be a topic for future user experiments.

Users stated that they felt uncomfortable with the cube shape when only one hand is used for continuous 3D input. This feedback shows that careful ergonomic studies are required for designing the shape of the next prototype. In addition to the form factor, the weight of the device was considered heavy especially when users performed long 3D interactions only. The current weight of the device is 140 gram. We believe that a reduction in the devices's weight can occur by using a plastic case, and a lighter battery pack. Two users stated that the speed of the

device is too slow and that it is hard to select objects due to the shaking of the hand.

Chapter 8

Conclusion

This chapter concludes the dissertation with a brief summary and a review of the main contributions. We categorizes the contributions in terms of the three proposed design components of 3D spatial gesture interfaces: input devices, gestures, and contexts. We also discuss possible directions for future research.

8.1 Summary

In this dissertation, we presented a novel framework to support the development of 3D spatial gesture interfaces in terms of gesture acquisition and modeling.

For the acquisition, the framework captures 3D spatial gestures via heterogeneous sensors. The main idea is to combine visual and body sensor technologies, thereby acquiring more robust gesture features and improve the recognition rate of gesture recognition. Several novel gesture input devices were developed to support users facilitating different types of sensors.

For the gesture modeling, we introduced the motion chunk, a gesture unit that stores continuous complex human gestures with a sequential combination of chunks. Based on the structure of motion chunks, the framework supports three major functionalities: gesture registration, recognition, and evaluation. The principles behind the framework have been demonstrated and validated by a series of experiments. In the gesture recognition task, we tested the use of visual and body features and compared the performance of two gesture recognizers. We also conducted usability evaluations to learn the 3D spatial gestures.

The results of our experimental evaluations contribute knowledge toward advancing gesture research in HCI. 3D spatial gesture should be captured with enough information so that various types of gestures can be designed from simple to complex. Our approach to combine visual and body sensors could improve the recognition rates under various conditions. Two pattern matching techniques (DTW and HMM) are used alternatively based on the number of training data. DTW enables the use of gestures right after creation without requiring a large training data. HMM can support more robust recognition when large training data

is available. In addition to gesture recognition, DTW supports gesture evaluation and HMM enables gesture registration that are useful for extending the gesture model.

Through the development of prototype applications, the framework has proven to be a versatile toolkit to develop 3D gesture interfaces. We identified and implemented the right abstractions and services for designing 3D spatial gesture interfaces. Several reusable hardware and software components have been developed for a variety of applications maximizing the amount of reuse across applications. These issues have not previously arisen because the efforts required to build a unified framework was too high to allow for any further exploration beyond research prototypes.

In this dissertation, we proposed two unique gesture interfaces developed using our framework: a spatial context aware interface and a versatile gesture interface. We demonstrated the usefulness of each gesture interface showing how hardware and software components are linked together and applied to support required functionalities. Our goal was not to make 3D spatial gestures inputs for all applications, but to achieve maximal effectiveness for the potential gesture-based applications.

Figure 8.1 shows how major contributions are related to the design of a 3D spatial gesture interface in terms of three design components: input device, gestures, and contexts. We categorized two types of users: interface designers and end-users. Interface designers will be able to easily facilitate input device, gestures, and contexts for the design of their gesture interfaces. Focusing on the main component, the interface designers are further categorized device designers, gesture designers, and context designers. Their main task is to analyze the requirements of applications for which each component of a gesture interface is intended. Our framework does not only support system developers, but also enables end-users to customize the interface that can be optimized to the conditions of the end-users.

Now, we review the contributions categorized into three design components of gesture interface as follows:

- **Input Devices: Gesture Acquisition and Versatile Input Device**
  - *Visual and body sensors for input device designers.* Input device designers can use visual and body sensors for building an input device to acquire 3D spatial gestures. The combined sensor data enables robust feature detection for 3D gesture analysis and recognition. The proposed acquisition technique is low in cost, easy-to-use and enables users to quickly customize a cost-effective solution for a specific application. Finally, the intended gesture characteristics can be acquired efficiently.
  - *Versatile gesture interface for end-users* End-users can use a wide range of gesture-based inputs without changing to another device or interfaces. In this dissertation, a versatile gesture interface was developed to explore a way of using 3D spatial gestures as a general input in

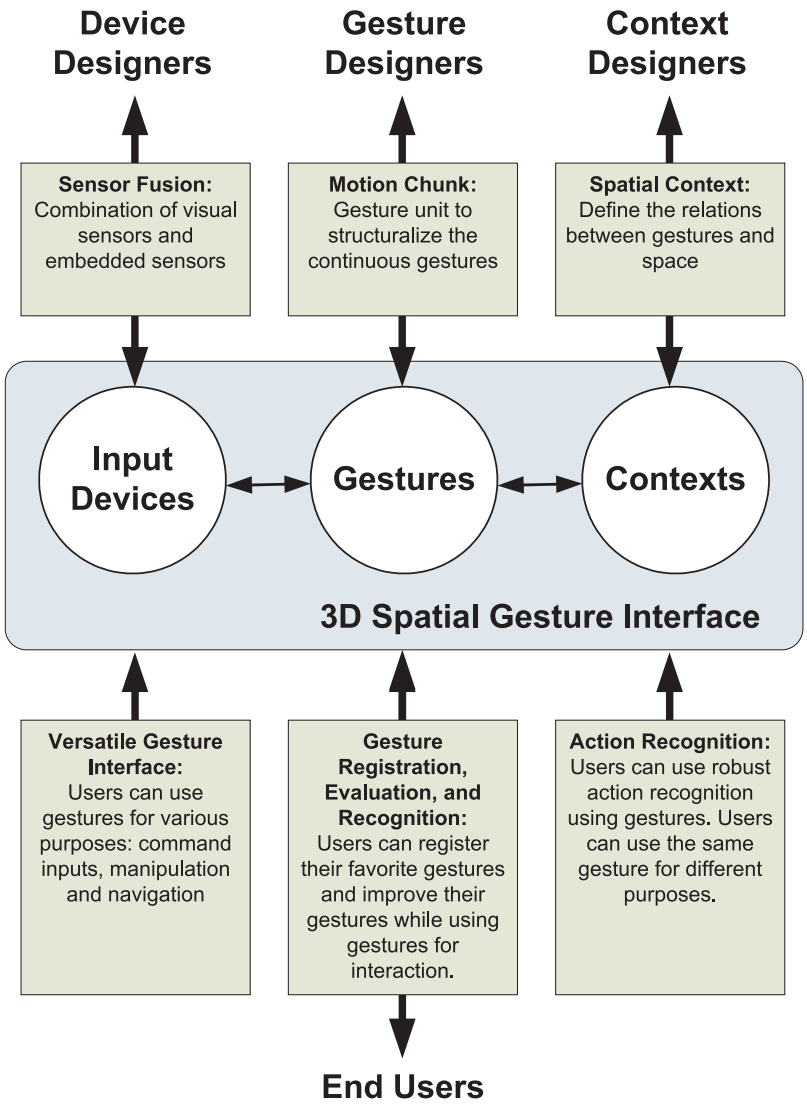


Figure 8.1: An overview of the major contributions according to the 3D Spatial gesture interface design. The possible organization and interaction of the design of a 3D spatial gesture interface based on the elements presented in this dissertation.

a wide variety of application domains including smart environments, computer graphics, VR and AR systems. The ultimate goal is to make the 3D spatial gesture interfaces adaptable to different scenarios.

- **Gestures: Representation, Registration, Recognition, and Evaluation**
  - *Motion chunk based gesture design for gesture designers.* Motion chunk was developed as the core representation scheme for gesture modeling. Based on the motion chunk, a gesture designer can design 3D spatial gestures. Start and end postures can be designed separately and the in-between gesture can be defined by connecting the previ-

ously fixed start and end postures. The designed gestures are registered to a gesture model and used for recognition and evaluation by using two pattern matching techniques (DTW and HMMs).

  - *Gesture registration and evaluation for end-users.* Gestures can be created and registered by end-users. End-users can consider their physical limitation related to their body and location to accomplish the required tasks by performing gestures. Therefore, end-users can operate gesture interface with the minimum efforts. On the other hand, if they have to learn pre-defined gestures, the learning period can be reduced because the system can evaluate their gestures by comparing them to the standards.

- **Spatial Contexts: Representation, Registration, Selection**
  - *Gesture targets and volumes for context designers.* Our framework supports context designers exploring the common issues shared between 3D spatial gestures and contexts. In particular, they focus on the concept of using the spatial information of gestures. In this dissertation, we provided a notion of spatial context with two ontologies: gesture target and volume. Using these spatial contexts, context designers can plan the locations and objects that are relevant to the gesture-based inputs, and minimize the system errors in misinterpreting the user's intention.
  - *Spatial context-aware gesture interface for end-users.* End-users can use gestures to control the functions incorporated with the physical and virtual objects in a smart environment. The spatial context aware gesture interface interprets gesture-based commands by incorporating spatial context information. In particular, this interface is novel in that it integrates spatial contexts acquired from the same sensors used for the acquisition of 3D spatial gestures. Therefore, end-users do not need additional sensors for the spatial contexts.

8.2 Future Directions

This dissertation described a new field of using 3D spatial gestures as an input. This section suggests several studies that would increase the understanding and usage of 3D spatial gestures as an input. We also discuss the advantages and drawbacks of the presented approaches and possible directions for future research.

8.2.1 A Design Method for Gesture Interface

When developing gesture interfaces in practice, there are several processes that designers must consider. For instance, they have to identify the type of gesture-

based inputs appropriate to the required target application tasks. They also need to consider the target users and determine the feasibility of using 3D spatial gesture inputs for a particular application. Designers then can think about how gesture-based inputs can be realized in the system while considering other factors namely acquisition techniques and recognition methods.

This dissertation presented a set of hardware and software components that can be used to design 3D spatial gesture interface. If there is a well defined design method, designers can improve the quality of their gesture interface. Based on our framework, it is possible to build such a design method for 3D spatial gesture interface which users can follow during the development of their own systems.

Figure 8.2 shows a possible design method that includes iterative processes. It consists of three major phases: *analysis*, *design*, and *evaluation*. In the analysis phase, interaction designers analyze the required tasks for target applications, and also analyze possible interaction goals that can be achieved using 3D spatial gestures. After the analysis phase, interaction designers can begin the actual design of 3D spatial gesture inputs following three sub-design tasks: device design, gesture design, and context design. These three tasks are equivalent to the three design components for 3D spatial gesture interfaces as described in Section 8.1.

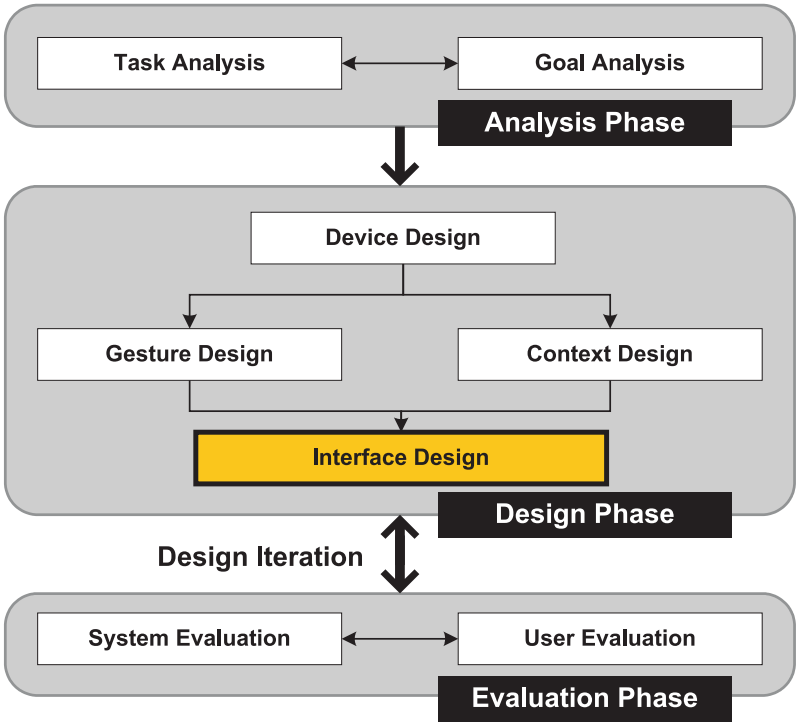


Figure 8.2: An overview of the design method for 3D spatial gesture inputs. There are three major phases: analysis, design, and evaluation. Each phase consists of sub-design tasks with possible interactions between tasks. Designer iteration can occur between design phase and evaluation phase.

8.2.2 Gesture Input Devices

We proposed two gesture input devices based on the concept of combining visual sensors and body sensors. The mWire was designed for easy connection to other sensors such as digital compasses and gyro sensors. The mCube was designed as a versatile gesture interface that supports various gesture-based inputs. In particular, this device supports both desktop and hand-held positions so that the device can be useful for ubiquitous computing environments equipped with various types of display platforms. The proposed devices are promising and potential. However, more work is required to gain a better understanding of the capabilities and limitations of the device. This section describes two future directions for the further development of gesture input devices.

Alternative Input Device Design

The next development of alternative input device is to examine the creation of smaller versions of the gesture input device while retaining important characteristics of the gesture input devices. These days, the advance of Micro-Electro-Mechanical Systems (MEMS) technology allows improvements in terms of size, accuracy, and communication. For instance, one available sensor package is designed around a stacked configurable circuit board architecture, with a high-bandwidth transceiver on the bottom, a general microprocessor and Analog-to-Digital Converter (ADC) in the middle, and a sensor board on top. With smaller devices, new application domains can emerge. In addition, we could minimize fatigue problems.

Figure 8.3 shows one of our alternatives called *cubeRing*. We study not only the smaller size but also ergonomic issues. The cubeRing is designed with a highly unobtrusive form retaining the benefits of the mCube. The flat bottom surface of the cubeRing provides a stable operation on the desktop surface, and the unique combination of a ring and a cube provides a comfortable grip when used in the desktop and the hand-held positions. The re-adjustable band is used to adapt to the different hand sizes.

Figure 8.3: An example of an alternative input device design called cubeRing. (a) The cubeRing with an adjustable finger ring on the top, and a small cube on the bottom. The cubeRing can be worn on the index finger, and operated on a desktop surface (b) and the cubeRing in the hand-held position (c).

A coin size MICA dot module [Cro] is used with a high-bandwidth transceiver, a general microprocessor and Analog-Digital-Converter (ADC), and a sensor board. The MICA dot modules is provided as a stacked configurable circuit board architecture. The cubeRing is currently equipped with a 2D-axis accelerometer, and colored LEDs following the concept of combining visual and embedded senors. Two color LEDs are attached, one on the top side and one on the bottom side to make robust tracking even with a hand rotation.

We investigate the design variations and gain deeper insights into the perceptual issues involved in interacting with this class of input devices. We also intend to perform usability studies to assess the capabilities and user acceptance of this versatile input system with various target applications.

Technical Analysis of Input Devices

Another future work for gesture input device is to analyze the technical properties of the gesture input devices. In general, there are three major features for the analysis: *resolution*, *sampling rate*, and *lag*. The resolution of the device is the smallest change it can detect in the quantity that is measuring. Usually, resolution varies with the proximity of the sensor to the source and other factors. The sampling rate is the number of measurements per unit time (e.g., samples per second). The lag is the phenomenon of not having immediate updates of the display in response to input actions.

The resolution and sampling rate should be as high as possible, and lag should be as low as possible. Obviously, these parameters are constrained or fixed at some reasonable level during the system design. Each feature affects the performance of gesture interface. The resolution is a major consideration when selecting a gesture input device for manipulations. The sampling rate can restrict the speed of gestural performance. For instance, when the input gesture is too fast then only few gesture samples can be acquired. Therefore, a high sampling rate is essential in 3D spatial gestures to support a wide range of gesture types and applications.

The lag restricts the system responsiveness taking hold. This can be critical when operating applications within a real-time environment. In gesture interface, there are three major stages to influence the lag: *acquisition*, *recognition*, and *output*. The recognition stage is optimized by reducing the computation time for gesture processing. If we can restrict a certain number of gesture candidates, then we can reduce the lag. The spatial context described in this dissertation can support the optimization of the gesture candidates.

8.2.3 Gesture Recognition

In this dissertation, we focused on analyzing single gestures. Future work will be devoted to the analysis of other gestures and development of methods to handle them. This section categorizes them into three groups: *activity level gestures*,

seamlessly continuous gestures, and *whole-body gestures*.

Recognition of Activity Level Gestures

The current gesture model can be extended to efficiently handle an *activity level* gesture. The activity level gestures can be composed with multiple sub-gestures. Motion chunk can be useful for designing activity level gestures which reduce the computational complexity. Rather than having many separate gesture models, we can construct a network of motion chunks and have paths through the network indicating a gesture or a sequence of gestures.

In addition, we can use the descriptive ability of a hidden Markov model (HMM) for the temporal structure of the motion chunks. For instance, a certain activity level gesture can be described as a collection of multiple Markov models, each associated with individual gestures. Similar to speech recognition, gesture grammar can be defined based on domain knowledge. We use previously classified gestures to constrain gesture candidates for the current recognition. Grammars also allow users to define a set of possible gestures in a certain situation with a location and time.

Recognition of Seamlessly Continuous Gestures

This dissertation discussed the gestures performed with clear boundaries between gestures. A user has to show the boundaries explicitly. For instance, the user needs to press a button while performing the gestures or pauses for a certain period of time before and after the gesture. Special purposes such as game control and sport analysis require handling seamlessly continuous gestures.

In general, the recognition of such gestures is non-trivial because the silent boundaries are no longer evident. Moreover, co-articulation can occur because gestures can be performed immediately after the previous gesture using different speeds and powers. The caused co-articulation effects change the shape of gesture signal. The major problem we face in this situation is segmentation - the problem of deciding where one gesture ends and the next begins. Similar to speech recognition, it is hard to segment gestures when different gestures are combined without having a pause in-between.

There could be heuristic techniques simultaneously optimizing the segmentation and recognition of the gestures. However, this approach is computationally expensive to handle several 3D gestures at the same time. The major reason is that every possible segmentation is hypothesized for all possible gesture sequences. One possible solution is to define gesture candidates depending on the user and contexts.

Recognition of Whole Body Gestures

For handling the whole body gestures, we can extend the sensor positions to other body parts such as ankles or shoulders. This would require not only large infrastructure and additional acquisition modules, but also fast and efficient gesture recognition methods. As the number of sensors increase, the amount of sensor data increases so that the computation needs to be faster in order to respond to the gesture inputs in real time.

These days, given the growth of portable computers (wearables, PDAs, etc. and advancing wireless sensor networks), one emerging application domain is human body sensor networks that can be used for the acquisition and use of whole body gestures. These systems generally use a set of small wireless sensor nodes equipped with various sensors. The sensor nodes are designed for different body location. Using these sensor data from all sensor nodes, whole body gestures can be tracked and analyzed for the application tasks.

8.2.4 User Evaluations

The dissertation presented results of the experimental evaluations that can be used as a guideline when developing 3D spatial gesture interface. However, most of experiments were performed in a short period of time and with a limited number of subjects. For more formal usability studies, it is necessary to design experiments with longer time period and larger user groups.

A possible user evaluation for future work can be designed to analyze the characteristics of 3D spatial gestures in terms of the user's body condition. From our preliminary tests, we found out that it is important to ensure that physically stressing gestures are avoided. It is highly required to study the ergonomics and bio-mechanics of gesturing given a performance task.

It is also required to evaluate the general capabilities of performing 3D spatial gestures. Since gesture interfaces can be applied to various applications, the morphology of 3D spatial gestures need to be separated from a specific task function. For more effective gesture interface in various application domains, the study of 3D spatial gesture capabilities is required.

This dissertation presented design principles for 3D gesture input devices and realized a versatility of the device. We also need to define a standard technique for evaluating 3D gesture input devices. Then both device engineers and interaction designers can be provided with valuable information for selecting the specifications of the device. Further study can be carried out to specify the optimal resolution and sampling rate for both manipulative gestures and communicative gestures.

Bibliography

[5DT] 5DT Data Gloves. Fifth Dimension Technologies, http://www.5dt.com/.

[6DM] 6DMouse^{TM}. Ascension Technology Corporation., http://www.ascension-tech.com.

[AAS05] J. Alon, V. Athitsos, and S. Sclaroff. Simultaneous localization and recognition of dynamic hand gestures. In *Proceedings of IEEE Motion Workshop*, 2005.

[AHT02] Brian Amento, Will Hill, and Loren Terveen. The sound of one hand: A wrist-mounted bio-acoustic fingertip gesture interface. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2002.

[AP04] J. K. Aggarwal and Sangho Park. Human motion: Modeling and recognition of actions and interactions. In *3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, pages 640–647, Thessaloniki, Greece, September 2004.

[BB01] Claus Bahlmann and Hans Burkhardt. Measuring HMM similarity with the Bayes probability of error and its application to online handwriting recognition. In *Proc. of the 6th ICDAR*, pages 406–411, 2001.

[BBKF97] R. Balakrishnan, T. Baudel, G. Kurtenbach, and G. Fitzmaurice. The rockin'mouse: Integral 3d manipulation on a plane. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 311–318, 1997.

[BBL93] T. Baudel and M. Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. In *Communications of the ACM*, volume 36, pages 28–35, July 1993.

[Bec97] D. Becker. Sensei: A real-time recognition, feedback, and training system for t'ai chi gestures. Master's thesis, Massachusetts Institute of Technology, 1997.

[Bel57] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, USA, 1957.

[Ber98] G. Berry. Small-wall: A multimodal human computer intelligent interaction test bed with applications. Master's thesis, University of Illinois at Urbana-Champaign,, 1998.

[BFH+03] N. Berthouze, T. Fushimi, M. Hasegawa, A. Kleinsmith, H. Takenaka, and L. Berthouze. Learning to recognize affective body postures. In *Computational Intelligence for Measurement Systems and Applications, 2003. CIMSA '03.*, pages 193–198, Washington, DC, USA, 2003. IEEE Computer Society.

[BGK+05] M. Barry, J. Gutknecht, I. Kulka, P. Lukowicz, and T. Stricker. From motion to emotion: A wearable system for the multimedial enrichment of a butoh dance performance. *Journal of Mobile Multimedia*, 1(2):112–132, 2005.

[BHI93] S. Bly, S. Harrison, and S. Irwin. Media spaces: bringing people together in a video, audio and computing environment. *Communications of the ACM*, 36(1):28–47, 1993.

[BI98] A. F. Bobick and Y. A. Ivanov. Action recognition using probabilistic parsing. In *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 196, Washington, DC, USA, 1998. IEEE Computer Society.

[BKKK04a] Seok-Hyung Bae, Takahiro Kobayash, Ryugo Kijima, and Won-Sup Kim. Tangible nurbs-curve manipulation techniques using graspable handles on a large display. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 81–90, New York, NY, USA, 2004. ACM Press.

[BKKK04b] S.H. Bae, T. Kobayash, R. Kijima, and W.S. Kim. Tangible nurbs-curve manipulation techniques using graspable handles on a large display. In S. Feiner and J.A. Landay, editors, *UIST*, pages 81–90. ACM, 2004.

[BKLP01] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola, and Ivan Poupyrev. An introduction to 3-d user interface design. *Presence: Teleoper. Virtual Environ.*, 10(1):96–108, 2001.

[BLB+03] Stephen Brewster, Joanna Lumsden, Marek Bell, Malcolm Hall, and Stuart Tasker. Multimodal 'eyes-free' interaction techniques for wearable devices. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 473–480, New York, NY, USA, 2003. ACM Press.

[BLK01] S. Baek, S. Lee, and G. J. Kim. Motion evaluation for vr-based motion training. In *Proceedings of Eurographics 2001*, 2001.

[BLK03] S. Baek, S. Lee, and G.J. Kim. Motion retargeting and evaluation for vr-based training of free motions. *The Visual Computer*, 19(4):222–242, July 2003.

[BLSB03] Jigna Bhatt, Niels Da Vitoria Lobo, Mubarak Shah, and George Bebis. Automatic recognition of a baby gesture. In *ICTAI '03: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, page 610, Washington, DC, USA, 2003. IEEE Computer Society.

[BLSM04] Jan Borchers, Eric Lee, Wolfgang Samminger, and Max Mühlhäuser. Personal orchestra: A real-time audio/video system for interactive conducting. *ACM Multimedia Systems Journal Special Issue on Multimedia Software Engineering*, 9(5):458–465, March 2004.

[Bol80] Richard A. Bolt. Put-that-there: Voice and gesture at the graphics interface. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, New York, NY, USA, 1980. ACM Press.

[BOP97] M. Brand, N. M. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, june 1997.

[BP98] Ravin Balakrishnan and Pranay Patel. The padmouse: facilitating selection and spatial positioning for the non-dominant hand. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 9–16. ACM Press/Addison-Wesley Publishing Co., 1998.

[BP02] Ari Y. Benbasat and Joseph A. Paradiso. An inertial measurement framework for gesture recognition and applications. In *GW '01: Revised Papers from the International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction*, pages 9–20, London, UK, 2002. Springer-Verlag.

[Bre97] Christoph Bregler. Learning and recognizing human dynamics in video sequences. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 568, Washington, DC, USA, 1997. IEEE Computer Society.

[C-c] C-control M2. Conrad Electronic, Inc., http://www.conrad.com.

[Cad94] Claude Cadoz. Les realites virtuelles. 1994.

[CB96] Lee W. Campbell and David A. Becker. Invariant features for 3-d gesture recognition. *Second International Workshop on Face and Gesture Recognition*, 1996.

[CB03] Xiang Cao and Ravin Balakrishnan. Visionwand: interaction techniques for large displays using a passive wand tracked in 3d. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 173–182. ACM Press, 2003.

[CBF05] Kyong I. Chang, Kevin W. Bowyer, and Patrick J. Flynn. Effects on facial expression in 3D face recognition. In *SPIE Conference on Biometric Technology for Human Identification*, volume 5779 of *SPIE Proceedings*, Orlando, FL, 2005.

[CD04] Diane Cook and Sajal Das. Smart environments: Technology, protocols and applications. 2004.

[CDSC03] C. Chua, N. H. Daly, V. Schaaf, and H. P. Camill. Training for physical tasks in virtual environments: Tai chi. In *Proceedings of IEEE Virtual Reality 2003 Conference*, pages 87–94, Los Angeles, Califonia, March 2003. IEEE Computer Society.

[CFBS97] Jeremy R. Cooperstock, Sidney S. Fels, William Buxton, and Kenneth C. Smith. Reactive environments. *Commun. ACM*, 40(9):65–73, 1997.

[CK00] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

[CNSD93] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142. ACM Press, 1993.

[Cor01] Andrea Corradini. Dynamic time warping for off-line recognition of a small gesture vocabulary. In *RATFG-RTS '01: Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01)*, page 82, Washington, DC, USA, 2001. IEEE Computer Society.

[Cro] Crossbow Technology., http://www.xbow.com/.

[CVWB02] G. S. Chambers, S. Venkatesh, G.A.W. West, and H.H. Bui. Hierarchical recognition of intentional human gestures for sports video annotation. In *Proceedings of ICPR02*, pages 1082–1085, Thessaloniki, Greece, September 2002.

[DB98] J. W. Davis and A. F. Bobick. Virtual pat: A virtual personal aerobics trainer. In *Proceedings of Workshop on Perceptual User Interfaces*, pages 13–18, November 1998.

[DHB+04] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a cappella: programming by demonstration of context-aware applications. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 33–40, New York, NY, USA, 2004. ACM Press.

[DL01] Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM Press.

[DP] T.J. Darrell and A.P. Pentland. Space-time gestures. In *Proc. Comp. Vis. and Pattern Rec.*

[Efr41] David Efron. Gesture and environments. 1941.

[Ele] Eleksen Limited. Elektex product literature, www.elektex.com.

[FH95] S. Fels and G. Hinton. Glove-talkii: An adaptive gesture-to-formant interface. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 1995.

[FHSH06] B. Froehlich, J. Hochstrate, V. Skuk, and Anke Huckauf. The globefish and the globemouse: Two new six degree of freedom input devices for graphics applications. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 191–199, 2006.

[Flo] Flock of Birds^{TM}. Ascension Technology Corporation, http://www.ascension-tech.com.

[FLZ98] A. Forsberg, J. LaViola, and R. Zeleznik. Ergodesk: A framework for two and three dimensional interaction at the activedesk, 1998.

[FMea99] J. Farringdon, A.J. Moore, and N. Tilbury et al. Wearable sensor badge and sensor jacket for context awareness. In *Proceedings of The Third International Symposium on Wearable Computers*, pages 107–113, 1999.

[FP00] Bernd Fröhlich and John Plate. The cubic mouse: a new device for three-dimensional input. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 526–531. ACM Press, 2000.

[GBK+01] Tovi Grossman, Ravin Balakrishnan, Gordon Kurtenbach, George Fitzmaurice, Azam Khan, and Bill Buxton. Interaction techniques for 3d modeling on large displays. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 17–23, New York, NY, USA, 2001. ACM Press.

[GJ95] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden Markov models. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 8, pages 472–478. MIT Press, 1995.

[Gli00] Albert Glinsky. Theremin: Ether music and espionage. 2000.

[Gro02] T. Grossman. Creating principal 3d curves with digital tape drawing. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 526–531, 2002.

[GWN+03a] M. Gross, S. Wuermlin, Martin Naef, E. Lamboraz, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. Vande Moere, and O. Staadt. blue-c: A spatially immersive display and 3d video portal for telepresence. In *Proceedings of ACM SIGGRAPH 2003*, pages 819–827, 2003.

[GWN+03b] M. Gross, S. Würmlin, M. Näf, E. Lamboray, C. Spagno, A. Kunz, A. V. Moere, K. S, S. Lang, T. Svoboda, E. Koller-Meier, L. V. Gool, and O. Staadt. blue-c: A spatially immersive display and 3d video portal for telepresence. volume 22, pages 819–827. ACM Press, 2003.

[HGC94] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994.

[HIH+05] Perttu Hämäläinen, Tommi Ilmonen, Johanna Höysniemi, Mikko Lindholm, and Ari Nykänen. Martial arts in artificial reality. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 781–790, New York, NY, USA, 2005. ACM Press.

[HS98a] C. Hummels and P. J. Stappers. Meaningful gestures for human computer interaction: Beyond hand postures. In *FG '98: Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, page 591, Washington, DC, USA, 1998. IEEE Computer Society.

[HS98b] C. Hummels and P. J. Stappers. Meaningful gestures for human computer interaction: Beyond hand postures. In *FG '98: Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, page 591, Washington, DC, USA, 1998. IEEE Computer Society.

[HSH+99] Ken Hinckley, Mike Sinclair, Erik Hanson, Richard Szeliski, and Matt Conway. The videomouse: a camera-based multi-degree-of-freedom input device. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 103–112. ACM Press, 1999.

[IvDC+00] H.A. Ingram, P. van Donkelaar, J. Cole, J.L. Vercher, G.M. Gauthier, and R.C. Miall. The role of proprioception and attention in visuomotor adaptation task. *Exp. Brain Res*, 132:114–126, 2000.

[JHT06] Wan-rong Jih, Jane Yung-jen Hsu Hsu, and Tse-Ming Tsai. Context-aware service integration for elderly care in a smart environment. In *Modeling and Retrieval of Context Retrieval of Context: Papers from the AAAI Workshop*,

number WS-06-12, pages 44– 48, Boston, Massachusetts, USA, July 16 – 20 2006.

[Jr01] L. Jr. Quill: a gesture design tool for pen-based user interfaces, 2001.

[Ken80] Adam Kendon. Gesticulation and speech: Two aspects of the process of utterance. pages 207–227, 1980.

[KFA+04] Azam Khan, George Fitzmaurice, Don Almeida, Nicolas Burtnyk, and Gordon Kurtenbach. A remote control interface for large displays. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 127–136, New York, NY, USA, 2004. ACM Press.

[KP01] E. Keogh and M. Pazzani. Derivative dynamic time warping, 2001.

[KPE+06] Paul Keir, John Payne, Jocelyn Elgoyhen, Martyn Horner, Martin Naef, and Paul Anderson. Gesture-recognition with non-referenced tracking. In *3DUI '06: Proceedings of the 3D User Interfaces (3DUI'06)*, pages 151–158, Washington, DC, USA, 2006. IEEE Computer Society.

[Kru91] W. Krueger. Artificial reality ii. 1991.

[Kur93] G. Kurtenbach. *The Design and Evaluation of Marking Menus*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1993.

[LEC+06] A. Luciani, M. Evarard, D. Courousse, N. Castagne, C. Cadoz, and J. Florens. A basic gesture and motion format for virtual realty multisensory applications. In *Proceedings of the 1st international GRAPP Conferences on Computer Graphics Theory and Applications*, 2006.

[Lip91] James S. Lipscomb. A trainable gesture recognizer. *Pattern Recogn.*, 24(9):895–907, 1991.

[LVS+03] Kristof Van Laerhoven, Nicolas Villar, Albrecht Schmidt, Gerd Kortuem, and Hans Gellersen. Using an autonomous cube for basic navigation and input. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 203–210. ACM Press, 2003.

[MA05] Florian 'Floyd' Mueller and Stefan Agamanolis. Sports over a distance. *Comput. Entertain.*, 3(3):4–4, 2005.

[MBS97] Mark R. Mine, Frederick P. Brooks, Jr., and Carlo H. Sequin. Moving objects in space: Exploiting proprioception in virtual-environment interaction. *Computer Graphics*, 31(Annual Conference Series):19–26, 1997.

[McN95] David McNeill. Hand and mind: What gestures reveal about thought. 1995.

[MDBP96] P. Maes, T. Darrell, B. Blumberg, and A. Pentland. The alive system: Wireless, full-body interaction with autonomous agents, 1996.

[ML04] S. Malik and J. Laszlo. Visual touchpad: a two-handed gestural input device. In R. Sharma, T. Darrell, M.P. Harper, G. Lazzari, and M. Turk, editors, *ICMI*, pages 289–296. ACM, 2004.

[Mot] Motion Analysis Corporation, http://www.motionanalysis.com .

[MS83] Richard J. Meinhold and Nozer D. Singpurwalla. Understanding the kalman filter. *The American Statistician*, 37(2):123–127, 1983.

[NFU99] H. Nishino, M. Fushimi, and K. Utssumiya. A virtual environment for modeling 3d obejcts through spatial interaction. In *1999 IEEE International Conference on Systems, Man, and Cybernetics*, Washington, DC, USA, 1999. IEEE Computer Society.

[NLP+02] Ara V. Nefian, Luhong Liang, Xiaobo Pi, Xiaoxing Liu, and Kevin Murphy. Dynamic bayesian networks for audio-visual speech recognition. *EURASIP Journal on Applied Signal Processing*, 2002(11):1274–1288, 2002.

[NW96] Y. Nam and K. Wohn. Recognition of space-time handgestures using hidden markov model, 1996.

[Ope] OpenSource Computer Vision Library. Intel Corp., http://www.intel.com.

[Par99] J. Paradiso. The brain opera technology: New instruments and gestural sensors for musical interaction and performance, 1999.

[PFea99] J.K. Perng, B. Fisher, and S. Hollar et al. Acceleration sensing glove. In *Proceedings of The Third International Symposium on Wearable Computers*, pages 178–179, 1999.

[PH97] Rosalind W. Picard and Jennifer Healey. Affective wearables. In *ISWC*, pages 90–97, 1997.

[PHBT00] J. A. Paradiso, K. Hsiao, A. Y. Benbasat, and Z. Teegarden. Design and implementation of expressive footwear. *IBM Syst. J.*, 39(3-4):511–529, 2000.

[PHH99] J. Paradiso, E. Hu, and K. Hsiao. The cybershoe: A wireless multisensor interface for a dancer's feet. In *Proc.International Dance and Technology 99*, pages 26–28, Feb, 1999. Tempe AZ.

[Pic97] R.W. Picard. *Affective Computing*. The MIT Press, Cambridge,MA, 1997.

[PKE+06] John Payne, Paul Keir, Jocelyn Elgoyhen, Mairghread McLundie, Martin Naef, Martyn Horner, and Paul Anderson. Gameplay issues in the design

of spatial 3d gestures for video games. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 1217–1222. ACM Press, 2006.

[PL99] Alex Pentland and Andrew Liu. Modeling and prediction of human behavior. *Neural Comput.*, 11(1):229–242, 1999.

[Pup] Puppet Works, http://www.puppetworks.com .

[Que94] Francis K. H. Quek. Toward a vision-based hand gesture interface. In *VRST '94: Proceedings of the conference on Virtual reality software and technology*, pages 17–31, River Edge, NJ, USA, 1994. World Scientific Publishing Co., Inc.

[Rab89] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, Feburuary 1989.

[RAM+05] Cliff Randell, Ian Anderson, Henk Muller, Andrew Moore, Philippa Brock, and Sharon Baurley. The sensor sleeve: Sensing affective gestures. In *Ninth International Symposium on Wearable Computers - Workshop on On-Body Sensing*, October 2005.

[RB05] Stephan Rusdorf and Guido Brunnett. Real time tracking of high speed movements in the context of a table tennis application. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 192–200, New York, NY, USA, 2005. ACM Press.

[Rek01] Jun Rekimoto. Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. In *ISWC '01: Proceedings of the 5th IEEE International Symposium on Wearable Computers*, page 21, Washington, DC, USA, 2001. IEEE Computer Society.

[Rek02] Jun Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, New York, NY, USA, 2002. ACM Press.

[RHS01] R. Duda R, P. Hart, and D. Stork. *Pattern Classification*. 2nd Edition. John Wiley and Sons Press., 2001.

[RJ93] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall PTR, April 1993.

[RS91] B. Rime and L. Schiaratura. Gesture and speech: Fundamentals of nonverbal behavior. 1991.

[RS99] J. Rekimoto and M. Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 378–385, 1999.

[RS00] Jun Rekimoto and Eduardo Sciammarella. Toolstone: Effective use of the physical manipulation vocabularies of input devices. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 109–117. ACM Press, 2000.

[RWC+98] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 179–188. ACM Press, 1998.

[SAA00] Thad Starner, Jake Auxier, and Daniel Ashbrook. The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *Proceedings of ISWC 2000*, 2000.

[SGB99] A. Schmidt, H.W. Gellersen, and M. Beigl. A wearable context-awareness component. In *Proceedings of The Third International Symposium on Wearable Computers*, pages 176–177, 1999.

[SH97] H. Sawada and S. Hashimoto. Gesture recognition using an accelerometer sensor and its application to musical performance control. In *Electronics and Communications in Japan*, number WS-06-12, 1997.

[SOMM06] P. Ravindra De Silva, Minetada Osano, Ashu Marasinghe, and Ajith P. Madurapperuma. Towards recognizing emotion with affective dimensions through body gestures. In *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 269–274, Washington, DC, USA, 2006. IEEE Computer Society.

[SP] T. Starner and A. Pentland. *Visual Recognition of American Sign Language Using Hidden Markov Models*. M.I.T. Media Laboratory, Cambridge MA.

[SP95] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *SCV95*, page 5B Systems and Applications, 1995.

[Spaa] SpaceBall™. 3Dconnexion, http://www.3Dconnexion.com.

[Spab] SpaceMouse™. 3Dconnexion, http://www.3dconnexion.com.

[SSH05] Petra Sundstrom, Anna Stahl, and Kristina Hook. A user-centred approach to affective interaction. *Computer Science*, 2005.

[SSL+03] J. G. Sheridan, B. W. Short., K. Van Laerhoven, N. Villar, and G. Kortuem. Exploring cube affordance: Towards a classification of non-verbal dynamics of physical interfaces for wearable computing. In *Proceedings of Eurowearables 2003*, 2003.

[Sta] Thad Starner. Visual recognition of american sign language using hidden markov models. Technical report.

[SW84] M. Smyth and A. Wing. *The Psychology of Human Movement*. Academic Press, 1984.

[Tse04] Edward Hiatt Tse. *The Single Display Groupware Toolkit*. PhD thesis, Department of Computer Science, The University of Calgary, Alberta, 2004.

[TSST04] M. Takahata, K. Shiraki, Y. Sakane, and Y. Takebayashi. Sound feedback for powerful karate training. In *Proceedings of International Conference on New Interfaces for Musical Expression (NIME04)*, 2004.

[TY02] K. Tsukada and M. Yasamura. Ubi-finger: Gesture input device for mobile use. In *Proceedings of APCHI '02*, pages 388–400, 2002.

[TYo02] Esa Tuulari and Arto Ylisaukko-oja. Soapbox: A platform for ubiquitous computing research and applications. In *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*, pages 125–138, London, UK, 2002. Springer-Verlag.

[UI97] Brygg Ullmer and Hiroshi Ishii. The metadesk: Models and prototypes for tangible user interfaces. In *ACM Symposium on User Interface Software and Technology*, pages 223–232, 1997.

[VB05] Daniel Vogel and Ravin Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 33–42. ACM Press, 2005.

[Ven93] D. Venolia. Facile 3d direct manipulation. In *Proceedings of INTERCHI 1993*, pages 31–36, 1993.

[Wac] Wacom Tablet Technology., http://www.wacom.com/.

[Wan] Wanda™. Ascension Technology Corporation., http://www.ascension-tech.com.

[WB03] Daniel Wigdor and Ravin Balakrishnan. Tilttext: using tilt for text input to mobile phones. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 81–90. ACM Press, 2003.

[Web02] Andrew R. Webb. *Statistical Pattern Recognition*. 2nd Edition. John Wiley and Sons Press., 2002.

[WF98] Andrew D. Wilson and Aaron F.Bobick. Recognition and interpretation of parametric gesture. In *Proceedings of International Conference on Computer Vision*, 1998.

[Wii] *Wii*. http://wii.nintendo.com/.

[WJ88] C. Ware and D. R. Jessome. Using the bat: a six dimensional mouse for object placement. In *Proceedings on Graphics interface '88*, pages 119–124, Toronto, Ont., Canada, Canada, 1988. Canadian Information Processing Society.

[WS03] A. Wilson and S. Shafer. Xwand: Ui for intelligent spaces. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 545–522, 2003.

[Yan99] U. Yang. Just follow me: An immersive vr-based motion training system. In *Proceedings of International Conference on Virtual Systems and Multimedia*, 1999.

[YXC94] Jie Yang, Yangsheng Xu, and C.S. Chen. Gesture interface: Modeling and learning. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1747–1752, 1994.

[Zha98] S. Zhai. User performance in relation to 3d input device design. *ACM Computer Graphics*, 32(4):50–54, 1998.

[Zha99] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the 7th International Conference on Computer Vision 1999*, pages 662–673, 1999.

[Zha01] Liwei Zhao. *Synthesis and acquisition of laban movement analysis qualitative parameters for communicative gestures*. PhD thesis, 2001. Supervisor-Norman I. Badler.

Curriculum Vitae

Personal Data

Name	Doo Young Kwon
E-Mail	dkwon@inf.ethz.ch
Address	Computer Graphics Laboratory ETH Zentrum Haldeneggsteig 4 8092 Zurich Switzerland `http://graphics.ethz.ch/~dkwon/`
Date of Birth	September 29, 1973
Nationality	Republic of Korea

Education

Jan. 2004 – Now	Ph.D. Candidate at Computer Graphics Laboratory Department of Computer Science, ETH Zurich, Switzerland
Apr. 2002 – Nov. 2003	M.S Degree in Design Computing Department of Architecture, University of Washington, USA
Oct. 1992 – Mar. 2002	Undergraduate Studies in Architecture Department of Architecture, Ajou University, KOREA

Professional Experiences

Jun. 2004 – Nov. 2007	Research Assistant at the Computer Graphics Laboratory Department of Computer Science, ETH Zurich, Switzerland
Aug. 2002 – Dec. 2003	Research Assistant at the Design Machine Group Department of Architecture, University of Washington, USA
Aug. 2001 – Aug. 2002	Researcher at Center for Computer Graphics and Virtual Reality Department of Computer Science, Ewha Womans University, Korea
Jun. 1999 – Jul. 2001	Full time Researcher at the Space Digital Laboratory Yonsei University, Seoul, Korea
Mai 1993 – Oct.1996	Rescue and Safety Specialist at the Fire and Rescue Department Military Service, Republic of Korea Air Force, Wonju, Korea

Scientific Publications

Jin Won Choi, Doo Young Kwon, Jie Eun Hwang, Jumphon Lertlakkhanakul
Real-time Management of Spatial Information of Design: A Space-based Floor Plan Representation of Buildings, Automation in Construction, Elsevier, vol. 16, nr. 4, pp. 449-459, 2007.

Doo Young Kwon, Stephan Würmlin, Markus Gross
mCube: Towards a Versatile Gesture Input Device for Ubiquitous Computing Environments, Lecture Notes in Computer Science (UCS 2007), Springler, vol. 4836, pp. 228-239, 2007.

Doo Young Kwon, Markus Gross
A Framework for 3D Spatial Gesture Design and Modeling Using a Wearable Input Device, Proceedings of the 11th IEEE International Symposium on Wearable Computers, pp. 95-101, 2007.

Doo Young Kwon, Markus Gross
Combining Body Sensors and Visual Sensors for Motion Training, AProceedings of ACM SIGCHI ACE 2005, pp. 94-101 (Valencia, Spain, June 15 - June 17), 2005.

Seon Min Rhee, Soo Mi Choi, Doo Young Kwon, Myoung Hee Kim
Architectural Design using Visual and Tactile Guide in the Virtual Table, Journal of KISS: Computing Practicee, Volume 10, Number 2, April, 2004.

Myoung Hee Kim, Soo Mi Choi, Seon Min Rhee, Doo Young Kwon, Hyo Sun Kim

A Guided Interaction Approach for Architectural Design in a Table-Type VR Environment, Lecture Notes in Computer Science, Springer-Verlag, 2002.

Jin Won Choi, Doo Young Kwon, Hyun Soo Lee
DesignBUF: Exploring and Extending 2D Boolean Set Operations with Multiple Modes in the Early Design Phase, CAAD FUTURE 2001, pp. 589-602, 2001.

Yeonjoo Oh, Doo Young Kwon, Babak Ziraknejad, Ken Camarata, Ellen Yi-Luen Do
WINDOW SEAT: Visual Experience with an Interactive Chair, GCAD 2004, Pittsburgh, USA, 2004.

Doo Young Kwon
Cover Story : Design By Points, IEEE Computer Graphics and Applications Vol.24 July/August, 2004.

Doo-Young Kwon, Ellen Yi-Luen Do
Inspired by Eisenman: ArchiDNA, A Creative Shape Generating System, CAAD FUTURE, 2003.

Doo Young Kwon
ArchiDNA: A Generative System for Shape Configuration, Master Thesis, University of Washington, 2003.

Seon Min Rhee, Doo Young Kwon, Soo Mi Choi, Myoung Hee Kim
Interactive Object Modeling Within a Table-type Virtual Environment, Korean Information Processing Society (KIPS), Vol. 29-1, pp.592-594, 2002.

Myoung-Hee Kim, Soo-Mi Choi, Seon-Min Rhee, Doo-Young Kwon, Hyo-Sun Kim
A Guided Interaction Approach for Architectural Design in a Table-Type VR Environment, IEEE Pacific-Rim Conference on Multimedia (PCM2002), 2002.

Doo Young Kwon, Jin Won Choi
Fundamental Study on the Development of an Intelligent Architectural CAD System Supporting Web-based Collaborative Design, CAD/CAM, pp.200-210, 2000.

Jin Won Choi, Myong Sik Lee, Doo Young Kwon
An Analysis A Fundamental Study on the Development of a Web-based Intelligent Architectural CAD System, he Architectural Institute of Korea,19(2), pp.349-354, 1999.