UNIVERSITATEA STEFAN CEL MARE DIN SUCEAVA

UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

REAL-TIME ACQUISITION OF HUMAN GESTURES FOR INTERACTING WITH VIRTUAL ENVIRONMENTS

THESIS

Defended on 18 March 2008 for the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

RADU-DANIEL VATAVU

in front of the jury committee

President:	Adrian GRAUR, PhD Professor, University Stefan cel Mare of Suceava
Supervisors:	Stefan-Gheorghe PENTIUC, PhD Professor, University Stefan cel Mare of Suceava
	Christophe CHAILLOU, PhD Professor, Université des Sciences et Technologies de Lille, Laboratoire d'Informatique Fondamentale de Lille
Readers:	Radu Patrice HORAUD, PhD Director of Research, INRIA Grenoble Rhône-Alpes, Montbonnot
	Dan GÂLEA, PhD Professor, Technical University Gh. Asachi of Iasi
	Laurent Grisoni, PhD, Maître de Conférences, Université des Sciences et Technologies de Lille, Laboratoire d'Informatique Fondamentale de Lille
	Alexandru VALACHI, PhD Professor, Technical University Gh. Asachi of Iasi

- 2008 -

Laboratoire d'Informatique Fondamentale de Lille – UMR CNRS 8022, Graphix / Alcove

Centrul de Cercetare in Stiinta Calculatoarelor Suceava, Association Universitaire de la Francophonie

Contents

List of Figures

Acknowledgments

First of all, I would like to express my gratitude to Prof. Stefan-Gheorghe Pentiuc that I've been knowing since 1999 and, ever since then, he has been continuously supportive in every initiative I took. Especially for introducing me to the challenging world of research, stimulating and motivating my imagination and creativity as a young researcher over all these years. I owe to him my today interests in image processing and pattern recognition. This co-directed thesis wouldn't have been possible without his very much implication and unconditionally support and I thank him for this great opportunity he once more offered.

I would like to thank Prof. Christophe Chaillou for the special view he brought on my thesis, every from our first discussion we had in February 2005. He taught me to have a broader perspective on my work, to look beyond particular details and always have in mind the bigger end. While I was struggling on whether to install one, two or three cameras for the acquisition of gestures, he was concerned on making me see the greater purpose behind all this: are human gestures suitable for HCI? and how do we prove this? Thank you Christophe for this and, over the years, I tried to apply this principle to everything I did, work and life. And I still remember what you said during our first meeting: *c'est ta vie avant tout*.

A very special Thank You goes to Laurent Grisoni who has been so supportive on so many levels. There is a lot of administrative tasks and paperwork (university inscription, cotutelle manuscript, doctoral school, international relations, on-campus

hostels...) that he did for me helping a lot even before the thesis had begun. The greatest recognition is due however for monitoring my research and providing many great continuation ideas during our meetings. The key part of my thesis regarding the work on splines and continuous segmentation of motion was very much influenced by his contribution and certainly my thesis wouldn't have looked as it does today if it hadn't been for him. He also taught me to aim higher when presenting my work (even though we got many papers rejected over the years).

I would like to thank the Alcove/Graphix team at LIFL for receiving me for the entire time of one year: Samuel Degrande especially for his support during the first period of my thesis and my stages at LIFL, Gery, Adrien, Luciana, Johann, Cedric, Damien, Nicolas, Jeremie, Frederic, Ahmed et Pierre-Jean.

There are a few special persons that made my stays at Lille truly fantastic. Catalin Chera for the time spent together, our trips visiting Lille and surroundings, our long discussions (research related, of course) as well as for "Duminica sportului". Thank you for always being there, for listening and the many advices you gave me over these years when I needed them most. Adriana Bacila for our spending time together, trips in Lille, Sundays long-drawn out launches and for a very interesting, and sudden as it appeared, research collaboration. Vicentiu and Laura Mitrofan for nice memories they left me from when hanging together. And Pankaj, this *Namaste* is for you, thank you for being there at an important time in my life.

I would also like to thank all the subjects that "willingly" helped me gather gesture data: Catalin, Ana, Sebi, John, Felicia, Ovidiu, Florin, Gabi, Alex, Adrien, Cedric, Frederic, Damien, Jeremie, Pankaj, Ahmed, Johann, Vicentiu. Also thank you to Jacob Wobbrock for kindly allowing re-using the xml gesture database for my own testing.

This thesis was funded by University Stefan cel Mare of Suceava (October 2004 - February 2008), LIFL (February - May 2005) and the AUF Bourse de Formation a

la Recherche (September 2005 - June 2006, September 2006 - June 2007, September 2007 - December 2007) for which I am grateful. Support has been also assured through the national research grant Ref. No. 131-CEEXII03/2.10.2006 - Gestures based Interaction with Information and Robotics Systems - in the frame of the Research of Excellence National Funding Grants.

So many Thank You's and the greatest recognition are due to my wife and parrents that supported me all these years having to cope with my toughest and irritable moods and to which I dedicate my 3 years of work.

To my family,
which surrounded me with gestures
that deserve the greatest recognition.

Introduction

***Novice user**: – How do I change the color of this sphere?*

***HCI chief designer**: – Well, it's simple, just hold your right hand in the fist posture, rotate it 45 degrees clock-wise and use your left hand to draw the letter C in mid-air while nodding your head. Say "start" to begin and "stop" when you're done. Doesn't anyone read manuals anymore?*

...

(after 20 minutes)

...

***Novice user**: – It doesn't work ?! All I get is the "Error: Bad command" Ok/Cancel message box.*

***HCI chief designer**: – Make sure that your C's respect the 3:8 width:height ratio !! And mind your speed, do it slowly !! Pfff... novices...*

Gestures in human-computer interfaces have emerged for a few years now since Bolt first demonstrated in the 80's how to easily *put that there* [Bol80]. Ever since then many technologies have been developed for the acquisition of gestures performed by hands, arms, legs or entire body and many recognition algorithms have been proposed of which [Wat93, PSH97, LaV99, TSW90] provide great surveys. Equally important, interacting by gestures needs now its own chapter in the HCI

book due to the many specifics that it brings up. Commercial or open source software applications (see Mozzila) decided to include gesture commands on top of their standard interaction techniques. The domain is a young one and clearly continuously evolving. Even more, the general everyday perception of gestures is that they will represent the "next thing" in the near future of computer interaction, perception that is very much sustained by the media and the film industry.

Things don't currently stand as in the above imaginary discussion but the dialog has its merits of highlighting a few important issues for the research community, such as: gesture dictionaries, self-revelatory interfaces, appropriate feedback or multi-user adaptive recognition algorithms. First of all, there is the obvious problem of gesture dictionaries and finding the appropriate set of gesture commands for a given application. An ideal gesture command should be precise, accurate, easy to understand and memorize, ergonomic and, if at all possible, self-revelatory (no manuals). Another interesting point that the above sketch brings up is related to multi-user gesture recognition. It is a known fact that there is a certain amount of variation that comes with each gesture execution: two users will execute the same gesture differently and even more, the same user will input different strokes for two consecutive executions of the same gesture command. Recognition algorithms need to account for this variation and to adapt to users' executions as failure of incorporating this extra knowledge often leads to failure in recognition. Gesture interfaces should be equally complemented by an appropriate visual feedback (no more error messages) letting users be aware of their interaction success in a fluent and graceful manner. Another important problem is related to the segmentation of continuous motion trajectories into meaningful gesture patterns: should the user be the one that clearly indicates where the gesture starts and where it ends or should this be left to the system? Automatic segmentation of motion is a difficult problem per se and, as to our knowledge, no robust real-time solution has been

proposed so far. Many open questions exist regarding gesture interfaces, of which one we extract from the above dialogue: do we need gestures in order to change the color of a sphere or are there better ways to do this while gestures become just complementary interaction?

This thesis deals with gesture recognition with the focus on providing a flexible model for gestures as well as for the amount of variation present in gesture execution. Variability in execution (VE) can be estimated for single or multiple users and the model of variation may be used in order to address the hard problem of continuous motion segmentation or performing ergonomic analysis on gesture dictionaries.

The first chapter presents an overview on the current state-of-the-art in gesture-based interaction. We start by simply *looking at human gestures* and thus bring into discussion several interesting results and observations as derived from various psycholinguistic studies [Ken86, McN92, Cad94, Cas96]. We are first interested in *what a gesture is* and *what types of gestures are there* and the psycholinguistic references provide a great start on the subject. Definitions of gesture from different disciplines of study are presented and an overview on gesture taxonomies is given. We then continue by inventorying the existing technologies that allow the *acquisition of human gestures* and provide references to commercial or research prototype implementations from the literature. A special focus is put on *video-based capture of gestures* by highlighting the advantages as well as drawbacks that come with vision processing. All the practical implementations that served as support to the results of this thesis acquire gestures using video cameras hence we present and discuss the specific *challenges of vision processing* right from the very starting chapter. The chapter ends with a *survey on recognition techniques* for the tracking, posture and trajectory matching of human gestures with references to the existing research literature as well as to available products or libraries.

Chapter two addresses the problem of visual acquisition of gestures performed

by one or both hands by discussing several specific details with regards to the interaction scenario, hands detection, posture recognition and retrieving of motion trajectories. We start by describing the *gesture acquisition scenario* that will be used in all further experiments. A simple *solution for hand postures recognition* is presented together with an interaction algorithm that implements a few common operations for manipulating virtual objects inside virtual environments, namely selection, translation, rotation and change of scale.

We introduce in the third chapter several *models for gestures and dictionary of gestures* from the point of view of the human computer interaction domain. By considering the amounts of static and dynamic information that are necessary in order to sufficiently and completely describe a gesture command, we have identified *four distinct types of gesture commands* that we referenced as simple static, complex static, simple dynamic and complex dynamic. Mathematical formulations as well as examples and discussions are given for each of the four gesture types. We provide a *general definition of gesture as command for HCI* that involves the characteristics of the four specific types. We further develop on top of our gesture definitions and consider aspects related to gesture dictionaries. We introduce measures of similarity and dissimilarity for gestures that we use in order to define several dictionary types. In the end of the chapter, we follow and discuss the multiple representations that gestures as commands take at various levels in human computer interaction systems.

The fourth chapter discusses the problem of detecting similarities between gesture patterns and we introduce for this purpose a *novel representation of gestures based on spline modelling*. The use of splines brings in a few advantages with regards to mathematical modelling, data dimensionality, speed and accuracy of processing. Gestures are known to contain embedded a certain *degree of variability* in the sense that no two gesture stroke executions are exactly the same, even if acquired from the same subject consecutively. We move toward this issue and further *enhance our*

spline representations with elastic properties in analogy with the elasticity theory from basic physics. We look at each spline gesture as a series of connected elastic springs that may be subjected to deformations such as stretching and bending. Our model becomes more flexible in order to accommodate the intrinsic variation of multiple executions. The similarity measure that we use is influenced by the works of [SKK03, SKK01, BCGJ98] on curves alignment using standard dynamic programming techniques. We describe a *training algorithm in the context of supervised learning* for automatic computation of gesture templates and test the performances of our pattern matcher on a large set of gesture samples.

Chapter five strengthens our start-up approach of addressing variation issues from chapter four by introducing *a model that describes the amount of variability present in gesture execution* . We further demonstrate two immediate applications of our variation execution (VE) model in connection with two important problems: segmenting continuous motion and performing ergonomic analysis on gestures. We begin by addressing a well-known hard problem which is the *automatic segmentation of continuous motion trajectories* into meaningful gesture patterns. The problem is hard due to the high complexity needed for the detection of patterns of any size, at any rotation and at any starting point in a given larger trajectory. We discuss an extension of our pattern matcher from chapter four for the detection of such gesture patterns in continuous motions. We conclude the chapter with a *validation of our VE model* by verifying two intuitive hypothesis on *how execution is affected by articulation speed or gesture complexity*. Our discussions connect to existing research, recently made available at ACM UIST in October 2007.

Chapter 1

An Overview on Current Gesture-based Interaction

We can pay our debt to the past
by putting the future in debt to ourselves.
John Buchan
(1875 – 1940)

The chapter describes the state-of-the-art in gesture-based interaction by considering general aspects related to human gestures as derived from various psycholinguistic studies, references to existing acquisition technologies and devices as well as methods and techniques employed for the recognition of gestures.

We are firstly interested in defining human gestures and we consider for this purpose several definitions as they are introduced by various research communities. Interesting aspects on gestures and gesture execution are derived from psycholinguistic studies that provide thorough gesture analysis as well as taxonomies on various criteria. We equally define, discuss and highlight the difference between postures as static information with respect to the dynamics of motion.

Current technologies for the acquisition of human gestures are further discussed together with references to practical systems implementations from the literature. Advantages as well as drawbacks of each technology are taken into account. A

special subchapter is dedicated to video-based acquisition: we discuss the specific advantages that video gesture acquisition brings as well as the current challenges encountered in the computer vision community. Discussions and comments are given with regards to the specific industry of gaming and the existing devices that are commercially available for capturing the human motion for the purpose of game control. Novel interaction techniques such as multi-touch screens or tabletops are presented as well with references to current research and commercially available products.

We continue with a survey on recognition techniques including template matching, principal components analysis, neural networks and ad-hoc methods for the tracking, posture recognition and trajectory matching of human gestures with references to the existing research literature as well as to available products or libraries.

1.1 Looking at Human Gestures

Gestures play an important part in our lives including art, science, music, dance, allowing us to work, communicate, express feelings, enhance and accompany speech. Using gestures is something we have been training for all our lives, still in the process of learning, and make use of it according to our personalities, jobs, social situations and events, most of the time without even realizing it. The naturalness and familiarity of gesturing are revealed even more by the fact that blind people gesture as they speak just as much as sighted individuals do, even when they know their listener is also blind [IGM98].

The vocabulary of gestures that people use can be at once informative, entertaining but also dangerous. Gestures may be instructive such as the signs made by an airport officer guiding planes or those performed by lecturers when sustaining their presentations; gestures may be warm in the form of giving a hug or a confident hand-shake; gestures may even be menacing: for example imagine two drivers that 'accidentally' met on a freeway.

Gestures express ideas, feelings and intentions, sometimes replacing words and enhancing speech. They convey information and are accompanied by content and semantics. Various psycholinguistic studies have been conducted in what concerns the understanding of gesture communication and they provide an excellent starting material for gesture studying and understanding [Ken86, McN92, Cad94, Cas96].

1.1.1 Definition

Gestures may be bluntly looked upon as physical movements of hands, arms, face and body with the intent of conveying information and meaning.

Actual definitions of gestures vary depending on the various research communities: sociologists, biologists, linguists, computer scientists (HCI, pattern recognition

experts, practitioners), etc. For example, from a biological and sociological perspective, gestures are loosely defined and thus researchers are free to visualize and classify gestures as they see fit. Biologists define gestures broadly, as in [NPL86]:

> *the notion of gesture is to embrace all kinds of instances where an individual engages in movements whose communicative intent is paramount, manifest, and openly acknowledged.*

Distinction must be made between *gesture* and *posture*. There is the tendency to capture the dynamic part in gesture while to consider posture as being static [Mul86]. Consulting a few dictionaries [Com00, Inc02], we end up with the following definitions for posture and gesture[1]:

> *Posture* (noun): 1. a. A position of the body or of body parts: a sitting posture. b. An attitude; a pose: assumed a posture of angry defiance.
>
> *Posture* (noun): 1. The position or bearing of the body whether characteristic or assumed for a special purpose: erect posture. 2. A conscious mental or outward behavioral attitude.
>
> *Posture synonims*: attitude, carriage, pose, stance. These nouns denote a position of the body and limbs: erect posture; an attitude of prayer; dignified carriage; a defiant pose; an athlete's alert stance.
>
> *Gesture* (noun): 1. A motion of the limbs or body made to express or help express thought or to emphasize speech. 2. The act of moving the limbs or body as an expression of thought or emphasis. (verb intr.) To make gestures (verb tr.) To show, express or direct by gestures

We are thus further considering posture as describing the position of body or of body parts. For example, holding the victory sign position for a certain amount of

[1] only fragments cited

time is considered to be a posture. [LaV99] sees postures as *static movements* that can be either simple or complex. For a simple posture each of the fingers is either extended or flexed but not in between, e.g. fist, thumbs up, index pointing, victory sign, while a complex posture allows for fingers to be bent at different angles other than 0 or 90 degrees, e.g. the ok sign, pinch or various sign language postures. A gesture is defined as a *dynamic movement*, such as waving good bye or describing the shape of a circle. According to [LaV99], dynamic movements are also simple and complex. Simple movements are represented by either a posture held still while changing the position and orientation of the hand or by moving fingers, i.e. changing postures. A complex movement will include changes in posture, position and orientation of the hand. Figure 1.1 illustrates the difference between our *posture* and *gesture* concepts.

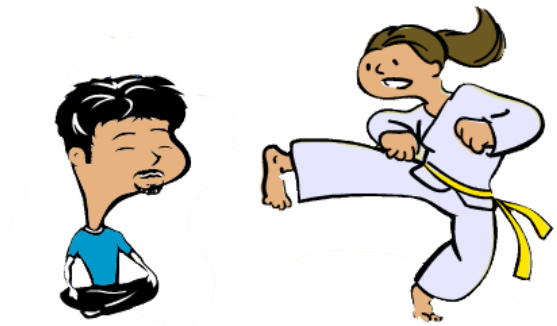


Figure 1.1: Posture as static (yoga pose) and gesture as dynamic (a karate kick). Source: http://office.microsoft.com/en-us/clipart/default.aspx

Gestures posses a variety of distinct characteristics that make them distinguishable among other types of activities that relate to the human body [Ken96] such as practical actions or postural adjustments:

- Gestures may be looked upon as excursions [Sch84] from a rest position always returning to a rest state after execution.

- They posses a peak structure with the center associated to the actual meaning of gesture.
- Gestures are well bounded: the action phrases which are perceived and identified as gestures have clear onsets and offsets.
- Gestures are symmetric: it is remarkable difficult to spot the differences of someone caught gesturing on a film that runs backwards and forwards.

Similarly to speech, gestures serve a variety of functions. They convey information to listeners [Ken94]; facilitate some aspects of memory [BB78, KMSC91]; facilitate the smoothness of interactions and increase linking between interaction partners [CB99]; communicate attitudes and emotions both voluntarily and involuntarily [GA75]; can provide insight into a speaker's mental representations [McN92].

Equally important, we want to isolate for the purpose of our discussion only those interactions for which gestures are articulated and recognized. The definition of articulated gesture as in [KH90] is more appropriate in this case:

> *A gesture is a motion of the body that contains information. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on it's way to hitting a key is neither observed nor significant. All that matters is which key was pressed.*

1.1.2 Gestures Taxonomies

Several taxonomies and classification criteria have been proposed for human gestures. We therefore start by citing and providing a small overview on commonly accepted classifications for human gestures as they may be encountered in the literature.

[Cad94] identifies three main categories by taking into account the functionality

aspect of gestures that manipulate, sense and communicate with the environment (Figure 1.2 illustrates a few examples):

- *Ergotic* gestures are associated with the idea of work and ability to model and manipulate the environment. The ergotic gesture acts directly on the environment by altering its form and properties, e.g. hand made pottery, knitting or sculpting.

- *Epistemic* gestures offer information that reveals the environment through perception of temperature, pressure, surface quality for a given object, shape, orientation, weight, and so on. The environment gets revealed through tactile experience or haptic exploration.

- *Semiotic* gestures produce meaningful informational messages for the environment and come as a result of commonly shared cultural experience. The intent is to convey information.

Figure 1.2: Ergotic (painting, manipulating the environment), epistemic (heavy lifting, sensing the environment) and semiotic (giving the ok sign) gestures. Source: http://office.microsoft.com/en-us/clipart/default.aspx

Semiotic gestures are further classified by [McN92] according to their role in communication:

- *Iconic* gestures describe an actual concrete object or event. They are closely related to the semantic content of speech, illustrating what is being said, e.g. when using hands to describe a physical item in order to show how big or small it is.

- *Metaphoric* gestures are similar to iconics but referring to abstract objects or events, depicting a general abstract idea.

- *Deictic* or pointing gestures.

- *Beat-like* which are gestures that accentuate the meaning of a word or a phrase, e.g. rhythmic beating of a finger or hand.

[Ken86] describes a gesture continuum that goes from gesticulation up to sign languages, as follows:

- *Gesticulation* or spontaneous movements of hands and arms that take place during speech and always accompany speech.

- *Language-like* gestures that represent gesticulation actually integrated into speech that replaces a word or a phrase. An example would be the following phrase: *I enjoyed eating the grapes but the cake that came after was [gesture]* where [gesture] integrates grammatically inside the phrase.

- *Pantomime* are gestures that depict objects, events or actions that may or not be accompanied by speech.

- *Emblems* or familiar gestures, e.g. the V sign for victory or thumbs up for ok.

- *Sign languages* are sets of gestures and postures that define linguistic communication systems such as ASL, the American Sign Language.

Starting from gesticulation to sign languages, the association with speech gets more and more reduced, spontaneity decreases and social regulation increases. In other words, the formalized, linguistic component of the expression present in speech is replaced by signs going from gesticulation to sign languages. This supports the idea that gesture and speech are generated by one integral. The first category, gesticulation as being spontaneous and associated with speech, represents approximate 90% of the total amount of gesturing people perform. [McN92] concluded that there is no body language but that instead gestures complement the spoken language. In Kendon's words [Ken80]:

> *the phrases of gesticulation that co-occur with speech are not to be thought of either as mere embellishments of expression or as by-products of the speech process. They are rather, an alternate manifestation of the process by which ideas are encoded into patterns of behavior which can be apprehended by others as reportive of those ideas.*

Familiar gestures are culture dependent. Very few gestures are universally understood and interpreted with the same signification. What is perfectly acceptable in the United States for example may prove be rude, inappropriate or even obscene in other cultures. For example, nodding head up and down to intend and say *Yes* actually means *No* in Bulgaria and Greece; passing an item to someone with one hand is considered to be very rude in Japan, etc.

[NPL86] consider a more detailed approach and propose a 3-level classification. With respect to the universality of gestures, the following types are identified:

- *Arbitrary* or uncommon gestures that need to be learned.
- *Mimetic* gestures, more common, usually encountered within culture.
- *Deictic* gestures similar to the classification of [McN92]. They include *specifics*

(gestures that point to a particular object), *generics* (gestures that point to a class of objects) and *function indication gestures* (that point to an object and equally indicate an action).

[Cas96] considers two categories of gestures: *autonomous* (or independent) and *natural* (or spontaneous). Autonomous gestures are not necessarily associated with verbal communication, possess fixed spatial-temporal properties and are speaker independent. Natural gestures are usually associated with speech in a conscientiously or unconscientiously manner. They are speaker dependent and much influenced by educational and cultural factors as well as by the actual situation at the moment they are produced. They were further classified by [McN92] into iconic, metaphoric, deictic and beat-like as discussed above.

The most numerous category of gestures is represented by gestures performed by hand due to the the ability of the human hand to acquire a large number of discernible configurations (the sign languages being a good example in this case). It seems very likely that the first and oldest purpose of using our hands was in order to perform actions on and to manipulate the physical world. This translates into changing an object's position, orientation, shape or any other property. [Mul86] classifies ergotic hand movements according to physical characteristics: object type, change effectuated, hands involved or indirection level.

It is more common to classify ergotic gestures according to their function, i.e. as either *prehensile* or *non-prehensile*. Non-prehensile movements include pushing, lifting, tapping and punching. Prehension may be looked upon as the application of functionally effective forces by the hand to an object for a task in the presence of various constraints. Prehensile movements are identified as being precision, power, hook and scissor grips [Nap93]. The type of grip used in any given activity is a function of the activity itself and does not depend on the shape or size of the object

to be gripped. The classification relates to the muscular-skeletal properties of the hand and merely refers to a frequently used hand movement.

1.2 Gesture Acquisition

1.2.1 Existing Technologies

The first step in order to use gestures as input for interacting with a computer system is data acquisition. Technologies for the acquisition of human gestures have been very rapidly proliferating and a great variety of trackers, pointing or whole hand or body devices are available today commercially. One main property of all these input devices is the number of DOFs (Degrees of Freedom) they posses. Data may be collected several ways [LaV99] by using:

- capture devices that are worn by the user that may provide a fine level of representation of the gesture (such as small variations when bending fingers as outputted by sensor gloves). The users are required however to wear additional equipment which may feel cumbersome and disturbing, burdening the actual interaction.

- video cameras, one or multiple, that capture a sequence of images and allow for detection of a 2D or 3D gesture. The main advantage is the feel of natural interaction but the capturing accuracy and frequency are lower than in the previous case.

- hybrid approaches that combine the above technologies.

[KS05] considers *perceptual* and *non-perceptual* input. Non-perceptual input involves the use of devices that require physical contact in order to transmit location, spatial or temporal information to the computing processor. Non-perceptual input includes: mouse and pen input, touch and press input, electronic sensing (wearable

or body mounted, gloves, sensor-embedded objects and tangible interfaces, tracking devices), audio input. On the other hand, perceptual input enables gestures to be recognized without requiring any physical contact via any input device or physical objects, allowing the user to communicate gestures without having to wear, hold or make any sort of physical contact [KS05]. Perceptual input technology includes visual, audio or motion sensors that are capable of receiving sensory input data from users directly from their actions, speech or physical location within the environment.

Taking into account the type of events the capturing devices are able to generate, one may distinguish between:

- *Discrete* input devices that generate one event at a time according to the user's need (events are fired for example when the user presses a button). Examples include the traditional keyboard, the pinch glove or the virtual tool belt. For example, considering the case of Fakespace Labs Pinch Gloves [Fak07], users will pinch two or more fingers for the device to signal an event (Figure 1.4 a). The gloves detect when two or more fingers are in contact and, after contact verification, a signal is fired. The time that elapsed between two consecutive gestures is also recorded. A complex variety of actions based on the simple pinch gesture can be programmed into applications (a pinching gesture may be used to grab a virtual object; a finger snap between the middle finger and thumb can be used to initiate an action, etc).

- *Continuous* input devices that generate a stream of events. Common examples are position / orientation trackers and data gloves.

- *Hybrid* devices that combine both discrete and continuous events. Examples include the ring mouse such as the Pegasus FreeD (Figure 1.4 b) and digital pen based tablets [Peg07].

With regards to tracking devices, one can discriminate between several technologies: magnetic, mechanical, acoustic, inertial, vision/video camera based or hybrid.

Magnetic trackers (such as the Ascension's Flock of Birds [Asc07], Figure 1.4 d) use the low-frequency magnetic field emitted by a transmitter for the receiver sensor to determine its position and orientation with respect to the magnetic source. The main disadvantage is the distortion of the magnetic field that metal or conductive metals will produce as well as the interference with nearby monitors. Mechanical trackers (such as the BOOM Tracker from Fakespace Labs [Fak07]) have as advantages accuracy and low latency. They may however be big or bulky with reduced mobility. Also, they may exhibit expensive prices.

Bolt's system Put-That-There [Bol80], the first system that implemented posture recognition, made use of such a magnetic-based space-sensing cube device attached to the user's wrist in order to capture position and orientation parameters, as Figure 1.3 illustrates. Users control simple shapes on a large display inside the Media Room by using pointing gestures and voice commands. Basic item shapes include circles, squares and diamonds with variable attributes such as color and size (large, medium, small). For example, users may point toward the screen where a small cursor is displayed and use natural voice command such as "Create a blue square there" for the creation of a new square object. Other commands are possible, for example [Bol80]: "Move the blue triangle to the right of the green square" (voice only); "Move that to the right of the green square" (voice combined with pointing); "Make the blue triangle smaller" becomes "Make that smaller" with the use of pointing; "Name that ... calendar" in order to assign names to existing objects. The rudimentary set of commands, concerning themselves with the simple management of a limited ensemble of non-representative objects, is intended to suggest the versatility and ease-of-use of voice and gesture for the management of the graphic

space.

Figure 1.3: Bolt's system Put-That-There [Bol80]

Acoustic trackers (the Fly Mouse from Logitech [Log07]) prove to be relatively inexpensive, lightweighted and with no interference with metals but they exhibit line-of-sight issues as well as sensitiveness to noise, see Figure 1.4 c. They use high-frequency sound emitted from a source placed on the object to be tracked while microphones placed in the environment pick up the signals from the source in order to determine its position and orientation. Inertial trackers (IS300 from Intersense [Int07]) use a variety of inertial measurement devices such as accelerometers or gyroscopes and have the advantage of speed, accuracy, long working range, non interference with metals as well as no need for a transmitter. Figure 1.4e shows the Intersense IS300 tracker. The sensors prove however to be bigger than the magnetic-based ones and they are likely subject of error accumulation. Also, they usually track only 3 DOFs: either position or orientation.

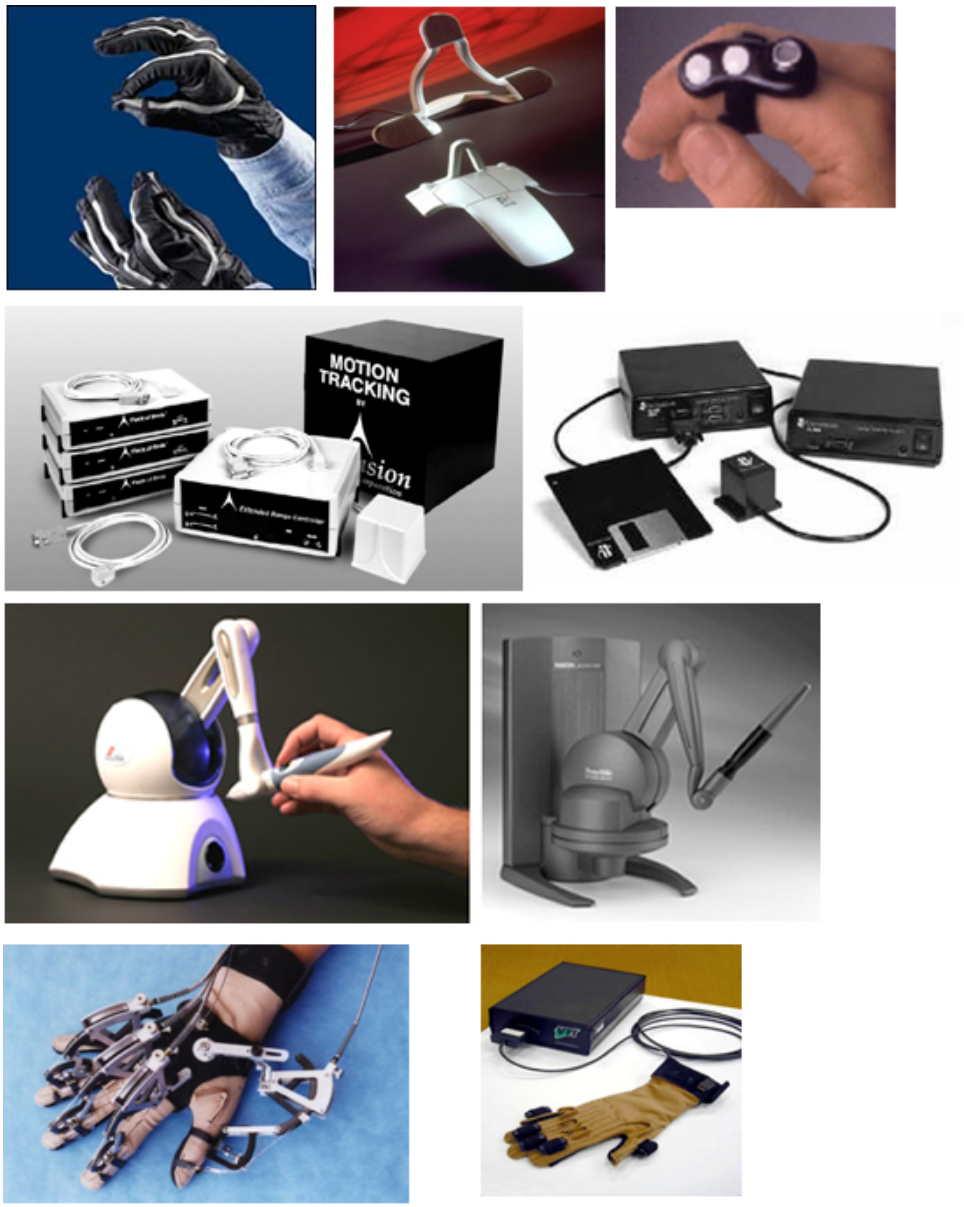


Figure 1.4: Gesture acquisition devices. From top to bottom, left to right: Fakespace Pinch Gloves [Fak07], Logitech Fly Mouse [Log07], Pegasus FreeD [Peg07], Ascension Flock of Birds [Asc07], Intersense IS300 [Int07], Phantom Omni and Phantom Desktop Haptic Devices [Sen07], Cyber Grasp [Imm07], Cyber Touch [Imm07]

Hybrid trackers (IS600 from Intersense [Int07]) have the advantage of combining multiple technologies for improving on accuracy and reducing latency but all this with a complexity cost however.

With respect to the main forms of feedback, one can classify devices into:

- *Ground referenced* (such as Sensable Phantom devices [Sen07], Figure 1.4 f).
- *Body referenced* (Cyber Grasp from Immersion [Imm07] which is a lightweighted force-reflecting exoskeleton that fits over a Cyber Glove data glove wired version and adds resistive force feedback to each finger as Figures 1.4 g and h illustrate. Grasp forces are produced by a network of tendons routed to the fingertips via the exoskeleton),
- *Tactile* (Cyber Touch from Immersion [Imm07], a tactile feedback option for Immersion's wired Cyber Glove instrumented glove. It disposes of small vibrotactile stimulators on each finger and the palm of the Cyber Glove system).

An example of a gesture recognition system using a data glove is Charade [BBL93] which allows a speaker giving a presentation to control a remote computer display with free-hand gestures while still using gestures for communicating with the audience. Figure 1.5 presents the Charade system at work. A VPL DataGlove measures the bending of each finger and the 3D orientation of the hand: when the pointing direction of the hand enters an active zone, a cursor appears on the screen and follows the hand. A gesture is detected when the user's hand is pointing in the active area while gesture segmentation is performed using start and end positions defined by wrist orientation and finger positions. There are 16 gestural commands for navigating through the hypertext and controlling the presentation (moving the hand from left to right goes to the next slide; pointing and circling an area highlights it on the screen). Recognition rates of up to 98% for trained users and 72% for first

time users were reported.

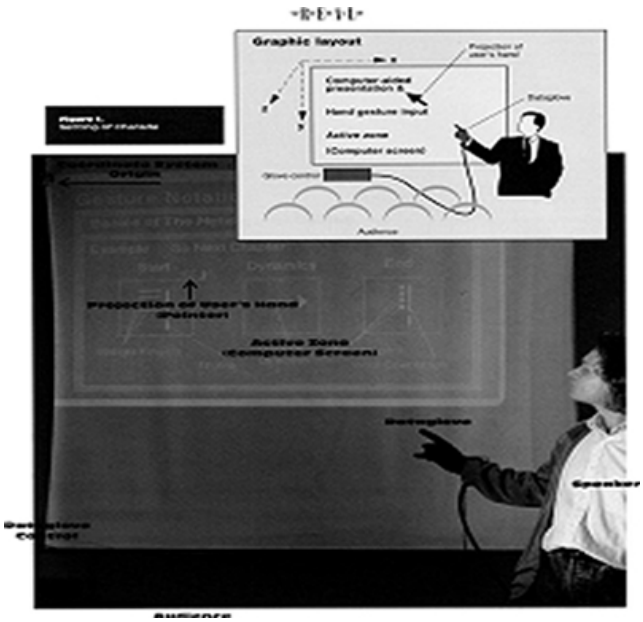


Figure 1.5: Charade: free-hand gestures are used to control the progress of a public presentation [BBL93]

1.2.1.1 Interaction for gaming

The gaming industry is a great target for gesture-based interfaces that would allow players to immerse even more into the game environments. Gesture-based game controlling was introduced by Playstation 2 in the form of EyeToy[2]. EyeToy makes use of a video camera that gets mounted on the top of the TV screen facing the player that sits in the range of a couple of meters away. The players are brought into the game and allowed to interact with various game elements by using legs, arms, head or the whole body. The processing technique is based on motion, color detection and sound.

[2]http://www.eyetoy.com/index.asp

Figure 1.6: EyeToy: players are emerged into the game. By making use of motion detection the system allows players to interact with arms, legs or whole body. Source: http://www.eyetoy.com/index.asp

Limitations of EyeToy are those common to computer vision applications: the video camera needs to be used in a well-lit room and the player must be in view in a given distance range. In order to help let the player know when there is not enough light, a red LED on the front of the camera will flash indicating too dark working conditions. Also, the gestures are limited to basic hit-like motions area-wise around the player's body.

Nintendo released at the end of 2006 the Wiimote[3] controller for the Wii console. By using its motion sensing capability implemented using accelerometer and optical sensors, the controller allows players to interact with game elements by moving, shaking or pointing. The Wiimote is multi-functional and adapts to all kinds of games and scenarios. For example, in a tennis game, it serves as the racket players swing with their arms as Figure 1.7 illustrates; in a driving game, it serves as the steering wheel; or it may act as a weapon for first-person shooters games. As feedback, the Wiimote provides basic audio and rumble functionality.

[3]http://wii.nintendo.com/controller.jsp

Figure 1.7: The Wiimote and one of its extension (top) and players using the controller during a tennis game (bottom). Source: http://wii.nintendo.com

1.2.1.2 Interaction with robots

Interaction with robots, either static or mobile, is another very active area of research powered by the general propagated common belief that future will be filled with robots that understand and respond appropriately to our gestures and speech. Commercially available smart toys exist such as Sony AIBO[4] which is able to move around, look for toys, play and communicate with the owner. Honda's humanoid robot ASIMO[5] is capable to interpret postures and gestures of humans and move independently in response. Using the visual information as aquired by a video camera mounted at the head level, ASIMO is able to detect movements of multiple objects and to estimate distance and direction. The result is moving according to indicated

[4] http://support.sony-europe.com/aibo/

[5] http://world.honda.com/ASIMO/

directions, following persons or greeting people when they approach. With regards to gestures detection and recognition, ASIMO is able to[6]: recognize an indicated location and move to that location (posture recognition); shake a person's hand when a handshake is offered (posture recognition); or respond to a wave by waving back (gesture recognition). On top, the built-in face recognizer helps remembering a number of registered persons and allows addressing them by name.

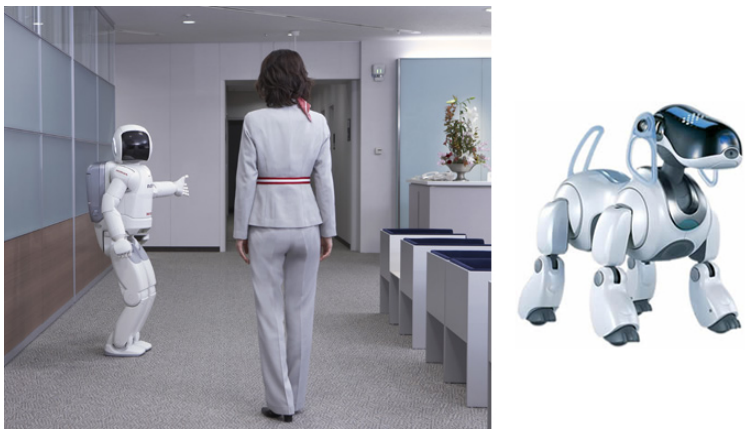


Figure 1.8: Honda's humanoid ASIMO leading the way and the four-legged companion AIBO from Sony. Sources: http://support.sony-europe.com/aibo, http://world.honda.com/ASIMO/

1.2.1.3 Multi-touch and tabletops

A particular technology for acquiring gestures makes use of touching. Touching (screens, surfaces, etc) is intuitive and presents several advantages, the most important being the direct contact as well as the haptic feedback the user immediately receives.

[Han05] introduced a simple and inexpensive technique for enabling multi-touch sensing at high resolution for interactive surfaces based on a technology called frus-

[6] http://world.honda.com/ASIMO/technology/intelligence.html

trated total internal reflection. [DL01] describe a technique for creating a touch-sensitive device that allows multiple users to simultaneously interact. Their surface generates modulated electric fields at each location that are capacitively coupled to receivers installed in the work environment.

Multi-touch may be implemented using computer vision as well. Wilson's system Touchlight [Wil04, Wil05b] uses image processing techniques in order to combine video frames acquired from two cameras. The IR video cameras are placed behind a semi-transparent plane facing the user together with a projector and mirror system as Figure 1.9 illustrates. By combining the distortion-corrected information from the two video sources, detection of objects that touch or are in a short distance of the surface plane is achieved.

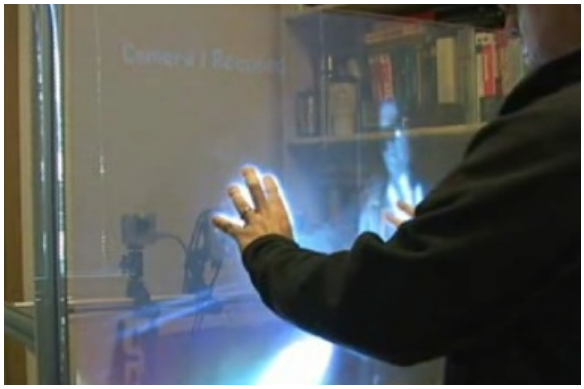


Figure 1.9: TouchLight: gesture interaction with input/output surface [Wil05b]

Multi-touch interaction is available commercially in the form of the Microsoft Surface[7]. Surface computing is an intuitive way to interact with digital content and approaches the tabletops interaction paradigms [WB03, Wil05a, WR06]. Currently applications include browsing photographs, playing videos, listening to music, viewing map locations or ordering menus. Interesting interactions are achieved by

[7] http://www.microsoft.com/surface/

connecting to external devices such as digital cameras or bluetooth mobile phones in order to exchange photographs [WS07].

Figure 1.10: Microsoft Surface. Source: http://www.microsoft.com/surface/

1.2.2 Video-based Systems

Camera-based or vision trackers make use of video information and video-based processing to achieve face, hand, fingers, arm or whole body tracking as Figure 1.11 illustrates.

The main advantage that comes with vision gesture acquisition and which provides the comfortable feeling of natural interaction is the fact that the technology is non intrusive and does not require users to wear additional equipments or devices. Users may interact freely with the system with no need for wearing or interacting via an additional device that may distract, restrict or burden the natural movement (e.g. wires more or less heavy attached to gloves or hand-wearable trackers, glove sizes that may be a bit small or large). Of equally importance, vision-based solutions are relatively inexpensive compared to trackers that exhibit a price range from several hundreds to tens of thousands dollars.

Figure 1.11: Video based gesture acquisition. Left: hands detection using a top-view mounted video camera. Right: face detection from live video television stream.

1.2.2.1 Skin Color for Hands Detection

Color is an important feature that has been intensively used for hands detection. Instead of wearing distinctly colored gloves or rings [Dor94, ZCF+04] or using LEDs or any other marker systems [SZ94], skin color detection has proven to be an important intermediate step for face and hands tracking algorithms [Ang01, CJH01, VSA03].

It has been observed on large image datasets [JR98, COB02, COB03] that skin color clusters in predefined intervals in several color spaces. Based on this comfortable property, a few approaches have been proposed: histograms of skin probability at different resolutions [JR98], single or mixtures of Gaussians modeling [COB02], elliptical [LY02], adaptive methods [VGD+05] or various curved and linear polygonal segments for skin cluster delimitation. A common conclusion is that skin color indeed clusters under known limits in several color spaces.

Using skin color as a feature may be divided into three subproblems: the choice of an appropriate color space, a method of modeling the skin color distribution and choosing the way segmentation gets performed - using independent pixel values or considering neighborhood's values as well.

There are several advantages when using skin color detection: invariance to

orientation and size of the object being tracked (e.g. hands) and low computation time. Skin processing also comes with a few drawbacks such as the dependence on illumination - skin appears different under different illuminations hence skin color depends on the scene context; skin color varies from person to person (Caucasian, African, Asian, etc.); many real life objects present the same color as skin, i.e. the color of skin (or its reflectance) is not unique and does not pertain to skin only (particularly when captured with low spectral resolution observers like a RGB camera).

A very simple and low cost procedure is to filter the current video frame using simple static thresholds in a given color space, for example, the intervals $[h_{low}, h_{high}]$ and $[s_{low}, s_{high}]$ for the hue and saturation components in the HSV color space.

1.2.2.2 Challenges in Video Acquisition

There are also drawbacks when it comes to processing video information for detecting and recognizing gestures, many of which are commonly encountered when it comes to computer vision applications:

- there is the dependency on the environment that translates into several issues: time varying lighting conditions; video cameras settings; users skin color; background in motion. All these issues burden the task of detectors which must prove to be stable, robust and continuously adapting to the environment.

- a calibration stage before using the system is required, usually related to color or point of view correction.

- there are sometimes constraints on the gesture dictionary in what regards the postures the system is able to successfully recognize. For example, fingers may occlude themselves while in movement in a single camera view; hands may

partially or totally occlude fingers or they can be as well occluded by other objects from the scene.

- video processing is CPU intensive in the conditions where real-time processing is a must for a natural interaction. There is a great demanding of processing power especially if multiple cameras are involved and there is strong dependency on computer vision algorithms which are very much currently under development (relatively new field of study). Latest research on GPGPU (General Purpose computing on the Graphics Processor Unit) demonstrated the use of the processing power of the GPU in order to run and execute highly-intensive CPU demanding computer vision algorithms [FM04]. OpenNVIDIA [FMA05] is an open source library that implements vision algorithms on computer graphics hardware using OpenGL and Cg, available at[8].

- technology and processing requirements limitations: video resolution may not be sufficient for detecting high fidelity movements of fingers for example; 25 fps of ubiquitous video capture devices may not be enough to capture quick hand movements (hand is quicker than the eye and or the capture device in this case).

- portability is an issue for most vision systems that require still placements of the video cameras. Nonetheless, mobile solutions exist and they consider wearable video computing (laptops with built-in cameras, head mounted displays and cameras [KTH04]) but they come with all the problems related to wearable equipments.

Special considerations for using video-based acquition solutions relate to:

[8] http://openvidia.sourceforge.net/

- the number of cameras to be used (whether if 2D or 3D gestures are targeted or depth or stereo information is needed).
- cameras placement in space should not affect the visibility of body parts that are performing the gesture.

Commonly employed techniques include: motion detection and motion flow following, color detection (and particularly skin color detection as a preprocessing stage for hands or face detection), feature based detectors (e.g. Haar-like features), non standard trackers (e.g. flock of birds) all combined with pattern recognition methods and techniques.

1.3 Recognition Techniques

The section gives an overview on the current state-of-the-art in gesture recognition including detection, tracking, pose and trajectory recognition techniques. The focus is mainly oriented on gestures performed with one or two hands in a video-based acquisition scenario although references to other body gestures or capture devices are equally mentioned. [Wat93, PSH97, LaV99, TSW90] provide excellent reviews on gesture recognition techniques. Also, good introductory courses on sketching exist such as [JJL06, LaV07].

Segmentation is the process of partitioning an image into multiple regions according to some predefined criteria, the goal being of separating the objects of interest from the background. Popular approaches take into consideration color, contour or custom defined local neighborhood image features in order to identify the objects of interest. Tracking is the process of locating an object in time in a sequence of video frames. There are several possible approaches to tracking such as blob, contour or visual features tracking.

1.3.1 Tracking

There are two main streams of research encountered in the literature with regards to hand tracking, *model-based* and *view-based* approaches [PSH97]. Model-based approaches make use of articulated 3D models of the hand that get projected onto the image plane. An error function is computated between the parameters of the model and various image features and the model parameters are adapted in correspondence with the the minimization of a certain cost functional. The view-based approaches use sets of features which are associated with a certain hand pose and common classifiers are trained from a previously collected database of feature samples. The set of features are searched for within the image, usually at multiple scales, looking for a high classifier output. The approach proves useful when the number of hand poses and the feature set is small.

Kolsch and Turk introduce a fast hand tracking technique that uses a flock of KLT features together with a learned foreground color distribution [KT04b, KT05]. The tracker was implemented for wearable computing as Figure 1.12 illustrates.

Figure 1.12: Hand tracking using the flock of birds and wearable computing. Source: http://www.movesinstitute.org/~kolsch/HandVu/HandVu.html

The KLT features, named after Kanande, Lucas and Tomasi [ST94], are based on the observation that a steep brightness gradient along at least two directions makes for a promising feature candidate to be tracked over time. A flock of features contains multiple instances of KLT features whose locations are updated independently from one video frame to another: they move to a new location for which the highest match correlation is found. The features don't follow a uniform direction, some might be lost while others may venture far from the flock. The concept of a flock inforces additional conditions with regards to features distribution: no two features must be closer to each other than a threshold distance and no feature must be farther away from the feature median than a second threshold distance. The tracker is fast, robust against background noise and has the ability to track objects that undergo rapid changes in orientation or deformations. An implementation is available under the HandVu library [KHD04, Kol].

Haar-like features as introduced by Viola and Jones [VJ01] for rapid object detection have been used for tracking and detection of several hand postures [KT04c, KT04a]. The Haar-like features are defined for gray-levels images as differences between the pixel intensities of various rectangle areas. A learning process will find the features that perform best on the training set and combine them into more powerful classifiers using Ada boosting methods. To achieve fast-real time detection, the classifiers are arranged in a cascade structure for which the first levels will discard most of the wrong candidates. The method is fast and combined with image pyramids structures assures size invariant object detection as well.

When only motion is important, the hands postures information may be discarded which takes the pressure off the tracking algorithms. Simple hand detection may be done using color tracking such as following colored gloves [Dor94] or bright

LEDs. Atlas Gloves[9], see Figure 1.13, is a physical interface for controlling 3D mapping applications. The user interface is a pair of gloves that have LEDs attached. The LEDs are used to track intuitive hand gestures like grabbing, pulling, reaching and rotating. A video camera translates each LED-enabled gesture into a set of possible actions: pan, zoom, rotate and tilt. Atlas Glove is open source software.

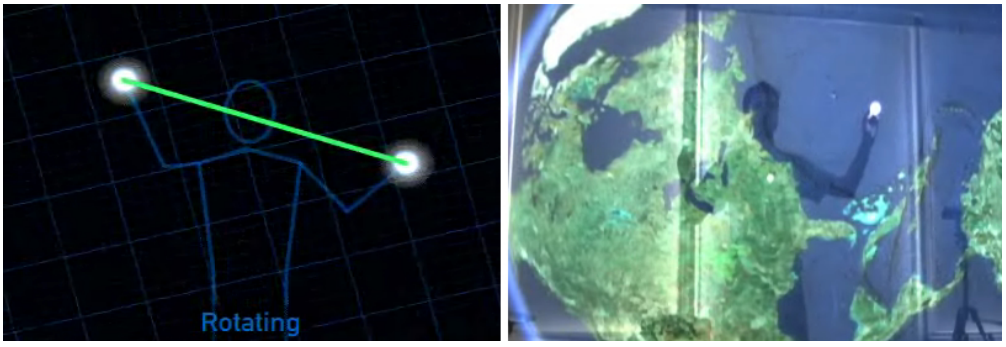


Figure 1.13: Atlas Glove: controlling Google Earth using simple and intuitive motion gestures. Source: http://atlasgloves.org/

Although tracking techniques are useful as they offer the possibility of continuously knowing the position, orientation or other parameters on the object of interest, they may not always be the best approach for acquiring gestures. For example, Wilson [Wil06] presents a very intuitive method of interacting by means of gesture using the pinch posture or the TAFFI interface (Thumb and Fore Finger). A camera placed on the top of the computer screen detects the pinch gestures, i.e. when the thumb and the fore fingers of the user's hand touch as Figure 1.14 illustrates.

The tracking algorithms are avoided by only paying attention to the moment when the actual gesture takes place, i.e. when the user performs the pinch posture. Detection of the pinch posture is achieved using blob analysis by identifying the blob that fits the specific constraints of the region between the thumb and the fore

[9] http://atlasgloves.org/

Figure 1.14: Thumb and Fore Finger Interface [Wil06]

finger. A good contrast between the user's hand and the background (represented by the keyboard in the majority of cases) is mandatory. Also the user is required to maintain the pinch posture as accurately as possible in the horizontal plane so that a blob could form between the two fingers.

Active shape models or smart snakes have been designed for exactly locating a feature in an image when given an initial guess. A contour, which is roughly the shape of the feature to be located is placed on the image at the feature approximate location. The contour is attracted to edges that are located near and its parameters change deforming it so that in the end it will converge to the actual shape of the feature. The process is iterative, moving and deforming the contour. The technique can be extended to tracking features across video frames: the position and shape of the feature in one frame is used as an approximation for the next frame. [HS95, CKJK05] use active shapes for tracking hands in real-time video.

1.3.2 Pose Recognition

1.3.2.1 Template matching

Template matching is one of the simplest techniques that have been used for recognizing hand postures [ZLB+87, Stu92, Wat93, New93]. Template matching determines if a new posture can be classified to a number of class templates that have been previously computed from training sets. The classification part is implemented by computing a distance measure to each template. Commonly encountered measures are the sum of absolute differences or city-block distance and the sum of squares.

Freeman and Weissman [FW94] demonstrate a TV-set control application using the template matching technique. Only one posture is used: the open hand facing the video camera. A hand icon on the computer screen display follows exactly the movements of the user's hand. The system thus exploits visual feedback: users see the icon hand on the screen so they know how much to move their hands as Figure 1.15 illustrates.

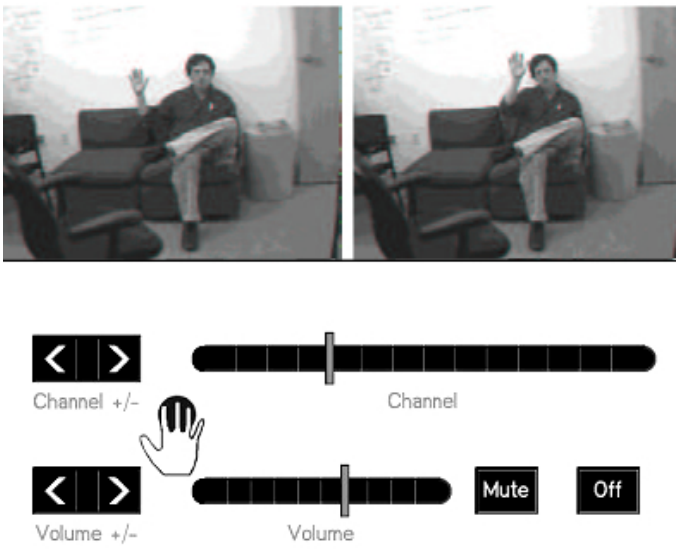


Figure 1.15: Controlling a TV-set using hand gestures [FW94]

The gesture is triggered when the user holds up the open hand facing the television which determines the TV to enter control mode. Leaving the control mode is achieved by closing the hand. Various graphical controls are adjusted using movements of the hand. The problems they address are how to provide a rich set of commands without training or memorizing complicated gestures and how to achieve command recognition in a complex visual environment. There are many possible commands to give the television ("mute", "channel 37", "louder") yet no universal set of hand signal to specify all of them. Voice is not appropriate for channel surfing nor for changing parameters by incrementers such as volume control.

One particular application for which posture recognition plays an important role is the automatic understanding of sign language. [Kad96] report recognition of 95 samples from the Australian Sign Language with an 80% accuracy. Postures are acquired using a Power Glove and matching is performed on features such as bounding boxes, distance in time and histograms on the x, y, z coordinates.

The technique of template matching may be efficient for a relative limited set of postures and it is simple to implement. It doesn't work well however for large sets of postures due to overlapping templates [Wat93].

1.3.2.2 Principal component analysis

Principal component analysis (PCA) is a statistical technique for reducing the dimensionality of a data set that contains interrelated variables while retaining as much of the variation in the dataset as possible. It is the purpose of principal components analysis to derive new variables, in decreasing order of importance, that are linear combinations of the original variables and are uncorellated. The new variables, called principal components, are given in decreasing order of importance so that the first variables contain most of the variation present in the original variables. Eigenvectors and eigenvalues are computed for the original data set and the eigen-

vector with the highest eigenvalue holds the highest variance. Similarly put, PCA produces an orthogonal coordinate system in which the axes are ordered in terms of the amount of variance in the original data for which the corresponding principal components account [Web02].

[BMM97] use PCA in order to achieve recognition of 25 gestures from the international hand alphabet and report 99% recognition accuracy for 1500 test images for which 1000 images were used during the training process. [DT02] use a combination of PCA/MDA (multiple discriminant analysis) for the recognition of a 100-signs vocabulary.

1.3.2.3 Neural networks

Neural networks have been intensively used for gesture recognition [MT91, BF92, BE92, FB93, FH93, FH95, VA95]. Common examples include multi-layered perceptrons, radial basis functions or Kohonen networks. Neural networks are previously learnt using a training set consisting of both positive and negative samples. For feed-forward networks having differentiable activation functions there exists a powerful and computationally efficient training method called error back-propagation for finding the derivatives of an error function with respect to the weights and biases of the network [Bis95].

[MT91] present a posture recognition system using recurent neural networks for a finger alphabet of 42 symbols of the Japanese sign language. Recognition rates varies between 77.0% and 98.0% depending on the number of training patterns and the structure of the training and testing sets.

Glove-TalkII [FH95] translates hand gestures to speech using an adaptive interface. Hand gestures are mapped continuously to 10 control parameters of a speech synthesizer, allowing the hand to act as an artificial vocal tract that produces speech in real-time. The gesture-to-speech task is divided into vowel and consonant pro-

duction by using a gating network to weight the outputs of a vowel and a consonant neural network.

1.3.3 Trajectory matching

Gestures do not consist in posture information only but they have motion trajectories associated as well. For example, describing the shape of a circle in free air using the hand may represent a gesture command for the creation of a circle or sphere-like object in a virtual environment. Trajectory recognition is an important problem and it relates to other fields such as handwritting or signature recognition or general shape recognition. The difficulty of shape recognition in general, be it gesture, handwritting or signature is represented by the variability that is present within the data: users execute the same shape/gesture differently with each execution. Providing robust algorithms that would take into account these variations whilst providing good accuracy is a considerable challenge.

Gesture trajectories, as acquired using a given capture technology, are represented by an array of 2D or 3D points $\{p_1, p_2, \ldots p_n\}$. The data may be processed directly for recognition or other features may be extracted. Feature extraction and analysis usually consists in processing the low-level information from the raw data in order to produce higher-level information in order to help the further representation and recognition levels.

Rubine's GRANDMA system [Rub91] performes recognition of 2D strokes using a set of 13 features such as: sine and cosine of the initial angle of the gesture; the length and angle of the diagonal of the bounding box; distance between first and last point; cosine and sine of the angle between the first and the last point; the total gesture length; total angle traversed; sum of the absolute value at each point; sum of the squared angles; maximum speed (squared) of the gesture; time duration of the gesture. Classification is done using a linear discriminator and reported recognition

rates are above 98%.

Simple trajectory matching of mouse gestures is supported under the Mozilla Firefox browser, Mozilla Thunderbird email client and Chatzilla. Figure 1.16[10] lists a few gesture types that Mozilla is able to recognize. The gestures are captured by holding down a mouse button (usually the right one) and moving the mouse in a certain pre-defined way (simple horizontal, diagonal or vertical strokes) in order to form a gesture after which the mouse button is released.

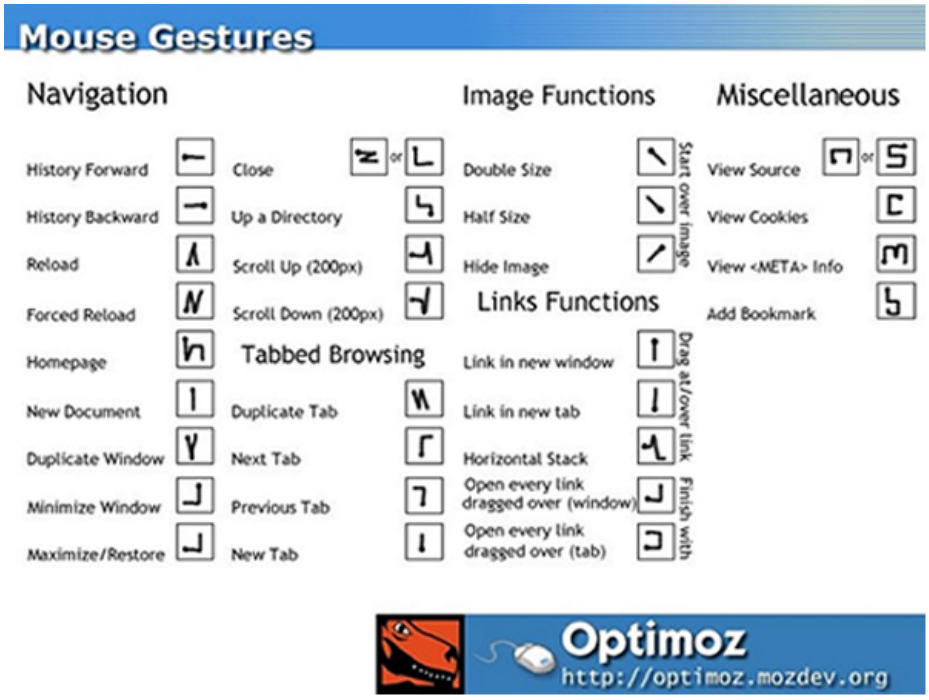


Figure 1.16: Mouse gestures available under the Mozilla Firefox browser. Source: http://optimoz.mozdev.org/gestures/

Mozilla gestures are combinations of URLD strokes (up/right/left/down) and 012 (left/middle/right mouse buttons). Mouse gestures operations include: working

[10] http://optimoz.mozdev.org/gestures/

with history (backward/forward); working with windows and tabs (new window, minimize, next tab, new tab, previous tab, close); scroll (up, down) and zoom; working with images (double size, half size, hide image) and working with links (link in new tab); miscellaneous (view source, add bookmark). The source code is also available[11].

[WWL07] introduce and make available to the HCI community an implementation of a simple yet robust algorithm for gesture trajectory matching[12]. They intitle their classifier a "$1 recognizier" as it is easy, cheap and may be implemented in about 100 lines of code. The accuracy attained over a set of around 4000 gesture samples was over 97% with only 1 loaded template and 99% accuracy with 3+ loaded templates.

Several programming libraries offer APIs that support gesture recognition. An example is the Siger library[13] available for download[14] for Microsoft Tablet PCs which comes with a few pre-defined gestures as Figure 1.17 illustrates. The significance of gestures is, from top-down and in left-right order: scratch-out - erase the content; triangle - inserts; square and star - action item; check - check-off; curlicue - cut; double curlicue - copy; circle is application specific; double circle - paste; left semicircle - undo; right semicircle - redo; caret - paste, insert; inverted caret - insert; chevron left and chevron right are application specific.

Each gesture comes with a hot point such as the starting point for circle, star, rectangle, a distinguished hot point for curlicue or double curlicue, corner for check

[11] http://optimoz.mozdev.org/gestures/

[12] http://faculty.washington.edu/wobbrock/proj/dollar/

[13] http://msdn2.microsoft.com/en-us/library/aa480673.aspx

[14] http://sourceforge.net/projects/siger/

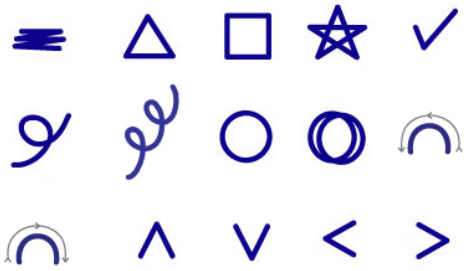


Figure 1.17: Extracts of pre-defined gesture templates available as part of the Siger library for Microsoft Tablet PCs. Source: http://msdn2.microsoft.com/en-us/library/aa480673.aspx

or apex for carets and chevrons. Also, executions are guided by notes[15] for each gesture type: "The upward stroke of the check must be two to four times as long as the smaller downward stroke" for the check gesture, "Draw the star with exactly five points. Do this in a single stroke without lifting the pen" for star or "Draw both sides of the chevron with equal length. Make sure the angle is sharp and that the point is not rounded to a curve" for the chevron right gesture.

Other library examples include SATIN [HL00, HLLM02], a java based toolkit for informal ink-based applications[16] or LipiTk [MVK06], an open source toolkit for online handwriting recognition[17]. LipiTk contains generic algorithms for the recognition of isolated handwritten shapes captured using TabletPCs, PDAs, stylus-equipped SmartPhones or external tablets and notes taking devices. The toolkit reports recognition accuracies of 97.94% for numerals, 93.35% for english upper and 88.34% for english lower characters and is available for Windows and several Linux distributions.

[15] http://msdn2.microsoft.com/en-us/library/ms818591.aspx

[16] http://dub.washington.edu/projects/satin/

[17] http://lipitk.sourceforge.net/

Chapter 2

Catching Movement: A Vision-based Approach

The problem with video is that
it gives you a thousand possibilities.
Lars von Trier
(born 1956)

The chapter addresses the problem of visual detection of gestures performed by one or both hands by discussing several specific details with regards to the acquisition scenario, hands detection, posture recognition and retrieving of motion trajectories.

We start by describing the acquisition scenario that we use through all of our experiments: users sit in front of a desk while a top-mounted video camera permanently monitors their hands watching for both postures and motions. The users receive a visual feedback of their actions on the monitor screen located at the opposite end of the interaction desk. The setup approaches somewhat the same scenarios specifics to tabletops and assures a non-fatigation interaction due to the comfortable sit position whilst the arms may rest as well on the surface of the desk.

Hands detection over the surface of the interaction table is further discussed [VGD+05] together with a simple method for recognizing hand postures using neural learning [VPC+06]. The chosen classifier was a multi layered perceptron with three

layers. Details are given on selecting the architecture of the neural classifier as well as on the recognition accuracy levels. A few hand postures were selected in accordance with several commonly encountered operations for interacting and working with virtual objects [VP06] such as: selection, translation, rotation and change in scale. The operations are executed by using either one or both of the hands. An interaction algorithm together with a simple demonstrative application for performing such simple operations on virtual objects are discussed. By keeping all the processing at a low level of complexity and by considering an appropriate control of the environment, we obtain a real-time 25 fps functional system with high detection and recognition accuracy results.

We can highlight the contributions of the chapter as: fast real-time hand detection technique in a non-fatigation interaction scenario; simple and robust method for the recognition of several hand postures for commonly encountered operations in virtual environments; proposal of an interaction algorithm for working with virtual objects using one or both hands.

2.1 Scenario Setup

The working scenario is presented in Figure 2.1: users sit in front of an interaction desk while a top-mounted video camera monitors the working area as well as the users' hands. Visual feedback of hands detection, command recognition and result of the command action are given on a monitor screen facing the users at the opposite end of the table. The scenario approaches somehow tabletops systems [Wil05a, WB03] but with the visual output being displayed on the facing monitor screen. Users sit in a comfortable position in front of the desk watching the output of their commands as well as the feedback of the hands detection on the monitor screen. Hands rest on the surface of the desk which reduces the fatigue factor that may intervene for longer working time intervals. The scenario is similar to others that users are already accustomed to such as working with real objects on a desk or typing at a keyboard while watching the results on the monitor screen.

Video capture is carried out at a resolution of 320x240 and 25 fps. The working desk has an homogeneous background with a good contrast with respect to the skin color which allows for a fast and accurate segmentation of the user's hands. The video camera auto controls the brightness and exposure settings. A snapshot of the camera viewing angle is presented as well in Figure 2.1.

2.2 Hands Detection

Hands segmentation over the color of the interaction desk is achieved using a simple low-cost skin filtering algorithm in the HSV color space on the hue and saturation components:

$$pixel\ p\ is\ skin \Leftrightarrow hue(p) \in [h_{low}, h_{high}] \wedge sat(p) \in [s_{low}, s_{high}] \qquad (2.1)$$

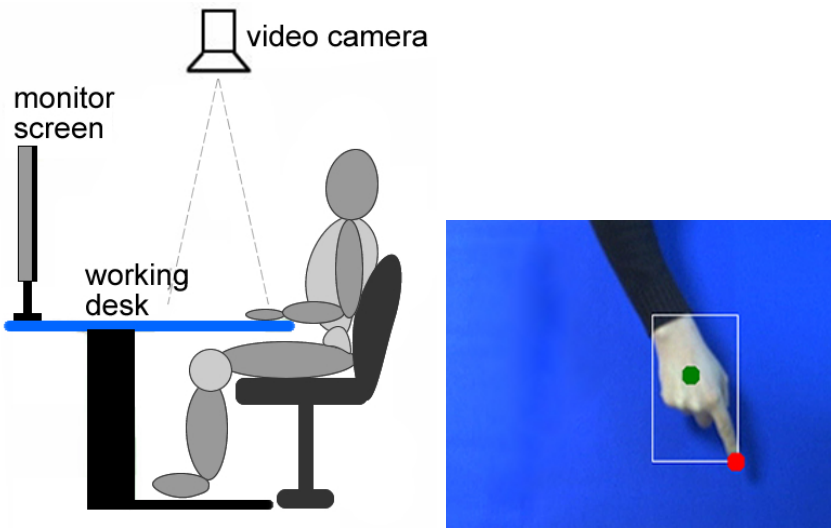


Figure 2.1: The working scenario: gestures are captured over the surface of a table using a top-mounted video camera. Left: system overview. Right: camera top view.

where p is the current pixel submitted to classification and $[h_{low}, h_{high}]$ and $[s_{low}, s_{high}]$ are the lower and upper thresholds for the hue and saturation components. The technique is very fast with a complexity order of $O(n)$ where n is the dimension of the video frame. Also, the method assures a very accurate hands segmentation under the previously mentioned working conditions. Segmentation results are given in Figure 2.2.

The values for the hue and saturation thresholds corresponding to skin color were chosen experimentally as $h_{low} = 10$, $h_{high} = 100$, $s_{low} = 0.07$ and $s_{high} = 0.6$ where hue varies from 0 to 359 and saturation from 0 to 1. Figure 2.3 illustrates the distribution of skin color in the Hue-Saturation space for a set of consecutive video frames which allows experimentally retrieving of the color thresholds. The hue component has been shifted right with 60 degrees in order for the h_{low} and h_{high} limits to respect $h_{low} < h_{high}$ as otherwise skin hue clusters at the two limits of the hue range interval [0..359]. The HSV color space was chosen because it allows for

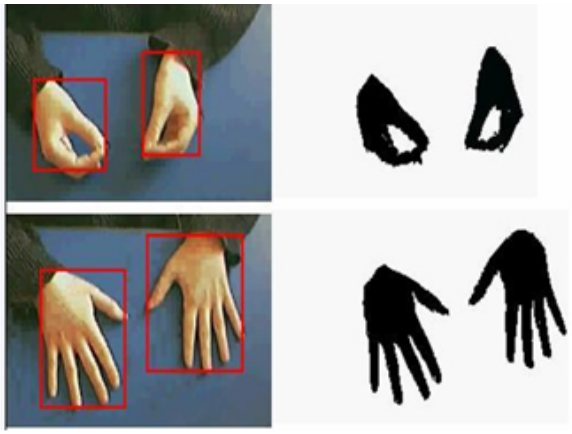


Figure 2.2: Hand segmentation results (segmentation is performed in the HSV color space by applying simple filters on the hue / saturation components).

separation of the brightness influence as given by the value component. Ignoring the brightness information and analyzing only hue and saturation leads to better color segmentation results (the same brightness value for example is scattered through all the red, green, blue components of the RGB color space).

The hands segmentation filter outputs a binary image for which a blob detection algorithm must be applied [Rus98, Pra07]. At the end, a set of blobs are detected $\{blob_1, blob_2, \ldots blob_n\}$ which must be filtered against area and geometry ratios in order to only keep the two hands. The area and size filters $Width\,(blob_i) \in [Width_{min}, Width_{max}]$, $Height\,(blob_i) \in [Height_{min}, Height_{max}]$ and $Area\,(blob_i) \in [Area_{min}, Area_{max}]$ make sure that only the blobs that fit the hands restrictions on width, height and area are kept. Also, the aspect ration $AR = height/width$ is used as well in order to further filter the blobs based on their bounding rectangle shape: $AR\,(blob_i) \in [AR_{min}, AR_{max}]$. At the end, only the blobs that respect the hands dimensions and ratios are considered for further processing. The algorithm for filtering the hands from the set of blobs is given below.

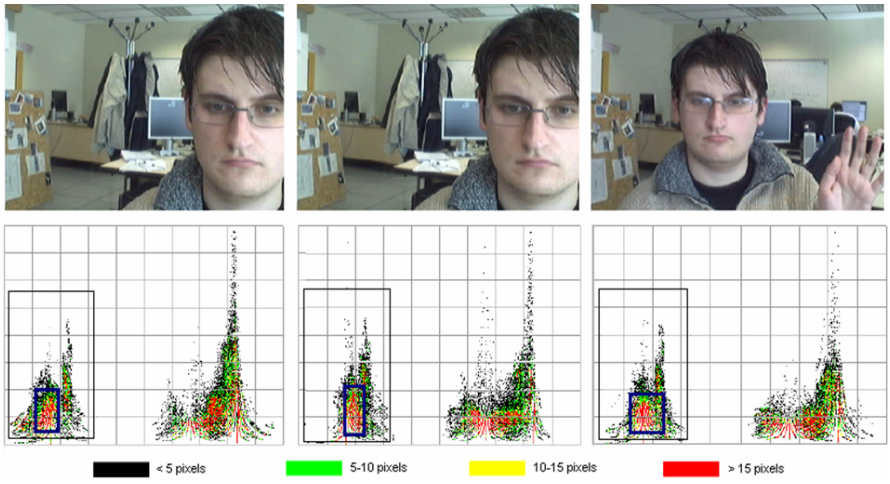


Figure 2.3: Three consecutive video frames and their associated 2D hue/saturation color histogram. The hue component varies from 0 to 359 and saturation from 0 to 1. Hue has been shifted to the right with 60 degrees.

FILTER-HANDS($blobs$)

```
1  for i = 1 to length[blobs]
2       do
3          if WIDTH(blobs[i]) ∉ [Width_min, Width_max]
4             then blobs ← blobs − blobs[i]
5          if HEIGHT(blobs[i]) ∉ [Height_min, Height_max]
6             then blobs ← blobs − blobs[i]
7          if AREA(blobs[i]) ∉ [Area_min, Area_max]
8             then blobs ← blobs − blobs[i]
9          if ASPECTRATIO(blobs[i]) ∉ [AR_min, AR_max]
10            then blobs ← blobs − blobs[i]
11 return blobs
```

The actual threshold values were determined experimentally as follows: $Width_{min} = 30$, $Width_{max} = 50$, $Height_{min} = 30$, $Height_{max} = 120$, $AR_{min} = 1.2$, $AR_{max} = 4.5$, $Area_{min} = 1000$, and $Area_{max} = 10,000$.

Extra processing includes blob rotation so that the blob's longest axis should be parallel to the vertical axis as see Figure 2.4 illustrates. Axis alignment is performed

by computing the two axis of the ellipse of inertia having the same area and center of mass as the hand blob.

Figure 2.4: Rotation of blob objects for automatic alignment with the vertical axis.

2.2.1 Posture Recognition

For the purpose of performing posture recognition, we represent each hand blob using a 6×5 matrix with real values in the [0..1] interval for which each value represents the percentage of pixels of the blob that lie in the corresponding cell as given by the row and column indexes. Figure 2.5 illustrates the concept.

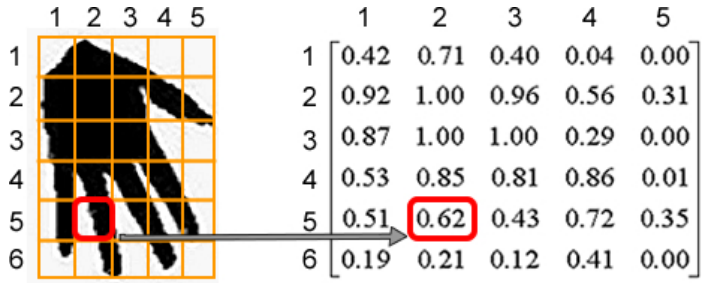


Figure 2.5: Representation of a hand blob as a pixel occupancy matrix.

Posture recognition is performed using a Multi-Layered Perceptron organized using a 3-layer structure of 39 neurons $(30 - 4 - 3)$ [VPC+06], as follows:

1. The 1st layer consists of 30 input neurons, each coding the hand blob using 6 x $5 = 30$ real normalized values in the interval [0..1]. Each value at position (i, j) represents the percentage of pixel occupancy for the the rectangle cell (i, j),

$i = 1, 6$ and $j = 1, 5$ as per Figure 2.5. The choice of the 6×5 structure was experimentally found as giving the best performance together with the selected number of neurons for the hidden layer, as shown by experimentations results displayed in Figure 2.7.

2. The 2nd layer uses 4 hidden neurons. Experiments showed that 4 neurons in the hidden layer offer the best performance on our testing set, see Figure 2.7 as well.

3. The 3rd layer is composed of 3 output neurons, each providing a real value in the [0..1] interval representing the probability of appearance of one out of the 3 hand postures at the entries of the network. The hand postures we selected are illustrated in Figure 2.6 and they correspond to simple manipulation operations on virtual objects as further discussed in the chapter.

Figure 2.6: Selected hand postures.

A database of 1665 hand postures samples was constructed, out of which 70% (1166 samples) were chosen as the training set and 30% (499 samples) made up the testing set. Data was acquired from a single subject using the scenario conditions as presented. The training algorithm we used was the basic backpropagation [Bis95] with no particular enhancements. We motivate our choice for the network architecture as being the minimal structure with respect to the number of neurons in the hidden layer that achieved the accuracy threshold of 98% of correct classification on the testing set. We started from small networks architectures and iteratively

incremented both the number of neurons from the input as well as from the hidden layer. Figure 2.7 shows the experimentations performed on the architecture of the neural network while Figure 2.8 gives the training error for the chosen $(30 - 4 - 3)$ configuration.

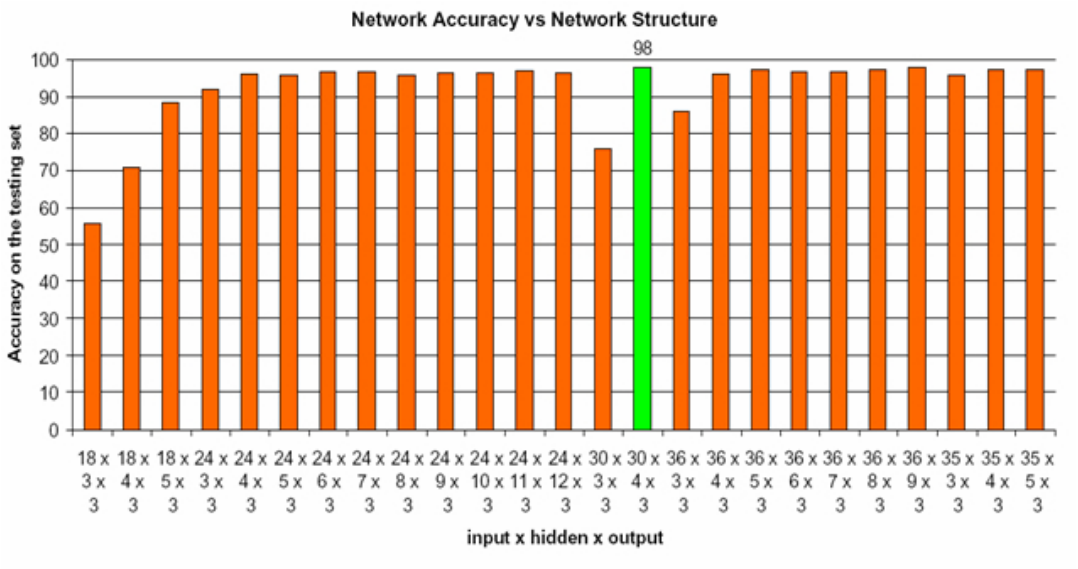


Figure 2.7: Accuracy on the testing set vs several choices for the neural network architecture.

2.2.2 Postures-Based Command Application

In order to test the performance of our classifier, a simple posture-based command application was implemented. The application allows one or two-handed manipulation of virtual objects. We selected three hand postures by considering a few common operations encountered when interacting with virtual objects such as: selection, translation, rotation and resize [VP06, VPC+06] as Figure 2.9 illustrates:

- Selection is the operation used to indicate the object for which all the future processing will apply to. Performing selection gets done using the first hand

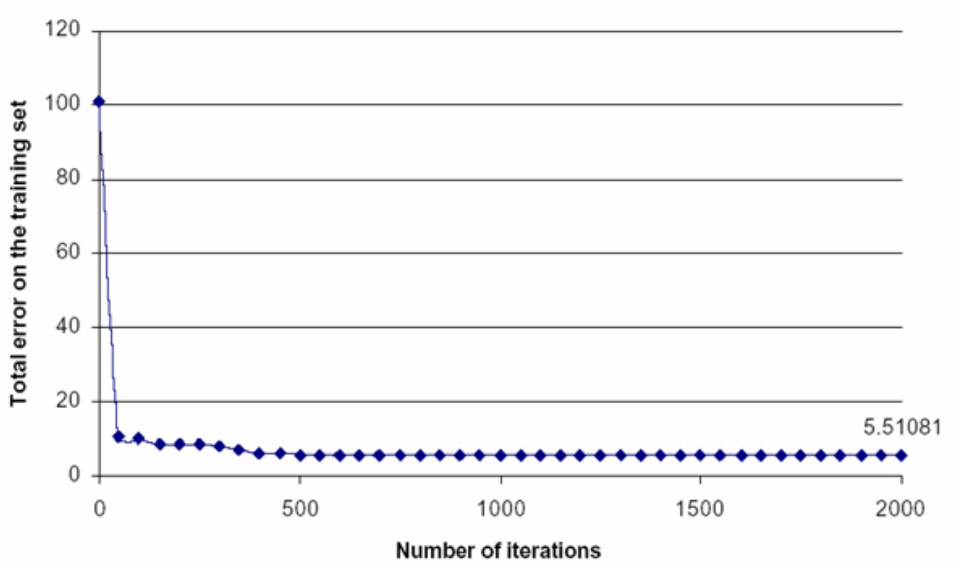


Figure 2.8: Total error on the training set vs number of iterations.

posture (index finger pointed) maintained for a short period of time in the small vicinity of the target object. The operation is performed using only one hand.

- Translation is the operation that allows moving the object in the virtual scene. It is performed using one hand while holding the *pinch* posture: thumb and index fingers are touching creating a small gap between them whilst the other fingers are closed.

- Rotation is performed with two hands, both holding the *pinch* posture. Each hand targets a point on the virtual object which determines rotation to be performed around the axis as indicated by the two points.

- Change of scale is an operation performed with two hands as well, each hand targeting a point on the virtual object while maintaining the *open hand* posture: all the fingers are open and the hand lays perpendicular to the table.

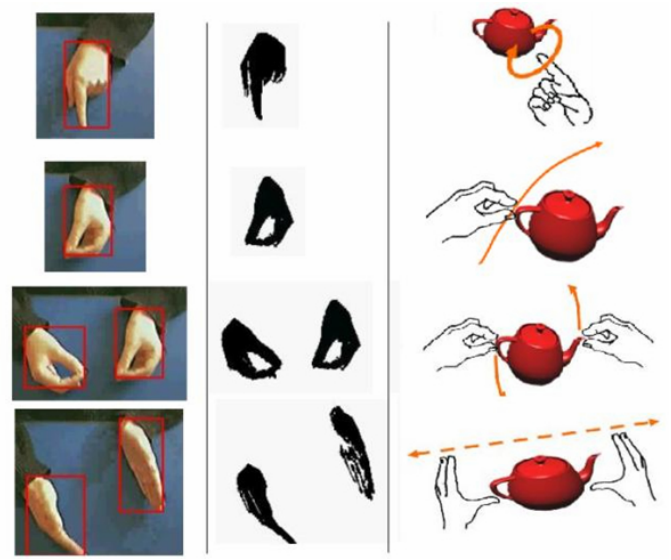


Figure 2.9: Set of hand postures and associated commands for interacting with virtual objects. First column: video snapshots, 2nd column: hands segmentation, 3rd column: corresponding action in the virtual environment.

For each hand posture a selection point is defined as presented in Figure 2.10. It is this point that allows for the actual interaction to happen. The selection points are computed as the closest points to the top-left corner of the bounding box for the *select* and *pinch* postures and middle-left corner for the *scale* posture. The interaction is always performed on the object that is closest to the selection point, according to the current hand posture. The algorithm is given below in pseudocode language formalism.

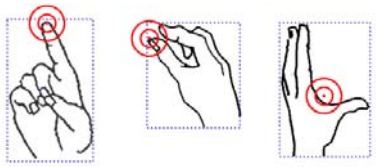


Figure 2.10: Selection points for each hand posture are used for the actual interaction.

POSTURES-BASED-INTERACTION()

```
1  while image ← ACQUIRE-IMAGE-FROM-VIDEO-CAMERA()
2        do
3           blobs ← SEGMENT-IMAGE-AND-RETRIEVE-BLOBS(image)
4           blobs ← FILTER-HANDS(blobs)
5           hand_left, hand_right ← CLASSIFY-HANDS-USING-MLP(blobs)
6           ▷ If no hands are detected
7           ▷ grab and process another video frame
8           if hand_left = nil AND hand_right = nil
9              then goto 1
10          object_left ← GET-OBJECT-CLOSE-TO(hand_left)
11          object_right ← GET-OBJECT-CLOSE-TO(hand_right)
12          ▷ If there are no objects in the range of the hands
13          ▷ grab and process another video frame
14          if object_left = nil AND object_right = nil
15             then goto 1
16
17          ▷ Determine the interaction scenario: is it selection?
18          if TYPE(hand_left) = SELECT-POSTURE
19             then SELECT-OBJECT(object_left)
20          if TYPE(hand_right) = SELECT-POSTURE
21             then SELECT-OBJECT(object_right)
22
23          ▷ Determine the interaction scenario: is it translation?
24          if TYPE(hand_left) = PINCH-POSTURE
25             then TRANSLATE-OBJECT(object_left, X(hand_left), Y(hand_left))
26          if TYPE(hand_right) = PINCH-POSTURE
27             then TRANSLATE-OBJECT(object_left, X(hand_right), Y(hand_right))
28
29          ▷ Is it two-handed interaction?
30          if TYPE(hand_left) = TYPE(hand_right) AND object_left = object_right
31             then
32                  if TYPE(hand_left) = PINCH-POSTURE
33                     then
34                          ROTATE-OBJECT(object_left, AXIS(hand_left, hand_right))
35                  elseif TYPE(hand_left) = SCALE-POSTURE
36                     then
37                          SCALE-OBJECT(object_left, AXIS(hand_left, hand_right))
```

Hands classification is achieved using the Multi-Layered Perceptron. An interesting observation is that the hand postures are symmetrical with respect to the vertical axis which leads to the possibility of discriminating between the left and the right hand. Figure 2.11 presents the symmetry of the hand postures. Each of the 3 output neurons of the network outputs a value in the [0..1] interval that represents the probability of the posture of being selection, pinch or scale. The maximum value outputted by the three neurons which overpasses a given threshold of 0.7 determines the hand posture. The pseudocode for the classification algorithm is given below.

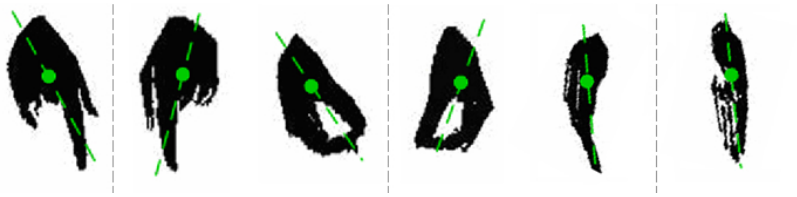


Figure 2.11: Postures performed by the left and right hands are symmetrical with respect to the vertical axis.

CLASSIFY-HANDS-USING-MLP($blobs$)

```
1  hand_left ← nil, hand_right ← nil
2  for i = 1 to length[blobs]
3        do
4           posture, probability ← MLP(blobs[i])
5           postureReversed, probabilityReversed ← MLP (REVERSE(blobs[i]))
6           if probability < 0.7 AND probabilityReversed < 0.7
7             then goto 3
8           if probability > MAX(probabilityReversed, PROBABILITY(hand_right)
9             then hand_right ← blobs[i]
10          if probabilityReversed > MAX(probability, PROBABILITY(hand_left)
11            then hand_left ← blobs[i]
12 return hand_left, hand_right
```

The interaction always takes place on the object that is closest to the hand in a given interaction area with the center being the selection point of each posture

as they were illustrated in Figure 2.10. A few snapshots as captured while running the demonstrative application are illustrated in Figure 2.12 including selection, translation, rotation and resize.

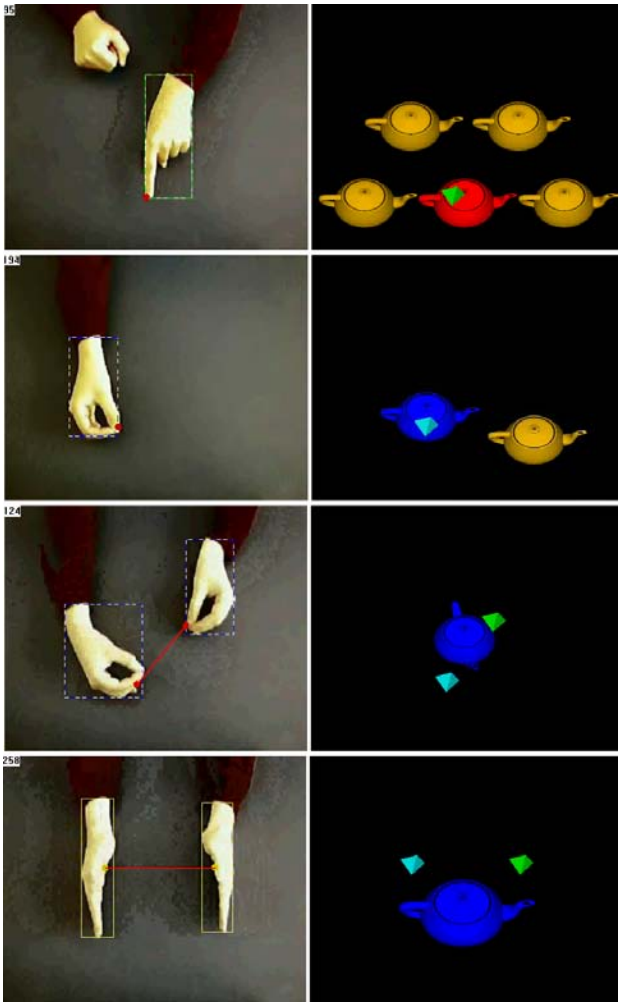


Figure 2.12: Snapshots captured while running the demonstrative application, illustrating selection, translation, rotation and change in scale.

2.2.3 Acquisition of Motion Data

Hand postures represent the static information of gestures as discussed in the previous chapter. Apart from postures, motion is equally important in the description and meaning of a gesture. As one may look at a dynamic gesture as a point moving in time, we consider for our top-view video camera scenario the motion generator point to be associated with the top of the forefinger while it is pointed. Figure 2.13 visually illustrates the acquisition of a trajectory motion describing the shape of a circle.

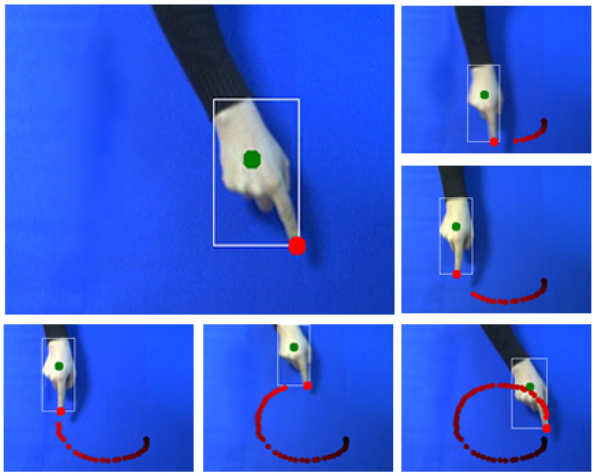


Figure 2.13: Gesture trajectories correspond to the top of the forefinger while pointed.

Motion data is acquired whilst the forefinger of the hand is pointed. Detection of the pointed forefinger is achieved by monitoring for the *select* posture using the MLP as discussed in the previous section. The index finger allows for several variations to be implemented. For example, capturing a motion gesture may start when the forefinger is stretched and ends when it is retracted back. This method may be very well applied in order to acquire the exact trajectory of a given gesture, as it is

perceived by the user and for which specific start and end timestamps are needed. The user is thus in control by designating in time the actual gesture to be recognized and interpreted by the system. Pointing/retracting the index finger may simulate as well the traditional mouse click in a similar scenario like the one introduced by the TAFFI interface [Wil06]. In the end, irrespectively whether the gesture trajectory is precisely indicated by the user with click-like hand signals or auto-segmented from a longer motion (see subsequent chapters), the motion component of gesture will be represented as a time-ordered series of 2D points:

$$motion = \left\{ p_i = (x_i, y_i) \in R^2 / i = 1, n \right\} \tag{2.2}$$

Chapter 3

Constructing Semantics: A Formalism for Gestures and Gesture Dictionaries

It is interesting thus to follow the intellectual truths of analysis in the phenomena of nature. This correspondence, of which the system of the world will offer us numerous examples, makes one of the greatest charms attached to mathematical speculations.
Pierre-Simon Laplace
(1749 – 1827)

We introduce and discuss in this chapter several models for gestures and dictionary of gestures from the point of view of the human computer interaction domain. We equally follow and discuss the multiple representations that gestures take at the various levels of processing for an interaction system.

By considering the amounts of static and dynamic information that are needed in order to sufficiently and completely describe a gesture command, we have identified four distinct types of gestures that we referenced as simple static, complex static, simple dynamic and complex dynamic. The four types range from static postures held for a period of time through sequences of postures, motion trajectories and sequences of consecutive or parallel motions. Mathematical formulations as well as examples and discussions are given for each of the four gesture types. We provide

a general definition of gesture as command for HCI that involves the characteristics of the four specific types.

We further develop on top of our gesture definitions and consider aspects related to gesture dictionaries. We introduce measures of similarity and dissimilarity for gestures that we use in order to define several dictionary types. We define gesture dictionaries as sets of interesting gestures for which there exists a dissimilarity measure such that any two gestures are considered different with respect to the given measure. Various definitions, propositions and discussions are given with regards to the mathematical formalization of gestures and gesture dictionaries [VP08].

In the end of the chapter, we discuss the multiple representations that gestures as commands take at various levels in human computer interaction systems: they are raw streams of data at the acquisition level, become mathematical models for analysis and processing and event triggers in high-level programming languages. We discuss the various issues that are encountered at each level of representation.

We can highlight the contributions of the chapter as: start-up grounds for a *gesture-as-command* theory for Human Computer Interaction including models for gesture and gesture dictionaries; an overview of gesture representations at various levels of processing in information systems, as they are of interest for HCI.

3.1 Defining gestures for HCI

When one looks at or considers gestures from the interaction point of view, the notion of gesture may be thoroughly captured into a mathematical formalization. In order to achieve this purpose, we need to look at the two components that gestures present: the posture or the static information and the associated motion trajectories or the dynamics. Depending on the interaction purpose and needs, a gesture command may be fully and completely described by a simple posture for example when confirming an application inquiry by holding the "thumbs up". Other interactions are described by solely the motion information while the posture may not be important such as when describing the shape of a circle or drawing an "M" for menu. Complex tasks may require complex gestures which in turn may take several forms: sequences of postures, sequences of motions or combining motion with postures at key moments.

We start by defining posture as a n-valued array of measurements.

Definition 3.1 *We understand by posture a set of measurements* $p = (p_1, p_2, \dots p_n)$ *from given values domain* $p_i \in D_i$ *that describe the pose of body or of body parts at one instant of time.*

Definition 3.2 *Let* P *be the set of all postures* $P = \{p/p = (p_1, p_2, \dots p_n), p_i \in D_i\}$.

The number of measurements being performed was denoted by n while p_i represents the value of the ith measurement or feature. The features may be quantitative (continuous, discrete) or qualitative (nominal, ordinal). D_i may be a real-valued domain such as $\Re$, $\Re^k$ or $[0..1]^k$. For example, we may refer to hand posture as the relative position and orientation of fingers at one instant of time. Consequently, we may choose our measurements for describing a hand posture to be real valued such as angles between fingers [WS99], relative distances between adjacent fingers,

hand orientation [BCD06], moments of different order [LaV99]. We may as well use qualitative variables such as $\{true, false\}$ for given predicates [EGG+03].

Interaction is performed using both motion trajectories and postures hence dynamic and static information. We may thus provide the following classification of gesture commands with regards to their structural pattern, i.e. the amount of posture / motion considered for performing the command:

- *simple static*
- *complex static*
- *simple dynamic* and
- *complex dynamic gestures.*

Simple static gestures (g_{ss}) are gestures that convey the desired information only through the use of a single posture that is maintained for a certain amount of time. They may be defined as the pair:

$$g_{ss} = (posture, time) \in P \times [0, \infty) \tag{3.1}$$

where ss stands for static simple and P represents the set of all postures. Figure 3.1 illustrates the confirmation or acknowledge gesture as an example of g_{ss}.

We also define the set of the simple static gestures.

Definition 3.3 *Let G_{ss} be the set of all simple static gestures:*
$G_{ss} = \{g_{ss} / g_{ss} = (posture, time) \in P \times [0, \infty)\}$.

For completeness purposes we consider the empty gestures set $\Phi \subset G_{ss}$ as the set of simple static gestures for which the time component is 0:

$$\Phi = \{g_{ss} \in G_{ss} / g_{ss} = (posture, 0)\} \tag{3.2}$$

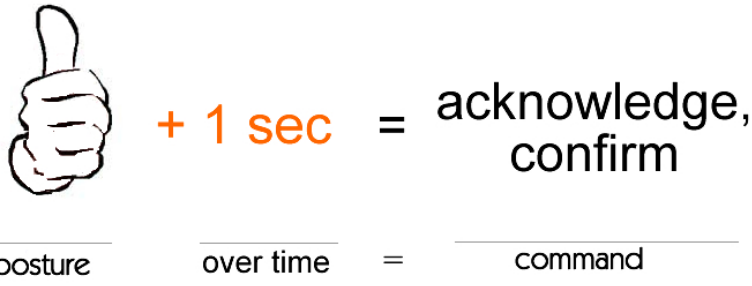


Figure 3.1: Example of a simple static gesture: the "thumbs up" posture held for a period of say 1 second may be associated with user acknowledging in response of an application confirmation inquiry.

Definition 3.4 *Any element* $\phi \in \Phi$ *is an empty gesture.*

Complex static gestures (g_{sc}) are gestures that are represented by a series of consecutive postures which are maintained for certain amounts of time. Again, only posture information is sufficient for grasping the meaning of the gesture command. They may be defined as a sequence of simple static gestures:

$$g_{sc} = \left\{ g_{ss}^1, g_{ss}^2, \ldots \right\} \tag{3.3}$$

Equivalently, the complex static gesture may be defined as a subset of the partitions set of G_{ss}: $g_{sc} \subset \Psi \{G_{ss}\}$. Figure 3.2 illustrates the "change scale" gesture as an example of g_{sc}.

A complex static gesture may be reduced to a simple gesture if the set order $|g_{sc}| = 1$. Also, a complex gesture may include at limit an infinity number of postures.

Simple dynamic gestures (g_{ds}) are gestures for which the posture information is not important as all the meaning lies within the underlying motion trajectory. As an example, Figure 3.3 illustrates the "undo" command. A simple dynamic gesture may be defined as a function of time (either continuous or discrete) having as values

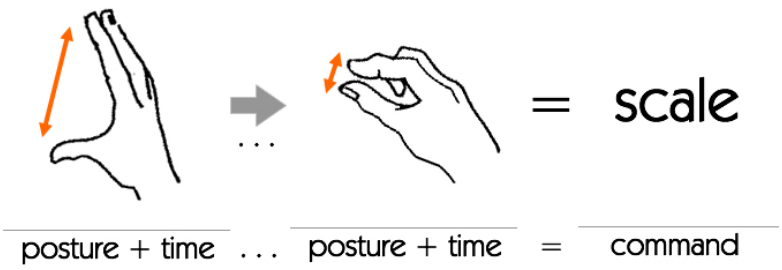


Figure 3.2: Example of a complex static gesture: the scaling gesture executed with one hand indicates a change in size or equivalently, a zoom operation that would be proportional to the distance between the index and thumb fingers. The gesture consists of several hand postures ranging say from a large distance to a small/null one between the user hand's index and thumb fingers.

the coordinates in R^d of the motion trajectory:

$$g_{ds}(t) : R \rightarrow R^d \tag{3.4}$$

where ds stands for dynamic simple and d is the dimension of motion coordinates space. d may be 2 for planar motions, 3 for gestures executed in space or it may have an additional dimension for the time component.

In a similar manner with G_{ss} we consider the set of simple dynamic gestures.

Definition 3.5 *Let G_{ds} be the set of all simple dynamic gestures:*
$G_{ds} = \left\{ g_{ds} / g_{ds}(t) : R \rightarrow R^d \right\}$.

Complex dynamic gestures (g_{dc}) are gestures that are represented by a series of consecutive motions which may be separated by periods of pause. Again, only the motion information is sufficient for grasping for which both the motion trajectory and posture are equally important for grasping the meaning of the meaning of the gesture command. They may be defined as a sequence of simple dynamic gestures:

$$g_{dc} = \left\{ g_{ds}^1, g_{ds}^2, \ldots \right\} \tag{3.5}$$

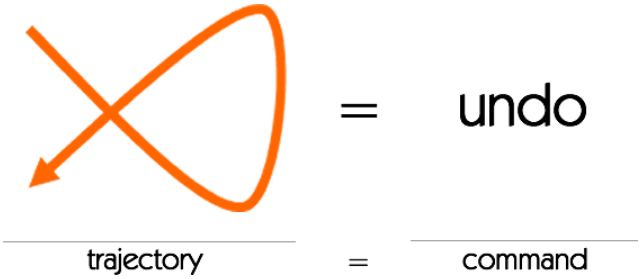


Figure 3.3: Example of a simple dynamic gesture: the gesture represented by an X-cross may be associated with performing of an "undo" operation, closing the current window or a cut operation in an editing scenario.

Equivalently, the complex dynamic gesture may be defined as a subset of the partitions set of G_{ds}: $g_{dc} \subset \Psi\{G_{ds}\}$. An example of a complex dynamic gesture would be drawing a rectangle for example that may be done by separately drawing the 4 sides or by specifying it using the both hands as Figure 3.4 illustrates.

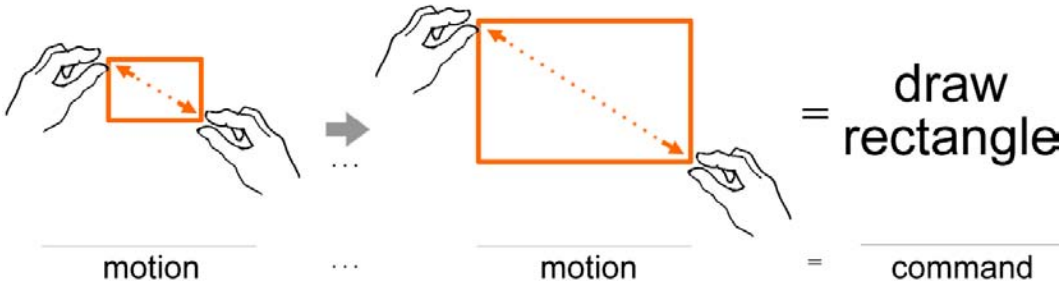


Figure 3.4: Example of a generalized dynamic gesture: specifying a rectangle shape may be performed by two hands that control the distance between two opposite corners.

We finally arrive at a general definition of gesture.

Definition 3.6 *A gesture g is a sequence of functions of time with values into the Cartesian product of the coordinate's space R^d and the set of all postures P:* $g = \left\{g_1(t) : R \to R^d \times P, g_2(t) : R \to R^d \times P, \ldots\right\}$.

Definition 3.7 *The cardinality of a gesture g is given by the number of functions in the gesture set and is denoted by $|g|$.*

The function components of a gesture may be in sequence for example when drawing the four sides of a rectangle one after another or may very well overlap in time for example when moving both hands in order to control the zoom and orientation of a geometrical object. In the later case, each hand will follow its motion trajectory and the orientation and size of the object would be proportional to the distance between the hands. The gesture in this later case is completely defined by the two motions of the hands and eventually their posture information.

As an example, Figure 3.5 illustrates the "drag & drop" gesture that consists in both posture and motion information with a cardinality of 1. Figures 3.6 and 3.7 illustrate the drawing and resizing of the rectangle examples with cardinalities of 4 and 2 respectively.

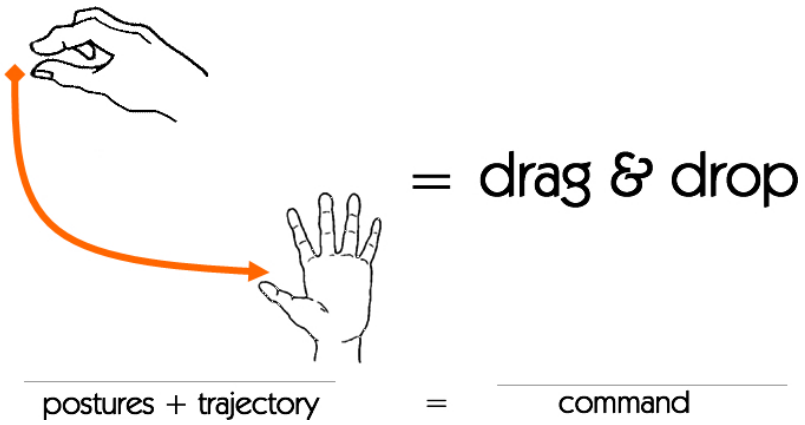


Figure 3.5: Example of a gesture of cardinality 1: the "drag & drop" operation may be implemented using two hand postures ("grab" and "release") and a motion trajectory necessary for the start and end locations.

We can derive from the general representation of gesture all the particular structural types by restricting the dimension d of the coordinates space, the set of postures

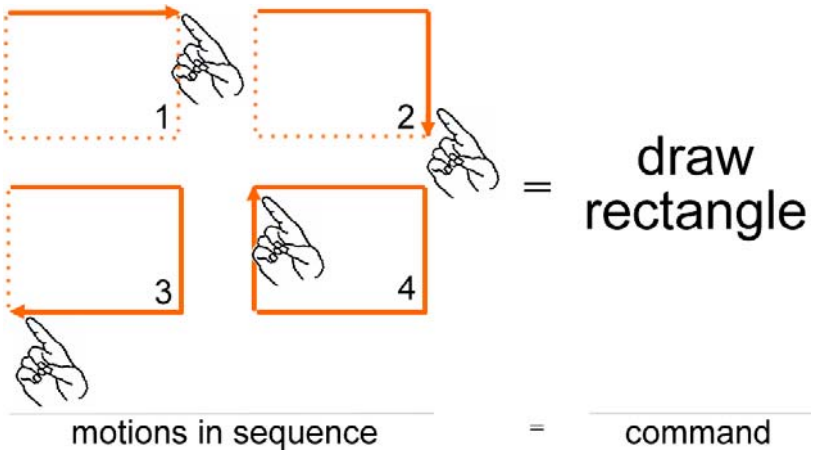


Figure 3.6: Example of a gesture of cardinality 4 with motions in sequence: drawing a rectangle may be performed by consecutively drawing its four sides (not necessarily in the order presented).

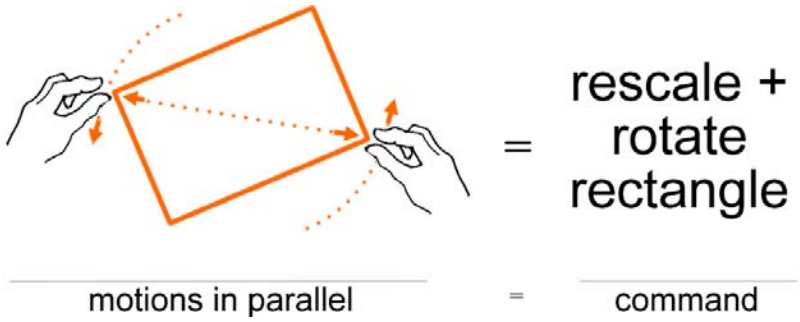


Figure 3.7: Example of a gesture of cardinality 2 with motions in parallel: rescaling and rotating a rectangle may be done by controlling two opposites corners.

P and the cardinality of the gesture g:

- $|g| = 0$ gives as the empty gesture
- for $|g| = 1$, $d = 0$ and $P = \{p_0\}$ the simple static gesture $g_{ss} = p_0$ is obtained
- for $|g| = 1$ and $d = 0$ we get the complex static gesture g_{sc}
- $|g| = 1$ and $d > 0$ provides for the simple dynamic gesture g_{ds}
- $d > 0$ gives the complex dynamic gesture g_{dc}.

Figure 3.8 illustrates graphically the four structural gesture types with regards to the amount of posture and motion information as well as their relations of inclusion.

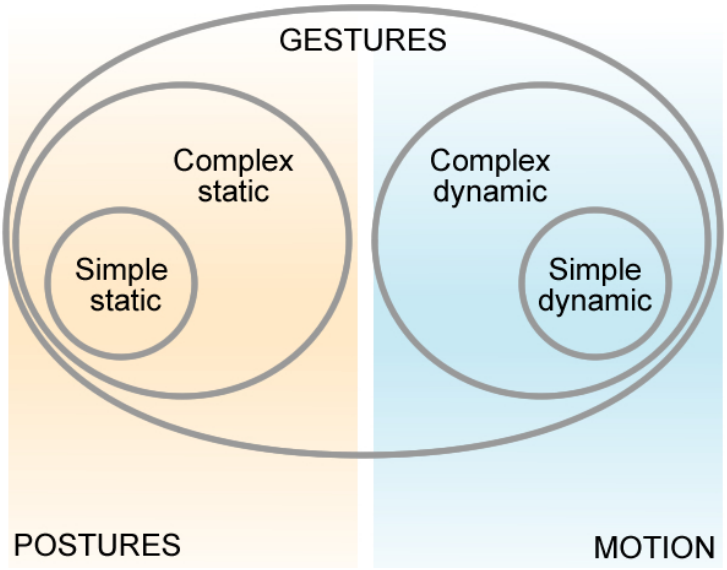


Figure 3.8: Gesture types function of the amount of posture and motion information included in their structure.

3.2 Gesture dictionaries

Definition 3.8 *Let G be the set of all gestures.*

Due to the fact that gesture is defined as a function over the whole range of time values, we further introduce the restriction of this function to a limited time range. This will allow us to select the "interesting" part of gesture, i.e. the moment when the gesture actually begins (for example the "grab" posture as in Figure 3.5) and the moment the gesture ends (the "release" posture), moments that match the start/stop sequence as perceived by the user.

Definition 3.9 *Let $g \in G$ be a gesture and $a, b \in R, a \leq b$ two real values. Then $g|_{a,b}$ is the restriction of gesture g on the $[a, b]$ time interval.*

The restriction of a gesture to a time interval allows for further definitions of amplitude of gesture $g(t)$, Ag and the interesting part of gesture, $intg$.

Definition 3.10 *Let $g \in G$ be a gesture. Let $a = inf\{t \in R/g(t) \notin \Phi\}$ and $b = sup\{t \in R/g(t) \notin \Phi\}$. Then $Ag = b - a$ is the time amplitude of g and $intg : [0, Ag] \rightarrow R^d \times P$, $intg = g(t + a)$ is the interesting part of g.*

Definition 3.11 *Let $intG$ be the set of all interesting gestures,*
$intG = \{\tilde{g} \in G/\exists g \in G$ so that $intg = \tilde{g}\}$.

The amplitude and the interesting restriction let us define identical gestures as follows.

Definition 3.12 *Two gestures $g, h \in G$ are identical and we denote $g = h \Leftrightarrow Ag = Ah$ and $intg = inth$.*

We further introduce similarity and dissimilarity measures over the set of all interesting gestures as well as two propositions that allow for transforming a similarity function into a dissimilarity one and vice-versa.

Definition 3.13 *Let* $s : intG \times intG \rightarrow R^+$. *s is a similarity measure over intG if the following conditions are met* $\forall g, h \in intG$*:*

a) $s(g, h) = s(h, g)$

b) $s(g, g) = s(h, h) \geq s(g, h)$

Definition 3.14 *Let* $d : intG \times intG \rightarrow R^+$. *d is a dissimilarity measure over intG if the following conditions are met* $\forall g, h \in intG$*:*

a) $d(g, h) = d(h, g)$

b) $d(g, g) = 0$

The definitions of similarity and dissimilarity functions are symmetrical hence one may turn a similarity measure into a dissimilarity one and vice versa.

Proposition 3.15 *Let* $s : intG \times intG \rightarrow R^+$ *be a similarity measure over intG and* $max(s) = max\{s(g, h)/g, h \in intG\}$. *Then* $d : intG \times intG \rightarrow R^+, d(g, h) = max(s) - s(g, h)$ *is a dissimilarity measure over intG.*

Proposition 3.16 *Let* $d : intG \times intG \rightarrow R^+$ *be a dissimilarity measure over intG and* $max(d) = max\{d(g, h)/g, h \in intG\}$. *Then* $s : intG \times intG \rightarrow R^+, s(g, h) = max(d) - d(g, h)$ *is a similarity measure over intG.*

A set of interesting gestures and a dissimilarity measure permits definition of gesture dictionaries.

Definition 3.17 *Let D be a set of interesting gestures,* $D \subset P\{intG\}$ *and* $d : intG \times intG \rightarrow R^+$ *a dissimilarity measure over intG. Then D is discriminative of degree 1 with respect to D if* $d(g, h) > 0 \ \forall g, h \in D$.

The degree 1 of discriminative power for a set D simply restricts that there are no two gestures in the set that are similar with respect to a dissimilarity measure. The definition below is more powerful and restricts the set D so that there are no two gestures in the set g, h where h would be similar to any part of g.

Definition 3.18 *Let D be a set of interesting gestures, $D \subset P\{intG\}$ and $d : intG \times intG \rightarrow R^+$ a dissimilarity measure over $intG$. Then D is discriminative of degree 2 with respect to D if $d(g, h|_{a,b}) > 0 \ \forall g, h \in D \ \forall a, b \in [0, Ah]$, $a \leq b$.*

The next proposition states an inclusion level between the sets of discriminative degrees 1 and 2.

Proposition 3.19 *Let $d : intG \times intG \rightarrow R^+$ a dissimilarity measure over $intG$, $D^1|_d$ be the set of all sets of interesting gestures that are discriminative of degree 1 with respect to d and $D^2|_d$ be the set of all sets of interesting gestures that are discriminative of degree 2 with respect to d. Then $D^2|_d \subset D^1|_d$.*

We may now define a gesture dictionary.

Definition 3.20 *Let D be a set of interesting gestures $D \subset P\{intG\}$ and $d : intG \times intG \rightarrow R^+$ a dissimilarity measure over $intG$. The pair (D, d) is a gesture dictionary if D is discriminative of degree 1 with respect to d.*

Due to the fact that gestures are functions of time and considering the restrictions of HCI systems that need to process, recognize gestures and provide feedback in real time, we introduce the sequentially gesture dictionary below.

Definition 3.21 *Let D be a set of interesting gestures $D \subset P\{intG\}$ and $d : intG \times intG \rightarrow R^+$ a dissimilarity measure over $intG$. The pair (D, d) is a sequentially gesture dictionary if D is discriminative of degree 2 with respect to d.*

3.3 Gesture Representations in HCI

The complex nature of gesture recognition makes human gestures have different representations at different levels. Three distinct processing layers may be identified (see Figure 3.9) corresponding to the actual gesture execution in the real world, acquisition process, modeling and application. We follow the gesture in all its representations starting from the raw format as provided by an acquisition device to the final actual interaction stage (where gesture triggers action) at the highest application level.

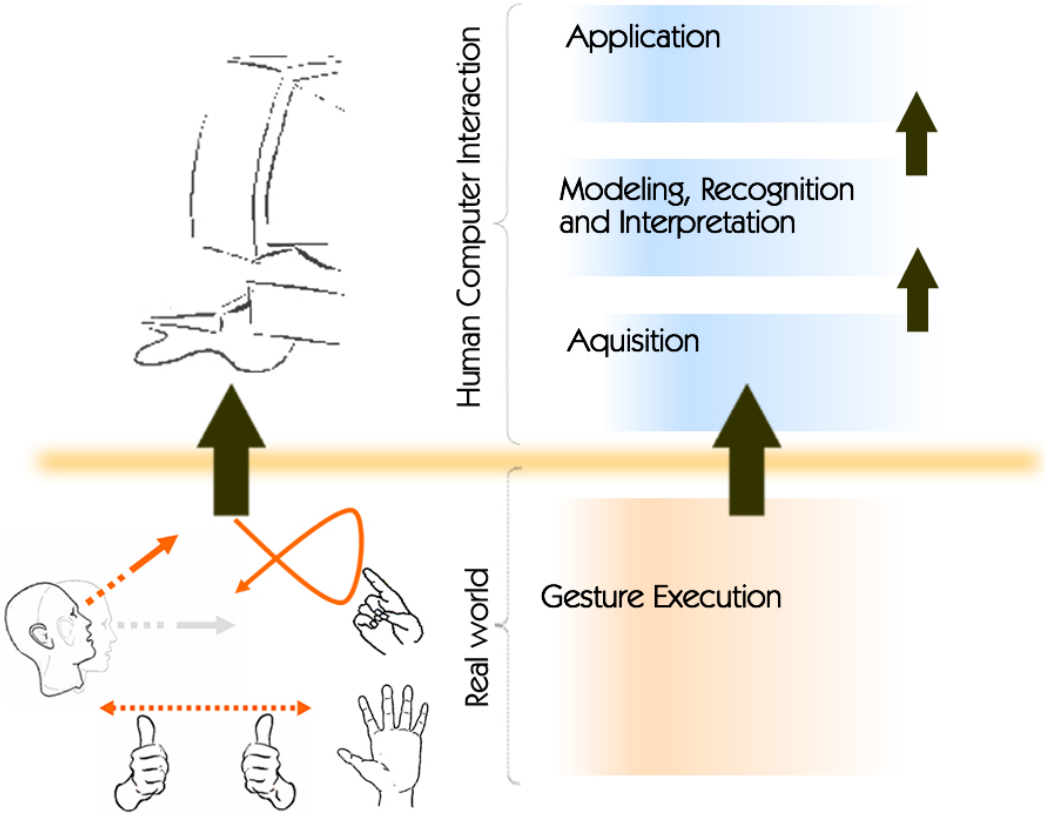


Figure 3.9: Different levels of representation when dealing with gestures for Human Computer Interaction.

We address the multiple forms of representation that human gesture takes at different levels for human computer interaction ranging from gesture acquisition, mathematical model for analysis, pattern for recognition, record for database up to end-level application event triggers.

3.3.1 Acquisition Level

At the acquisition level, a gesture is represented by a stream of data according to the technology used of the capturing device. At this level we dispose of a raw representation of the gesture that may contain a large amount of noise. For example, a gesture may be represented by a single image that contains the user's hand with a specific posture in front of a working desk that may contain additional information: part of the desk, notebooks, the PC keyboard, the user's watch or the color of its shirt as Figure 3.10 illustrates. We are dealing with the raw representation of gesture as provided by the acquisition device and we can easily note the huge amount of extra information that is not needed at all. Noise is heavily present in this basic gesture representation.

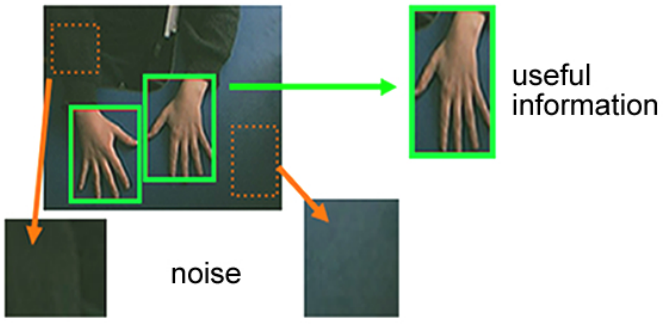


Figure 3.10: Gesture is a raw stream of data at the acquisition level as outputted by the capture device (picture represents a snapshot of lab developed hand gesture acquisition system).

If the interest is on dynamic gestures implying motion trajectories and the

technology is video, the raw representation as outputted by the capturing device would be a sequence of video frames, each containing a huge amount of extra not needed information that we can qualify as noise. Another raw representation of a gesture at this level could be a stream of data points given say as $(x, y, z) \in R^3$ pairs as outputted by a tracking device, for example [Asc07, Sen07]. We would deal in this case of an incomplete representation of our gesture as posture would not been taken into account.

The gesture representation depends on the technology the captor uses. It may be a single image or a sequence of video frames for visual gesture acquisition; a stream of data points for a tracking device; a set of measurements (angles, flexions) for a data glove and many others. This is due to a large amount of non-traditional immersive devices for interacting with virtual environments that have been very rapidly proliferating. They include spatial input devices (or trackers), pointing devices and whole hand devices that allow for hand gestures input. The technology varies including: magnetic, mechanical, acoustic, inertial, vision/video camera based or hybrid [LaV99, VPC05].

3.3.2 Modelling, Recognition and Interpretation Level

At this level we dispose of a mathematical model for gesture, all related to the interaction purpose as stated in the gesture definition paragraph above. We make distinction between posture (as static information) and gesture (in dynamics).

Several stages may be encountered at this level as Figure 3.11 illustrates:

- Gesture modeling: we defined gestures as series of functions of time with values into the Cartesian product of the coordinates space R^d and the set of all postures P.

- Classification / Semantic Interpretation: gesture is also a pattern after feature

extraction has been performed on the mathematical model. Pattern recognition at this level aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. Semantic interpretation may further associate the gesture to a class of semantic types such as commands, gesticulation, etc. Production rules and formal logic or any other artificial intelligence techniques for association to knowledge sets may be used.

- Gesture storage / dictionary of gestures: gesture is also a record in a database that allows for storage of the gesture model (a bijection Ψ may be defined between the gesture model representation and the database storage specific format).

Real problems arise in what concerns choosing the best dictionary for human gestures for a given application. Although gestures are perceived as a natural mean of interacting and conveying information hence a gesture based interface would prove to be ideal, gestures may also be described as imprecise, not self revealing and also non ergonomic. A particular problem relates to finding the right gestures that would feel comfortable and natural from the user's experience point of view. Several attempts have been made on defining gesture dictionaries for application specific needs [Mul86, Soa04, Wex95]. [NMSG03] conducted an ergonomic study for selecting the appropriate gesture commands for operations such as: selection, move, scale, copy, confirm, yes / no, undo using both single hand and two hands gestures in a video camera based top view of a working table.

Another problem arises from the fact that gesture commands have to be identified (or designed) with the particularity of assuring a natural and comfortable user experience, all this considering the existing GUIs and interaction paradigms, for

example WIMP (Windows, Icon, Menu, Pointing). In this particular case, which would be the most suitable gesture for activating / closing a menu or for maximizing / minimizing an application window? A simple proposal is given in Figure 3.12 [VPC05] but the question still remains: are the proposed gestures natural or are we looking for a compromise between natural and new to be learned gestures?

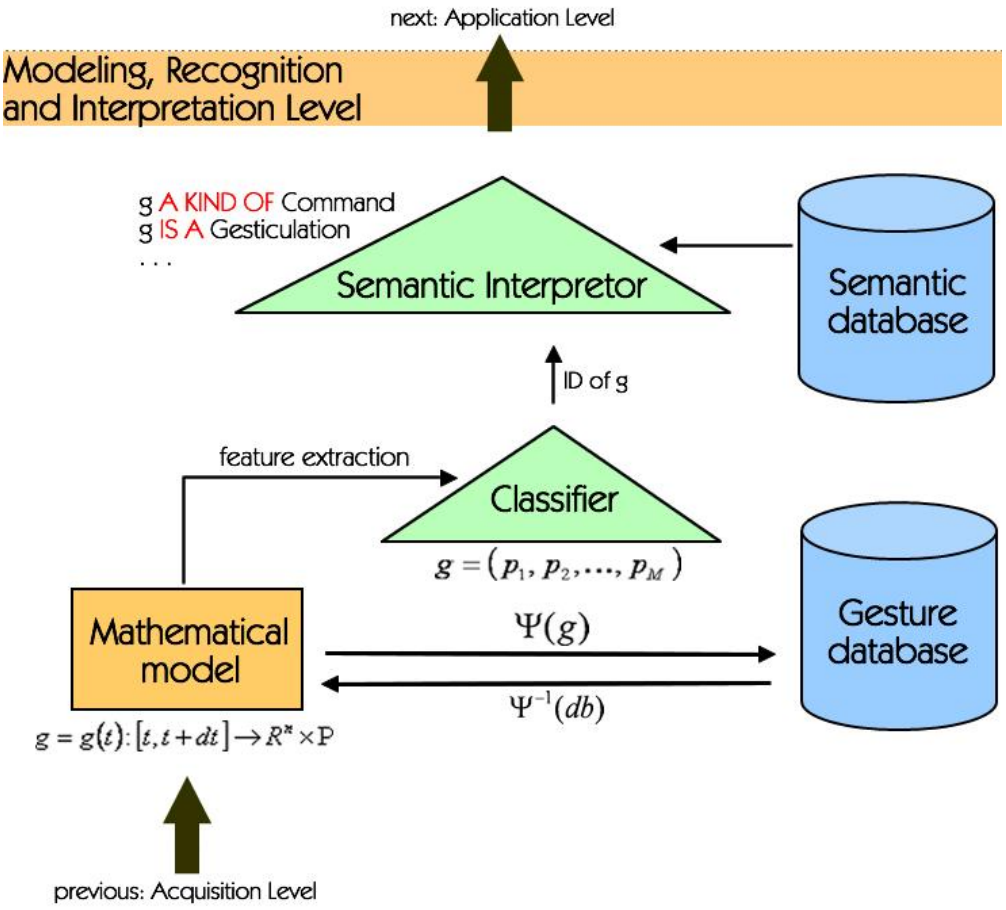


Figure 3.11: Gesture is a mathematical model, a pattern or a database record at the Modeling, Recognition and Interpretation level.

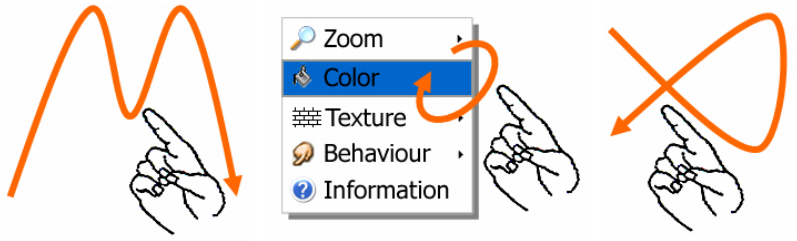


Figure 3.12: Sample dynamic gestures for Open Menu / Select Item / Undo commands.

3.3.3 Application Level

At the application level, the gesture is associated with the trigger of a certain action, according to the current selected working scenario. We can look at the gesture as an event that is fired whenever the gesture interface is enabled and allows for executing the associated action. This can be described using pseudo code language-like formalism as follows:

```
Application app = new Application();
app.EventHandler += new EventHandler(OnGestureInputEvent);
...
result OnGestureInputEvent(EventParams e)
{
    switch(e.GestureType)
    {
        // take appropriate action
    }
}
```

Chapter 4

Detecting the Similarity of Gesture Patterns

Turn away with fear and horror from this lamentable plague of continuous functions that do not have a derivative.
Charles Hermite
(1822 - 1901)

The chapter discuses the problem of motion recognition by introducing a novel representation of gestures based on spline modelling. The use of splines brings in a few advantages with regards to data dimensionality, speed and accuracy of processing. We describe several algorithms for data reduction, spline creation, evaluation and spline alignment.

We further enhance our spline model with elastic properties in analogy with the elasticity theory from basic physics. We look at each spline gesture as a series of connected elastic springs that may be subjected to deformations such as stretching and bending. We may thus associate energy costs to the two deformation types by considering differences in length as representing stretching and differences in curvature for bending. The formulation of energies and the spline alignment algorithm provide for a measure of distance between two gestures represented as spline curves. The similarity measure that we use is influenced by the works of [SKK03, SKK01, BCGJ98]

on curves alignment using standard dynamic programming techniques. We formulate the gesture matching problem using the Nearest Neighbor approach for a set of pre-defined gesture templates.

We equally provide a training algorithm in the context of supervised learning in order to automatically compute gesture templates from examples acquired from user subjects [VGP07]. We discuss for this purpose the average gesture as computed for a set of samples using repeated alignment procedures. The performance of the gesture matching algorithm is evaluated on our own-collected gestures dataset consisting in multiple gestures executed by several subjects. In the end of the chapter we describe a simple demonstrative application for our gesture matching algorithm that allows creation of several 3D objects.

The contribution of this chapter is represented by the novel spline-based representation we introduce to model motion gestures. We show how the use of splines brings in a few advantages with regards to mathematical modelling, data dimensionality, speed and accuracy of processing. Also, our elastic-enhanced splines permit addressing the issue of variability in execution.

4.1 Spline-based Gesture Representation

As defined in a previous chapter, a simple gesture for which only the motion component is important, i.e. a simple dynamic gesture, may be defined as a function of time having values in a given coordinates space R^d:

$$g(t) : R \rightarrow R^d \tag{4.1}$$

where d is the dimensionality of the coordinates space. We restrict our further discussions to only planar gestures for which the dimensionality $d = 2$ however our observations also apply for the $3D$ case as mentioned were appropriately in the chapter.

We are thus looking at a gesture g as a two-dimensional point moving in time:

$$g(t) : R \rightarrow R^2 \tag{4.2}$$

4.1.1 Data reduction

The raw data, as acquired from the tracking device, may be subjected to small acquisition errors, small user execution mistakes or may contain too many points as needed. We are thus preprocessing the acquisition data with a simple filtering algorithm inspired from the Douglas-Peucker polyline reduction algorithm [HS92].

The raw acquisition data comes from the tracking device as a series of r two-dimensional points sampled at equal intervals of time:

$$g_{raw} = \{p_0, p_1, \ldots p_{r-1}\} = \{p(0), p(T), p(2 \cdot T), \ldots p((r-1) \cdot T)\} \tag{4.3}$$

where T is the sampling interval in time and $p_i = p(i \cdot T) = (x_i, y_i) \in R^2$ the ith two-dimensional sampled point.

An initial simple filter that may be applied on the raw data consists in removing of all the points that are too close to their neighbours than a given threshold *eps*.

The principle is illustrated in Figure 4.1. The pseudocode for the filter is given below:

DATA-REDUCTION-1(P, eps)

```
1  for i ← 1 to length[P]
2       do
3          ▷ Remove the point P[i] if it is too close to its neighbour to the left
4          if EUCLIDEAN-DISTANCE(P[i], P[i − 1]) ≤ eps
5             then
6                  P ← P − i
```

The complexity of the algorithm is $O(r)$ where r is the length of the points array.

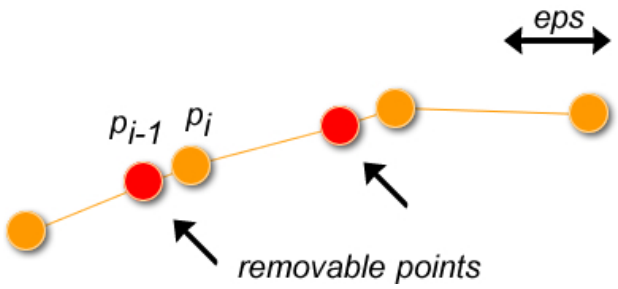


Figure 4.1: Data Reduction, first filter: points that are too close as given by a threshold value *eps* may be removed.

We may further consider a second filter for which we see as superfluous hence removable all the points p_j of the raw data trajectory for which the euclidean distance to the line segment having as extremities the previous and the next points p_{j-1} and p_{j+1} is less than a given error margin *eps*. Even more, we can further enhance the filter by removing all the points p_j that lie between two points p_{start} and p_{end} with $start < j < end$, $start, j, end \in \{0..r\}$ if the average euclidean distance from all the points p_j to the line segment having as extremities p_{start} and p_{end} is less than *eps*. The principle is illustrated in Figure 4.2.

The data reduction algorithm is given in psedo-code below:

DATA-REDUCTION-2(P, eps)

```
1  start ← 0
2  end ← 2
3  while end ≤ length[P]
4        do
5           totalDistance ← 0
6           allPointsUnderLimit ← true
7           for i ← start +1 to end
8                 do
9                    ▷ Update the total distance
10                   distance ← EUCLIDEAN-DISTANCE(P[i], P[start]P[end])
11                   totalDistance ← totalDistance +distance
12                   ▷ Check average distance condition
13                   if totalDistance /(i − start) > eps
14                      then
15                           allPointsUnderLimit ← false
16                           ▷ Exit loop
17                           i ← end
18          if allPointsUnderLimit = true
19             then
20                   end ← end +1
21             else
22                   ▷ Remove points between start + 1 and end − 2
23                   P ← P − {start + 1, start + 2, . . . end − 2}
24                   start ← start +1
25                   end ← start +2
26 ▷ Remove points at the end
27 P ← P − {start + 1, start + 2, . . . end − 2}
```

The complexity of the data reduction algorithm is $O\left(r^2\right)$ where r is the length of the points array. A faster version with several enhancements running in $O\left(r \cdot log(r)\right)$ is described in [HS92].

After data has been reduced by running the two filters, it is sometimes useful to add a number of N linearly interpolated points between any two consecutive points p_i and p_{i+1}. The reason for this is that further modelling of the reduced points using splines leads to a much finer and smoother result. Usually, good results are

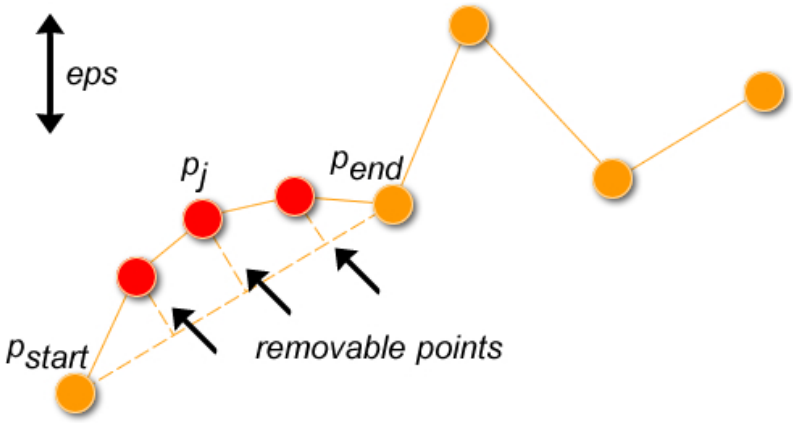


Figure 4.2: Data Reduction, second filter: points that are too close by a threshold value *eps* from the line segment between neighbours may be removed.

obtained for $N = 2$ or $N = 3$ but this step is not mandatory. The linear interpolation pseudocode is given below:

DATA-ENHANCEMENT-1(P, N)

```
1  P' ← nil
2  for i ← 0 to length[P] − 1
3        do
4           for j ← 1 to N
5                 do
6                    ▷ Interpolate the jth point out of N
7                    t ← (j − 1)/N
8                    p' ← (1 − t) · P[i] + t · P[i + 1]
9                    ▷ Add interpolated point to the new array
10                   P' ← P' ∪ p'
11 ▷ Add last point
12 P' ← P' ∪ P [length[P]]
13 return P'
```

The complexity of the algorithm is $O(N \cdot r)$ where r is the length of the points array but it may be considered as $O(r)$ due to the fact that N takes small values such as $1, 2$ or 3. The interpolation may even further be constrained by the actual

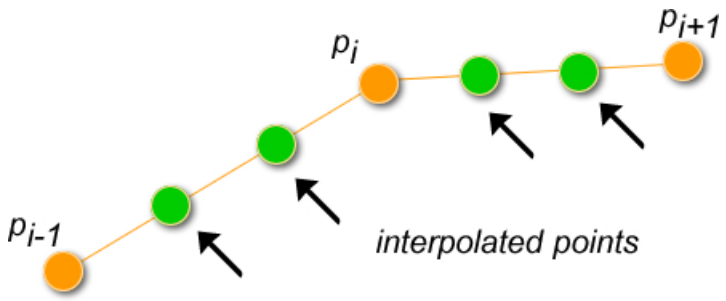


Figure 4.3: Linear interpolation between consecutive points on the trajectory.

distance between two consecutive points p_i and p_{i+1} on the trajectory using same conditions on a threshold value *eps*.

4.1.2 Catmull-Rom splines

Splines represent a powerful mathematical tool for the represention of curves through the means of control points at given timestamps (knots) and blending functions that allow computation of the curve points.

The Catmull-Rom splines are a family of cubic interpolating splines defined such that the tangent t_i at each control point p_i is calculated using the previous and next points on the spline, p_{i-1} and p_{i+1}:

$$\vec{t_i} = r_i \cdot p_{i-1}\vec{p_i} + (1 - r_i) \cdot p_i\vec{p_{i+1}} \tag{4.4}$$

or, equivalently:

$$\vec{t_i} = r_i \cdot (p_i - p_{i-1}) + (1 - r_i) \cdot (p_{i+1} - p_i) \tag{4.5}$$

The spline is completely defined by the control points p_i and their associated tangents $\vec{t_i}$, as Figure 4.4. The ith segment of the spline is defined between the control points p_i and p_{i+1} as:

$$\lambda(u) = \sum_{k=0}^{3} c_k \cdot u^k \tag{4.6}$$

where u is a local parameter that varies between $[0, 1]$. The coefficients $c_k, k = 0, 3$ are computed for each segment using end continuity conditions:

$$\begin{cases} p_i & = \lambda(0) & = c_0 \\ p_{i+1} & = \lambda(1) & = c_0 + c_1 + c_2 + c_3 \\ t_i & = \lambda'(0) & = c_1 \\ t_{i+1} & = \lambda'(1) & = c_1 + 2c_2 + 3c_3 \end{cases} \tag{4.7}$$

The system may be put under a concise matrix form:

$$C = G \cdot P \tag{4.8}$$

where C is the coefficients vector $C = [c_0\ c_1\ c_2\ c_3]^T$, P is the control points vector $P = [p_{i-1}\ p_i\ p_{i+1}\ p_{i+2}]^T$ and G is the geometry matrix as given by:

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -r_i & 2r_i - 1 & 1 - r_i & 0 \\ 2r_i & -4r_i + r_{i+1} - 1 & 2r_i - 2r_{i+1} + 2 & r_{i+1} - 1 \\ -r_i & 2r_i - r_{i+1} + 1 & 2r_{i+1} - r_i - 2 & 1 - r_{i+1} \end{bmatrix} \tag{4.9}$$

Details of computations are given in Appendix A.

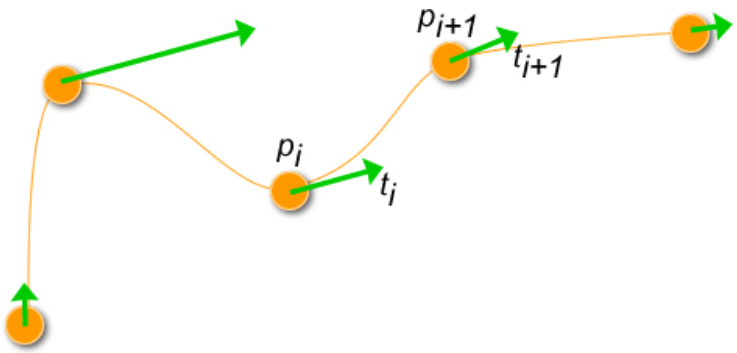


Figure 4.4: A Catmull-Rom spline is defined by the control points and the tangent at each point.

The parameters r_i are known as tensions and affect how sharply the curve bends at the interpolated control point p_i. A common value is 0.5. We have chosen r_i to be determined by the length of the adjiacent segments $p_{i-1}p_i$ and p_ip_{i+1}:

$$r_i = \frac{|p_ip_{i+1}|}{|p_{i-1}p_i| + |p_ip_{i+1}|} \tag{4.10}$$

where $|\cdot|$ denotes the length of a segment. This choice of the tension values allows for each segment $p_i p_{i+1}$ to be pondered inversely proportional to its length when considered for the computation of the tangents. The result that is obtained in the end is a much smoother curve shape. Also, in accordance with the computation details available under the appendix A, the system 4.8 has an unique solution if $r_i \neq 0$ and $r_{i+1} \neq 1$, conditions that are fullfiled due to the fact that $0 < r_i < 1$ as it may be noted from equation 4.10.

The algorithms for computing the tension values, tangents and for generating the Catmull-Rom spline curve from a series of points are given in pseudocode below.

COMPUTE-TENSIONS(P)

```
1  ▷ First tension needs to be aproximated
2  ▷ because the previous point is not available
3  ▷ use default value
4  r[0] ← 0.5
5  for i ← 1 to length[P] − 1
6        do
7           distPrevious ← EUCLIDEAN-DISTANCE(P[i − 1], P[i])
8           distNext ← EUCLIDEAN-DISTANCE(P[i], P[i + 1])
9           r[i] ← distPrevious / (distPrevious + distNext)
10 ▷ Last tension needs to be approximated
11 ▷ because the next point is not available
12 r[length[P]] ← 0.5
13 return r
```

COMPUTE-TANGENTS(P, r)

```
1  ▷ First tangent needs to be approximated
2  ▷ because the previous point is not available
3  tangents[0] ← r[0] · (P[1] − P[0])
4  for i ← 1 to length[P] − 1
5        do
6            tangents[i] ← r[i] · (P[i] − P[i − 1]) + (1 − r[i]) · (P[i + 1] − P[i])
7  ▷ Last tangent needs to be approximated
8  ▷ because the next point is not available
9  tangents[length[P]] ← r[length[P]] · (P[length[P] − P[length[P] − 1]
10 return tangents
```

GENERATE-CATMULL-ROM-SPLINE(P)

```
1  r ← Compute-Tensions(P)
2  tangents ← Compute-Tangents(P, r)
3  T ← 0
4  for i ← 0 to length[P] − 1
5        do
6            ▷ Solve the 4-variable unique-solution system 4.8
7            c[i][0..3] ← Solve-System(P[i], tangents[i], P[i + 1], tangents[i + 1]
8            ▷ Each segment is defined over [0..1]
9            T ← T +1
10 return c, idT
```

Together with the coefficients $c[i][0..3]$ for each of the spline segment, the last procedure also returns the total time-length of the spline, T. Creating a Catmull-Rom spline from a series of r points P is achieved with a complexity of $O(r)$ as the r points are only traversed thrice: for the computation of tensions $O(r)$, tangents $O(r)$ and finally for generating the spline coefficients $O(r)$.

Computing the value of the two-dimensional point on the spline at a given moment T is done in $O(1)$ as presented in the pseudocode below although even faster evaluation methods exist [BG88]:

EVALUATE-SPLINE(c, t, T)

```
1  timeLength ← Σ t[i]
2  if T < 0 or T > timeLength
3     then return nil
4     else
5          ▷ Compute the segment index j
6          j ← ⌊T⌋
7          ▷ ... and the local parameter u of the segment j
8          u ← T − ⌊T⌋
9          return c[j][0] + c[j][1] · u + c[j][2] · u^2 + c[j][3] · u^3
```

The Catmull-Rom splines have a few interesting properties [CR74]:

- They passes through all of the control points p_i that are located at a time interval of 1.0 one from another.
- They have local control which means that modifying one control point only affects the part of the curve near that control point. This can be seen from the system 4.8 where the coefficients of the cubic spline segment i only depend on the points p_{i-1}, p_i, p_{i+1} and p_{i+2}.
- Catmull-Rom splines are continuous functions of class C^1, i.e. there are no discontinuities in the tangent direction or magnitude. They are not C^2 continuous however: the second derivative is linearly interpolated within each segment, causing the curvature to vary linearly over the length of the segment.
- They present affine invariance which means that an affine change in the coordinates system does not affect the geometry of the curve. Practically, the curve shape rests the same whether it is rotated, scaled or translated.
- Catmull-Rom are cubic functions which makes them easy to compute.

4.1.3 Gestures as splines

By using the spline modelling formalism for the gesture representations, the definition of the simple dynamic gesture takes the following particular form:

$$g(t) : [0, T] \rightarrow R^2, g(t) = \begin{cases} \lambda_0(t) & t \in [0, 1] \\ \lambda_1(t-1) & t \in [1, 2] \\ \vdots & \\ \lambda_i(t-i) & t \in [i, i+1] \\ \vdots & \\ \lambda_{n-1}(t-(n-1)) & t \in [n-1, n] \end{cases} \tag{4.11}$$

where T is the time amplitude or time-length of the spline and restricts the definition interval and n is the number of control points. Due to the fact that we used the interval [0..1] as the range of the local parameter for all the spline segments, T and n are the same in this case: $T = n$. λ_i are cubic polynomials that define the shape of each segment:

$$\lambda_i(u) : [0..1] \rightarrow R^2, \lambda_i(u) = \sum_{k=0}^{3} c_k^i \cdot u^k \tag{4.12}$$

Figure 4.5 illustrates the raw acquisition data and the associated spline representation for a few gesture shapes: rectangle, triangle and heart. The raw acquisition points as well as the control points of the splines are highlighted. Also, the raw data and the spline representation are displayed once more overimposed for the purpose of visual comparison.

Representing gesture motions as splines brings in several advantages:

- First of all, the dimensionality of data is considerably reduced. For example, the triangle gesture as illustrated in Figure 4.5 may be represented using just 10 points instead ≈ 200. of the raw data; similarly, for the rectangle only 13 points are needed while the more complex shape of the heart requires 26 points instead of ≈ 300.

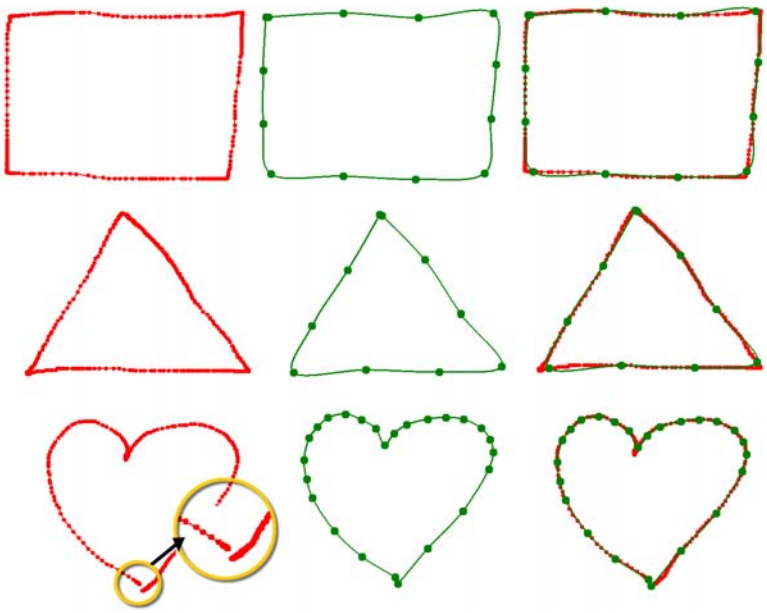


Figure 4.5: Examples of acquisition raw data and spline representation for three gesture types: rectangle, triangle and heart.

- Second, we benefit of a continuous C^1 class function for our gestures which allows the application of results directly from differential geometry. This in turn provides more accurate results for local or global shape parameters (such as tangents for example) instead of values that would have been computed through approximation methods.

- The interpolatory property of the spline may attenuate small execution mistakes or small acquisition errors, by providing in the end smooth curves.

- Spline gestures may be sampled at any given resolution, as fine as desired as Figure 4.6 illustrates. The samplings at different resolutions, coarser or finer, may be needed by further discrete computations. This useful property derives directly from the fact that our representation takes the form of a continuous function and hence we are allowed to select any sampling step $TStep$ in the

spline time-length interval $[0..T]$. Equally important, the sampling may be done uniformly, at equally spaced intervals, or non-uniformly which is a definite improvement from the initial series of raw acquisition points.

- As Catmull-Rom splines are affine invariant, the gesture motion representations present as well translation, rotation and size invariance which is a definitely plus for any classifier algorithm that is comparing or matching shapes.

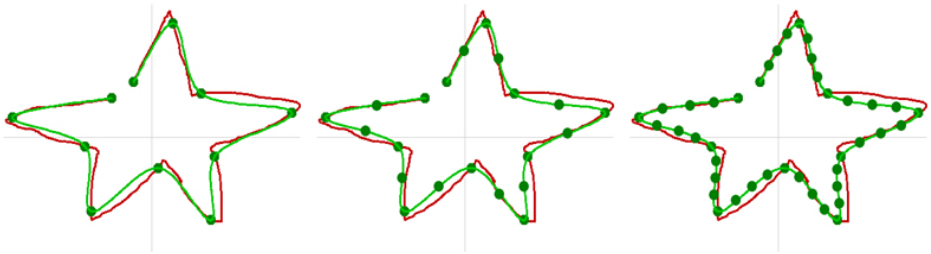


Figure 4.6: Spline gestures may be sampled at any resolution, fine or coarse, as desired for further discrete computations.

4.1.4 Curvature functions

The curvature $k(s)$ of a curve C at a given arc-length value s represents the rate of change of the tangential angle with respect to arc-length:

$$k(s) = \frac{d\phi}{ds} \tag{4.13}$$

Figure 4.7 illustrates the concept visually. The same definition, expressed in time coordinates for $C(t) = (x(t), y(t))$ becomes:

$$k(t) = \frac{d\phi/dt}{ds/dt} = \frac{d\phi/dt}{\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}} \tag{4.14}$$

The arc-length $s(t)$ represents the length of the curve up to t:

$$s(t) = \int_0^t \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2}\, du \tag{4.15}$$

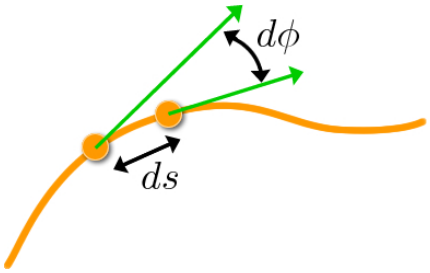


Figure 4.7: The curvature represents the rate of change of the tangential angle with respect to arc-length.

By taking all s from 0 up to the total length of the curve, L, we obtain the curvature function of the curve C:

$$k(s) : [0, L] \rightarrow R, k(s) = \frac{d\phi}{ds} \tag{4.16}$$

The fundamental theorem of differential geometry of curves [Car76] states that the curvature signature function $\kappa(s)$ of a planar curve $C(s)$ parameterized by arc-length s fully prescribes it up to a rigid motion transformation. Hence the curvature functions describe the shape of gestures completely and may be used for shape matching. Several existing shape analysis approaches work only with the curvature information: some make use of curvature scale space representations [AMK99, MA02], detect high curvature points [DF90], propose similarity measures based on curvature differences [FRF04] or introduce different curvature based representations [GGGZ05, MM92].

Moreover, plotting the curvature functions for multiple instances of the same gesture visually confirms a clear discernible high inter-class variance and small intra-class variance. Figure 4.8 illustrates plotting the curvature functions of several gesture trajectories. The figure also gives an idea of the variability that exists within gestures while they are executed. The three types of gestures illustrated, rectangle, star and triangle shaped-like, were performed by the same user at different moments

in time. The intra-class variance is given by variations in length and bending while executing gestures.

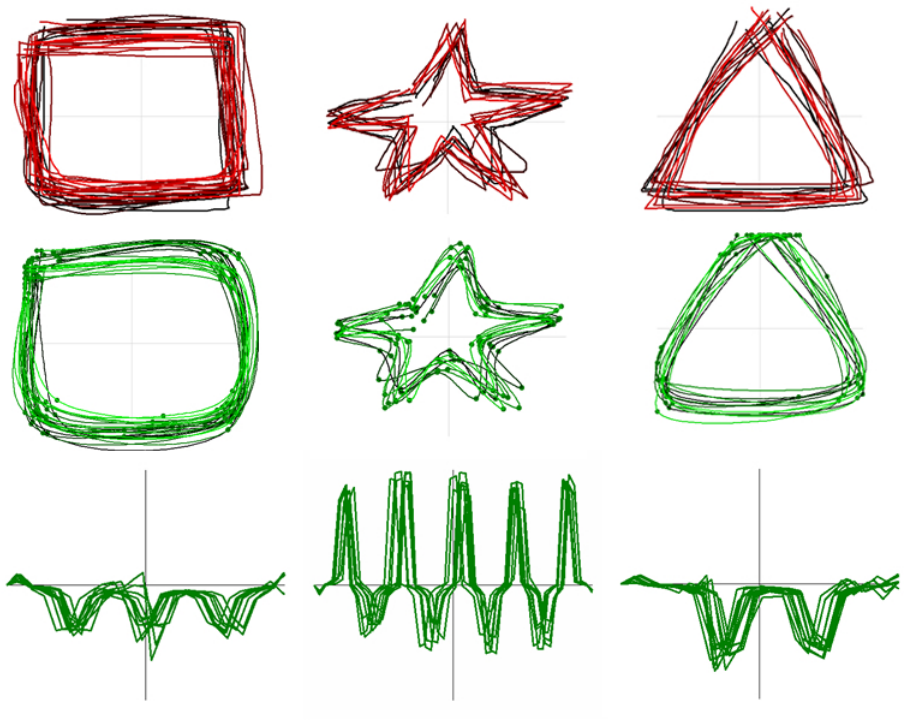


Figure 4.8: Different representations for gestures acquired from the same user at different moments in time. Top row: acquired trajectories. Middle: spline representations. Bottom: curvature functions.

4.1.5 Elastic gestures

We further enhance our gesture spline representations with elastic properties in direct analogy with the elasticity theory from physics. The elastic view of splines will prove useful in the subsequent chapters where we discuss classification and matching algorithms.

We will be looking at a spline curve as a chain of connected elastic springs with infinitesimal lengths. Figure 4.9 illustrates this idea for a star shape-like gesture.

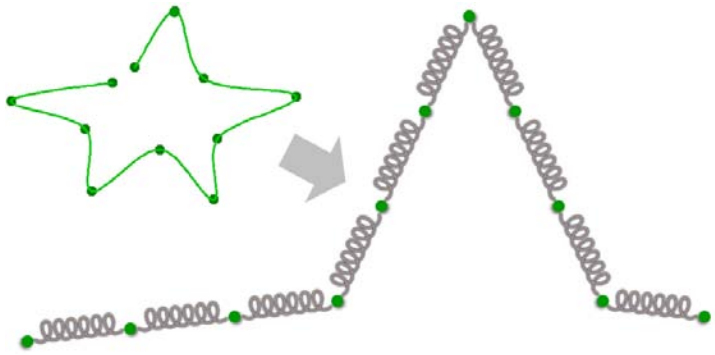


Figure 4.9: Elastic model for gesture: a gesture trajectory is seen as a chain of connected elastic springs.

Each string may be subjected to stretching and bending. The stretching energy required for deforming one spring of length ds to a new length $\overline{ds}$ may be given in analogy with Hookes' law from the elasticity theory as:

$$E_{stretching} = \frac{1}{2}\alpha\left(ds - \overline{ds}\right)^2 \tag{4.17}$$

where α is the stiffness coefficient of the spring. Similarly, instead of considering springs of length we may consider springs that are associated with angles. The energy needed to bend a spring of angle $d\phi$ to a new angle $\overline{d\phi}$ may be expressed in terms of curvatures as:

$$E_{bending} = \frac{1}{2}\alpha\left(d\phi - \overline{d\phi}\right)^2 = \frac{1}{2}\alpha\left(dk \cdot ds - \overline{dk} \cdot \overline{ds}\right)^2 \tag{4.18}$$

where the curvature is defined as the rate of change of the tangent angle with respect to the arc-length $dk = \frac{d\phi}{ds}$. Figure 4.10 illustrates the two types of deformation.

The amount of stretching is measured by the difference in length while bending accounts for the difference in curvature. The larger these differences are, the larger are the energy terms associated with them and hence the cost of deforming the spring becomes bigger. The total energy of deformation required to stretch and

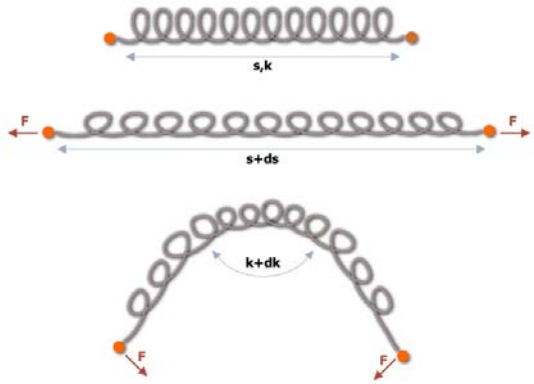


Figure 4.10: String deformations: stretching as difference in length and bending as difference in curvature.

bend a spring of length ds and curvature dk to a new length $\overline{ds}$ and curvature $\overline{dk}$ is given by:

$$\begin{aligned} E_{deformation} &= E_{stretching} + R \cdot E_{bending} \\ &= \tfrac{1}{2}\alpha \left(ds - \overline{ds}\right)^2 + R \cdot \tfrac{1}{2}\alpha \left(dk \cdot ds - \overline{dk} \cdot \overline{ds}\right)^2 \end{aligned} \tag{4.19}$$

where R is a coefficient that controls the distribution of the energy terms.

4.2 Gesture Recognition

Prior to any analysis or matching between two gestures represented as spline curves, an alignment process is necessary to be applied between the two curves. This will make sure that corresponding parts on the two curves will be compared together, for example a corner on the first curve will have a similar corresponding corner on the second one or a semi-circle shaped like portion of the first curve should be matched to a corresponding one on the second. All this of course if the two curves represent the same gesture type, a star for example. Otherwise, the alignment procedure will have as result an optimal association of similar regions of the two curves showing

where they look alike and where they differ.

4.2.1 Energy-efficient Alignment of Curves

Deformation based approaches for aligning curves consider the transformation of one curve into another by minimizing a performance functional of energies in accordance with the elasticity theory [SKK03, CAS92, SKK01, BCGJ98, ACY96, VGP07]. Ideal alignments between curves will allow for similar parts to be compared together, leaving out errors that may be caused by articulations, deformation of parts or other variations in the curves' shapes. The main idea is that similar curves should need small energy variations for transforming one into the other while different curves would require bigger deformation costs.

Let $C(s)$ and $\overline{C}(\overline{s})$ be two curves indexed by arc-length where $s \in [0, L], \overline{s} \in [0, \overline{L}]$ and $L, \overline{L}$ are the length of the curves. Let a mapping

$$g : [0, L] \rightarrow [0, \overline{L}], g(s) = \overline{s} \tag{4.20}$$

represent an alignment of the two curves. The principle of elastic deformation may be described as in [CAS92] by comparing the displacement and bending energies of the two curves using the following functional form:

$$\mu[g] = \int_{C} (C(s) - \overline{C}(\overline{s}))^2 + R \cdot (k(s) - \overline{k}(\overline{s}))^2 ds \tag{4.21}$$

where $k(s), \overline{k}(\overline{s})$ are the curvatures along the $C, \overline{C}$ curves and R is the parameter that controls the contribution of each energy term. The optimal match is given as:

$$\mu^* = \min_{g} \{\mu[g]\} \tag{4.22}$$

A premise of the approach is that goodness of the optimal match is the sum of the goodness of small infinitesimal matches.

An equivalent form of the energy functional integral is given by [BCGJ98]. Both these formulation however are asymmetric as they allow the explicit dependence on the alignment function g. [Tag99] proposes a bimorphism to match the curves to be compared by seeking a pair of two functions ϕ_1, ϕ_2 as part of the bimorphism which optimize the energy functional.

The asymmetry issue is also approached by [SKK03, SKK01] by introducing the notion of alignment curve which can be viewed as the pairing of two particles, one on each curve traversing their respective paths monotonically but with finite stops allowed. This alignment can be specified in terms of two functions $h, \overline{h}$ relating the arc-lengths along $C, \overline{C}$ to a newly defined curve parameter $\xi, s = h(\xi), \overline{s} = \overline{h}(\xi)$. The energy functional formulation becomes:

$$\mu[h, \overline{h}] = \int_{\xi} \left| \frac{dh}{d\xi} - \frac{d\overline{h}}{d\xi} \right| + R \cdot \left| k(h(\xi)) \cdot \frac{dh}{d\xi} - \overline{k}(\overline{h}(\xi)) \cdot \frac{d\overline{h}}{d\xi} \right| d\xi \tag{4.23}$$

Furthermore, [SKK03] show that the optimal alignment of the two curves:

$$\mu(h^*, \overline{h}^*) = min_{(h,\overline{h})} \left\{ \mu[h, \overline{h}] \right\} \tag{4.24}$$

respects the properties of a metric function.

In the discrete case, the two curves $C, \overline{C}$ are sampled into n and m points: $C = \{p_i = (x_i, y_i)/i = 1, n\}$ and $\overline{C} = \{q_j = (x_j, y_j/j = 1, m\}$. Let also k_i and k_j be the curvature values computed at each point $i = 1, n$ and $j = 1, m$ respectively. Curvatures may be computed directly from the continuous spline representations or using any approximation method of choice [HK07]. For example, the curvature at point p_i may be approximated as the angle between the two adjacent segments: $k_i = angle \langle p_{i-1}p_i, p_ip_{i+1} \rangle$. Equally, the arc-length values s_i that measure the length of the curve up to point p_i become: $s_i = \sum_{j=2,i} |p_{i-1}p_i|$.

The alignment between the two curves is represented by a sequence of ordered pairs $\alpha_k = (s_{i_k}, \overline{s_{j_k}})$ with $(s_1, \overline{s_1})$ and $(s_n, \overline{s_m})$ being the first and last pairs of the

sequence, $i_k \in \{1, n\}$ and $j_k \in \{1, m\}$. The optimal alignment is found via dynamic programming considering an energy cost propagation scheme. Let $cost_{i,j}$ be the total cost of transforming the first i segments of curve C into the first j segments of curve $\overline{C}$. Also let $e_{i \to j}$ be the energy term required to transform the ith spring segment of curve C into the jth spring of curve $\overline{C}$:

$$\begin{aligned} e_{i \to j} &= e_{stretching,i \to j} + R \cdot e_{bending,i \to j} \\ &= (ds_i - \overline{ds_j})^2 + R \cdot (k_i - \overline{k_j})^2 \end{aligned} \tag{4.25}$$

where $ds_i = s_i - s_{i-1}$ for $i = 2, n$, $s_0 = 0$ and $\overline{ds_j} = \overline{s_j} - \overline{s_{j-1}}$ for $j = 2, n$, $\overline{s_0} = 0$.

We also define the required energy for removing the ith spring as transforming it into a void/nil spring of 0 length and 0 curvature:

$$e_{i \to nil} = ds_i^2 + R \cdot k_i^2 \tag{4.26}$$

The energy cost propagation scheme is given by the following equations and visually illustrated in Figures 4.11 and 4.12.

$$\begin{cases} cost_{1,1} = e_{1 \to 1} \\ cost_{1,j} = cost_{1,j-1} + e_{j \to nil} \\ cost_{i,1} = cost_{i-1,1} + e_{i \to nil} \\ cost_{i,j} = min \begin{cases} cost_{i-1,j-1} + e_{i \to j} \\ cost_{i-1,j} + e_{i \to nil} \\ cost_{i,j-1} + e_{j \to nil} \end{cases} \end{cases} \tag{4.27}$$

The algorithm follows the standard dynamic programming approach [Bel03]: the process of finding the total deformation cost is split in stages with a decision required at each stage. The initialization stages consist in computing $cost_{1,1}$ as the transformation of the first segment of C into the first segment of $\overline{C}$ as well as computing the costs $cost_{1,j}$ and $cost_{i,1}$ corresponding to costs of transforming the first segment of C into the first j segments of $\overline{C}$ and vice-versa. The cost of transforming the first i segments of C into the first j segments of j is found by adding to the already optimum alignment (either $cost_{i-1,j-1}$, $cost_{i-1,j}$ or $cost_{i,j-1}$)

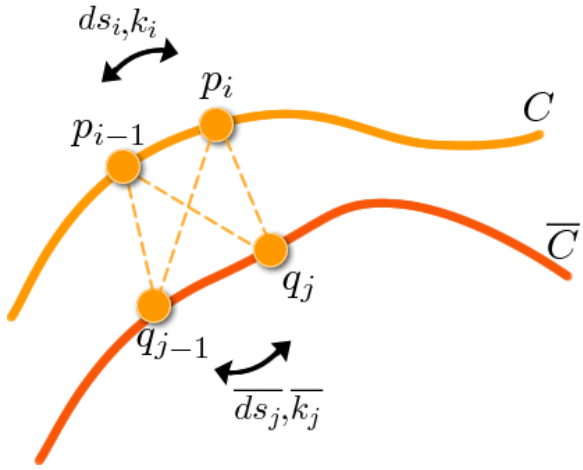


Figure 4.11: Alignment propagation scheme that choses the minimum energy transformation cost path (i, j), $(i-1, j)$ or $(i, j-1)$ (splines view).

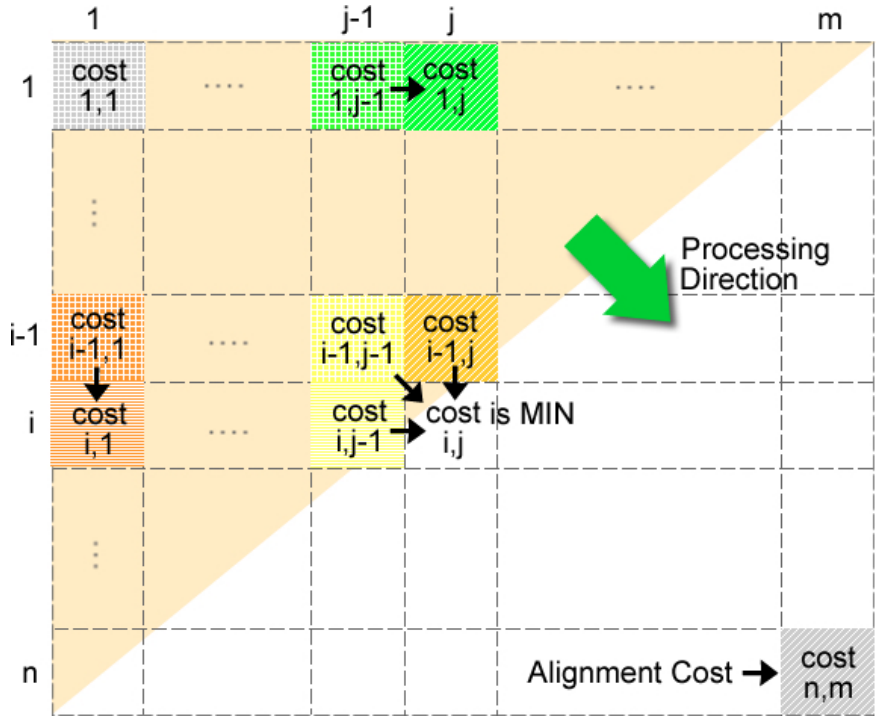


Figure 4.12: Alignment propagation scheme that choses the minimum energy transformation cost path (i, j), $(i-1, j)$ or $(i, j-1)$ (matrix view).

the connection of points that produces the total minimum deformation cost. In the end, the deformation cost required to transform C into $\overline{C}$ is found to be $cost_{n,m}$.

ALIGN-SPLINES$(C, n, \overline{C}, m)$

```
1  ▷ Cost of transforming the first segment of P
2  ▷ into the first segment of Q
3  cost[1,1] ← ENERGY-TO-TRANSFORM(P,1,Q,1)
4  ▷ Compute the energy costs for deforming the first segment of P
5  ▷ into the first j segments of Q
6  for j ← 1 to m
7       do
8          cost[1,j] ← cost[1,j−1] + ENERGY-TO-REMOVE(Q,j)
9  ▷ Compute the energy costs for deforming the first i segments of P
10 ▷ into the first segment of Q
11 for i ← 1 to n
12      do
13         cost[i,1] ← cost[i−1,1] + ENERGY-TO-REMOVE(P,i)
14 ▷ Compute the rest of the deformation costs
15 for i ← 2 to n
16      do
17         for j ← 2 to m
18              do
19                 cost1 ← cost[i−1,j−1] + ENERGY-TO-TRANSFORM(P,i,Q,j)
20                 cost2 ← cost[i−1,j] + ENERGY-TO-REMOVE(P,i)
21                 cost3 ← cost[i,j−1] + ENERGY-TO-REMOVE[i,j−1])
22                 if cost1 ≤ cost2 and cost1 ≤ cost3
23                   then
24                        cost[i,j] ← cost1
25                        α ← α ∪ (i−1,j−1)
26                 elseif cost2 ≤ cost1 and cost2 ≤ cost3
27                   then
28                        cost[i,j] ← cost2
29                        α ← α ∪ (i−1,j)
30                   else
31                        cost[i,j] ← cost3
32                        α ← α ∪ (i,j−1)
33 return α, cost[n,m]
```

The algorithm for aligning the two spline curves C and $\overline{C}$ listed in the pseu-

docode above returns the optimal alignment α as a set of connected points together with the total energy deformation cost, $cost_{n,m}$. The complexity is quadratic with respect to the number of spring segments of the two splines C and $\overline{C}$, $O(n \cdot m)$.

A few examples of the alignment procedure are illustrated in Figure 4.13 for two gesture types, check and question-mark. Also, due to the fact that the curvatures are functions of the arc-length s and in the propagation scheme the curvatures values enter directly, we can further apply the alignment procedure directly on the curvature functions as given in figure 4.8. An illustration of the alignment result between the curvature functions of two star shape-like gestures is given in figure 4.14.

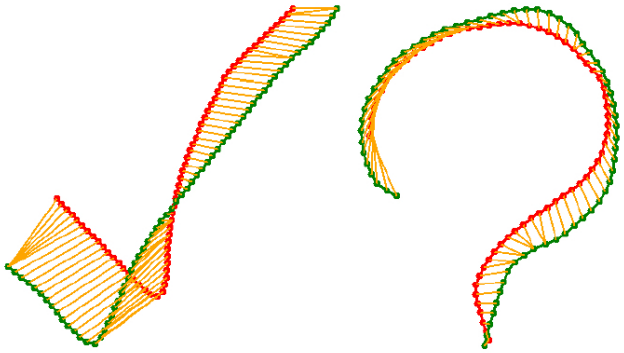


Figure 4.13: Trajectory alignment for the check and question-mark gestures.

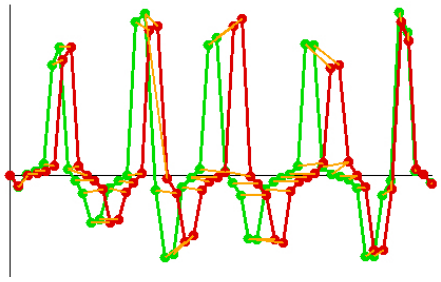


Figure 4.14: Trajectory alignment for the star gesture applied directly on the curvature functions.

4.2.2 Recognition Algorithm

Recognizing a gesture may be performed by computing the alignment cost to several pre-defined gesture templates and classifying the new gesture to the class for which the deformation cost is minimum. The approach follows the standard procedure of the Nearest Neighbor classification [CH67, DH73].

Let G be the spline representation of the new to be classified gesture. Also, let $T = \{T_1, T_2, \ldots T_p\}$ be a set of spline templates corresponding to several gestures. Determining the class index j of G in accordance with the Nearest Neighbor approach may be written as:

$$j = \arg \min_{i=1,p} \{\mu[G, T_i]\} \tag{4.28}$$

The pseudcode for classifying a new gesture to a series of templates is given below.

CLASSIFY-GESTURE(G, T)

```
1  min ← ALIGN-SPLINES(G, |G|, T[1], |T[1]|
2  class ← 1
3  for i ← 2 to length[T]
4        do
5           localMin ← ALIGN-SPLINES(G, |G|, T[i], |T[i]|)
6           if min > localMin
7              then
8                   min ← localMin
9                   class ← i
10 return class
```

An additional improvement may be added to the matching algorithm in the form of a rejection rule. If the minimum alignment cost is greater than a specified threshold then the classification may be rejected with the statement that the new gesture is unknown, not belonging to any of the pre-defined categories.

The complexity of matching a new gesture is $O(p \cdot r \cdot m)$ where p is the number of gesture templates, r represents the number of segments of G and m is the maximum

number of segments for all T_i: $m = \max_{i=1,p} |T[i]|$. This is due to the fact that computation of an alignment between two gestures is performed in quadratic time and searching for the Nearest Neighbor requires computations to each of the template in the set. The value of p is however small due to the limited number of gestures that a dictionary may include, usually in the order of $5 - 15$. The limited choice of p is also related to cognitive load issues and human capacity of memorizing gestures as well as on the specifics of the actual application. Also, the values of r and m are small as discussed in the results section.

4.2.3 Supervised Learning of Gesture Templates

The Nearest Neighbor matching algorithm classifies a new gesture to a set of predefined templates. The only question that still needs to be addressed is how to choose or build a gesture template for a given class of gestures, e.g. stars, rectangles or triangles. In order to do this, we use the concept of average shape of [SKK01] that we apply to our spline representations.

Let $C(g) = \{C^1, C^2, \ldots C^n\}$ be n spline representations of the same gesture g acquired from the one or multiple users. We will call $C(g)$ the training set of the gesture g. Also, let s_i^t and k_i^t be the arc-length and curvature values for the tth sample, $t = 1..n$ and $i = 1..|C^t|$, where $|C^t|$ is the number of sampled points on C^t.

When two gestures C^t and C^u are aligned, each arc-length/curvature pair (s_i^t, k_i^t) of C^t corresponds to a set of pairs $\alpha_i^t = \left\{\left(s_j^u, k_j^u\right)\right\}$ from C^u. Aligning C^t to all the rest of the curves $u = 1, n$, $u \neq t$ will result in the pair (s_i^t, k_i^t) to correspond to the set $\{\alpha_i^u / u = 1, n, u \neq t\}$. The concept is illustrated visually in Figure 4.15. By having computed all these alignment sets, we may consider the curve C^t as being a reference curve and compute the average arc-lengths $\tilde{s_i}$ and curvatures $\tilde{k_i}$ that correspond to each s_i^t and k_i^t pair on the reference curve, $i = 1, |C^t|$:

$$\tilde{s}_i = \frac{\sum_{u=1,u\neq t}^{n} \sum_{\alpha_i^u=\left(s_i^t \rightarrow s_{j_r}^u\right)} s_{j_r}^u}{\sum_{u=1,u\neq t}^{n} \sum_{\alpha_i^u=\left(s_i^t \rightarrow s_{j_r}^u\right)} 1} \tag{4.29}$$

$$\tilde{k}_i = \frac{\sum_{u=1,u\neq t}^{n} \sum_{\alpha_i^u=\left(k_i^t \rightarrow k_{j_r}^u\right)} k_{j_r}^u}{\sum_{u=1,u\neq t}^{n} \sum_{\alpha_i^u=\left(k_i^t \rightarrow k_{j_r}^u\right)} 1} \tag{4.30}$$

where $\left(s_i^t \rightarrow s_{j_r}^u\right)$ means that the arc-length value s_i^t is to $s_{j_r}^u$ and $\left(k_i^t \rightarrow k_{j_r}^u\right)$ denotes the same for curvatures.

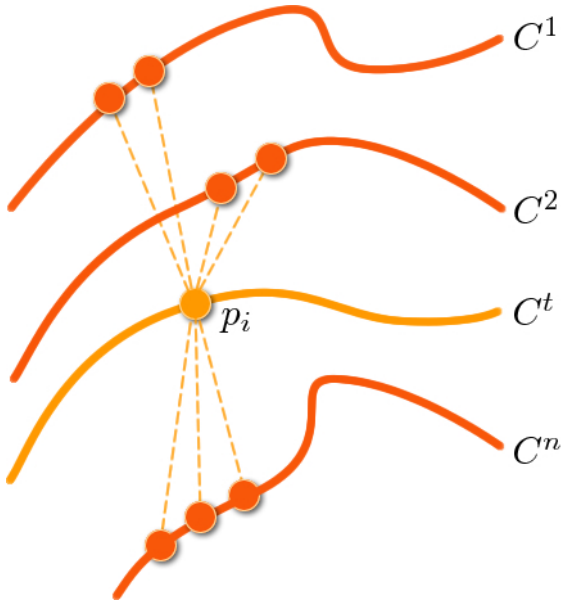


Figure 4.15: A point on a spline may be aligned to multiple consecutive ordered points on the other splines.

The average gesture $\tilde{C}$ will have the same number of sampled points as the reference curve C^t: $|\tilde{C}| = |C^t|$. [SKK01] note that choosing any shape from the set as being the reference has little impact on the average shape result. However, we

choose as being the reference the gesture index t^* that minimizes the sum of elastic alignments to it:

$$t^* = \arg\min_{t=1,n} \left\{ \sum_{i=1,n,i\neq t} \mu[C^t, C^i] \right\} \tag{4.31}$$

The pseudocode for computing the average template for gesture g from a given set of samples $C(g)$ and a reference index curve t is given below.

COMPUTE-GESTURE-TEMPLATE(C, t)

```
1  ▷ Initial starting values
2  for i ← 1 to |C[t]|
3        do
4           C̃[i] ← (0, 0)
5           ñ[i] ← 0
6  ▷ Align C^t to the rest of gestures C^u
7  for u ← 1 to length[C], u ≠ t
8        do
9           α ← ALIGN-SPLINES(C[t], |C[t]|, C[u], |C[u]|)
10          for i ← 1 to length[α]
11                do
12                   ▷ α[i] represents an association between two indexes
13                   ▷ j and r from the align curves C[t] and C[u]
14                   (j, r) ← α[i]
15                   s̃[j] ← s̃[j] + s[u][r]
16                   k̃[j] ← k̃[j] + k[u][r]
17                   ñ[j] ← ñ[j] + 1
18 ▷ Perform averaging
19 for i ← 1 to |C[t]|
20       do
21          C̃[i] ← (s̃[i]/ñ[i], k̃[i]/ñ[i])
22 return C̃
```

Computing the average gesture is performed with a complexity order of $O(n \cdot |C^t| \cdot m$ where n is the number of samples in the training set and m represents the maximum number of sampled points from all the curves in the set: $m = \max_{u=1,n} \{|C^u|\}$.

4.2.4 Results and Discussion

We acquired a data set of 2,000 gesture samples from 10 volunteers. Each volunteer performed each of the 10 gesture types from Figure 4.16 several times. Figure 4.17 lists a few matching results. The classification results are given twice: as actual cost values and using visual cues. The gray-levels cues range from white to black where black means maximum difference from gesture to template and white perfect match.

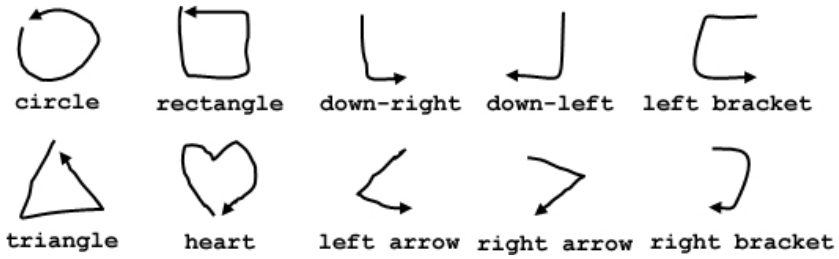


Figure 4.16: Set of 10 gesture types.

For each gesture type, we had a total of 200 samples. Out of these samples, we chose T that made up the training set.We performed the training stage 100 times by randomly choosing the T samples. The rest of $200 - T$ samples were added to the testing set. From each testing set for each of the 10 gesture types, one sample was randomly selected and classified against the just-trained classifiers. We performed these steps 100 times and updated each time the classification error rate (10 x 100 = 1,000 tests were computed for a specified T value). We varied T from 2 to 10 samples which lead us to an error rate value of 6% (or 94% accuracy) when using only 2 training samples to an error rate value of 2.5% (97.5% accuracy) for 10 training samples (or 1 sample from each user).

The main cause of classification errors was related to wrongly classifying down-left gestures as right-arrows and down-right gestures as left-arrows and vice versa. This is due to the fact that our classifiers are rotation invariant hence the only

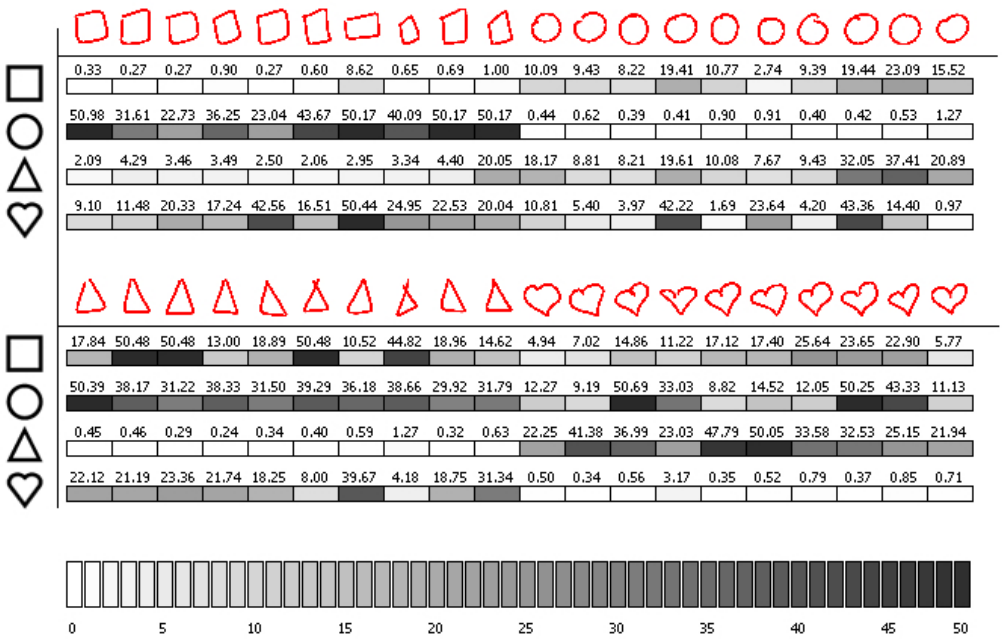


Figure 4.17: Numerical classification results for a few gesture shapes.

difference between the mentioned gesture types only stands in their turn angle (which is 90 degrees for down-left and down-right gestures and smaller for right- and left-arrows). By selecting only 2 stroke samples from the entire data acquired from 10 users (or 0.2 samples per user), the accuracy performance achieved was of 94%. The performance accuracy percent went up to 97.5% with 10 stroke samples (or 1 sample from each user) which makes our method suitable for multi-user gesture recognition.

4.3 Motion-based Command Application

In order to test our gesture recognition technique, we set up a simple demonstrative application that allows creating and translating virtual objects. The following gesture commands (which may be looked upon as a raw and basic gesture dictionary) have been selected, see Figures 4.18 and 4.19:

- Rectangle, triangle and circle shaped-like gestures determine creation of corresponding virtual objects: a cube, cone or sphere.
- Left and right arrows enforce the previous created virtual object to leave the scene to the indicated direction.

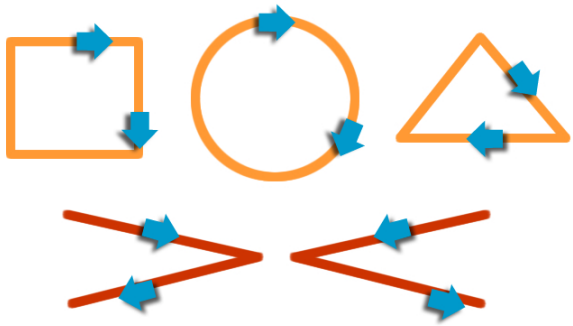


Figure 4.18: Five gesture trajectory commands for a demonstrative application. Top row commands (rectangle, triangle, circle) create virtual objects (cube, cone, sphere). Bottom row commands (go right, go left) allow objects to leave the scene to the indicated direction.

The acquisition scenario is as described in chapter 2 with the video camera mounted on top and monitoring the user's hand above the surface of the working desk. The gesture input corresponds to the motion trajectory captured between the moments when the indexfinger is pointed and when it is retracted back.

There is a permanent visual feedback displayed on the screen that allows the user to see the result of both the hand detection as well as the trajectory capture processes. Tthe detected hand region is visually marked by overimposing a white bounding rectangle as well as a green filled circle representing the center of the hand on top of the video information. Equally, the entire trajectory is displayed on top of the video feed using distinctive colors that code the total time of execution: brightest red for the most recently captured points back to black for the oldest points. This visual feedback helps subjects see how the system is doing in terms of

accuracy and, in case of failure in hand detection or trajectory acquisition, adjust their hand posture accordingly. As described in chapter 2 the acquisition process is fast and robust although there are situations when the index-pointed posture is not correctly detected which leads to failure in the trajectory acquisition. However, the permanent visual feedback makes users aware of this in due time.

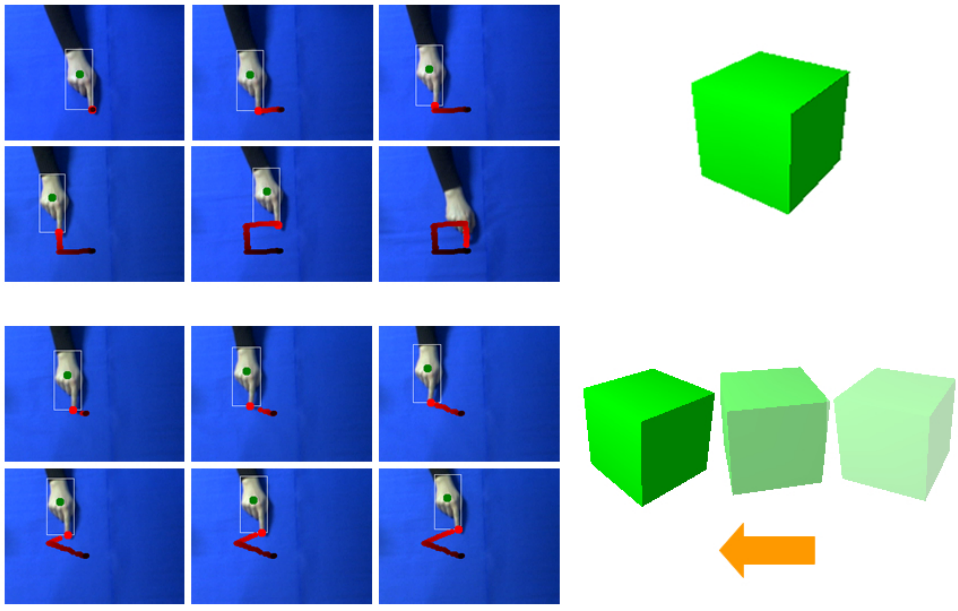


Figure 4.19: Creating a cube object and making it leave the scene to the left.

Video-based acquisition testing was performed on a dual core P4 2.66GHz Windows machine with DirectShow 9 installed. Processing was carried out at a video rate of 25fps with an image resolution of 320x240 pixels. The CPU load varied around $25 - 30\%$ for the entire running mainly due to the video acquisition part $(15 - 20\%)$.

The maximum allowed time frame interval for executing a gesture was of 5 seconds which at a processing speed of 25fps allowed for an accumulation of maximum 125 control points for our spline representations. The resolution step chosen for the

spline segments was 5 giving in the end a maximum dimensionality for the algorithm data of order $n \approx 600$ with an average of $n \approx 250$. The complexity of our classification algorithm for a gesture set composed of $|G|$ templates is $O(|G| \cdot n \cdot m)$ where n and m are the dimensionalities of the spline representations for the two gestures. The complexity is quadratic however, due to our spline representation, the dimensionality of each component is very small of order 250 in the average case.

Chapter 5

Dealing with Variability: From Continuous Motion to Gesture Patterns

Variability is the law of life, and as no two faces
are the same, so no two bodies are alike,
and no two individuals react alike and behave alike.
William Osler
(1849 - 1919)

In this chapter we describe a technique for measuring the amount of variability that is present in gesture execution by introducing a model composed of stretching and bending variation terms. The problem we address, and which has been left unattended so far, is how to provide quantitative criteria for both measuring and comparing the variability in execution across multiple users and multiple gesture types. The model may serve as a basis or provide valuable criteria for researchers involved in gesture recognition and gesture-based interfaces by letting them know in advance how much variability to expect at the user execution level and thus tune the gesture recognizers appropriately. The method is also suited for HCI researchers addressing questions on human factors in gesture execution.

We further use the VE model in order to address a well-known hard problem which is the automatic segmentation of motion trajectories. The problem is hard due to the high complexity needed of detecting patterns of any size, at any rotation and at any point in a given larger trajectory. We present an extension of our elastic matching algorithm for the detection of such gesture patterns in continuous motions. The algorithm is fast as per our discussions on its complexity. Also, our method is invariant to rotation, translation and, very important, to the size of the gesture patterns.

In the end, we validate our VE model by verifying two intuitive hypothesis on how execution is affected by articulation speed or gesture complexity. We discuss experimental results for a database of gestures made very recently (October 2007) specially available for the HCI community that consists in multiple gesture types, multiple users and several speeds of execution.

We can highlight the contributions of the chapter as follows: we introduce a model that measures the amount of variation present in gesture execution and discuss its direct applications to segmentation of continuous motion into gesture patterns as well as performing ergonomic analysis on gesture dictionaries. The model may be used for addressing other open problems in HCI and may provide useful for other research comunities as well, as we discuss in the chapter.

5.1 Variation in Execution

Irrespectively of the methods that are used for acquisition and recognition, there is always a common problem that gesture recognizers have to face and which may be easily observed by looking at the acquisition data: there is a certain amount of variability that comes with each gesture execution. Different users will execute the same gesture differently and, even more, the same user will perform the same gesture with several variations with each execution.

Figure 5.1 illustrates the Variability in Execution problem for single and multiple users. It may be easily observed that the variation is much smaller for the single user case where 10 executions are showed rather than when we selected one execution from 10 users and displayed them over-imposed. The variation is much greater when even more executions are displayed together: 10 users x 10 times = 100 executions.

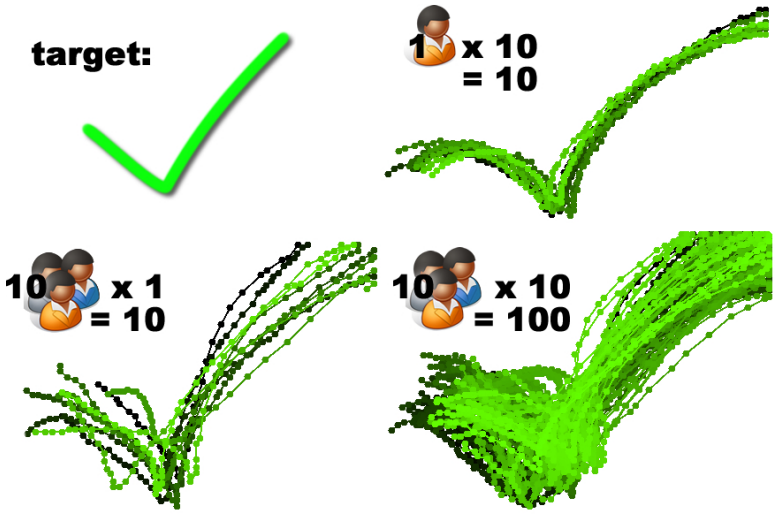


Figure 5.1: Variability in execution. Top left: the *check* gesture. Top right: 10 executions performed by the same user. Bottom left: 10 executions from 10 users (one execution per user). Bottom right: 100 executions from 10 users.

We further address the problem of Variability in Execution (VE) by introducing a model that measures how much users tend to variate their executions at any point of their gesture trajectories. The parameters of the model consist in variations in length and curvature corresponding to local stretching and bending tendencies in execution and are computed directly from a set of gesture examples.

Despite the developments and advances in gesture recognition technique as well as applied systems, few models have been proposed for assessing the human performance for making stroke gestures. Equally, to our knowledge, there is no method or model that would describe the variations that users tend to have when executing gestures. A model for stroke gestures that used the number of approximating straight line segments in a gesture as a predictor of complexity correlating to writing time was introduced in [Iso01]. A quantitative performance model based on Curves, Lines and Corners (CLC) for single-stroke pen gestures was introduced in [CZ07] having error constraints in terms of production time. The relation between figural and kinematic aspects of movement was studied in handwriting and drawing [VT82]. It was found that, throughout the movement, the tangential velocity V is proportional to the radius of curvature r of the trajectory: $V = k \cdot r$. Gesture analysis under the scope of motor control theory is equally performed by [GKP03] that describe a few typical invariants for planar pointing and tracing gestures.

Measuring the amount of variability may act as an important feature for revealing several aspects related to a particular gesture and may affect the decision of including that specific gesture in the gestures set of a given application. Designers of gestures dictionaries may benefit of an objective and quantitative measure of how users will execute gestures and the importance they give to particular details in the structure of each gesture - which may lead to more robust dictionary designs; gesture recognizers may be tuned in advance for how much toleration to allow based on the amount of variability in execution as computed from the examples set; researchers

working on human factors aspects related to gesture execution are given a model for quantitative analysis and comparison of different executions.

5.1.1 A model for Variability in Execution

Our VE model builds on the average gesture for a given set of samples, as defined in the previous chapter.

Let $C(g) = \{C^1, C^2, \dots C^n\}$ be the training set for gesture g, composed of n spline representations acquired from the one or multiple users. Let also $\tilde{C}$ the average gesture computed in accordance with equations 4.29 and 4.30.

Having computed all the alignments α^u from all the $u = 1, n, u \neq t$ samples in the set to the reference gesture g^t, standard deviations for length and curvature may be determined as well. The alignments to one point are visually illustrated in Figure 4.15. The deviations in length $\sigma_{s,i}$ account for tendencies in local stretching while deviations in curvature $\sigma_{k,i}$ model the users' tendency to bend their executions, $i = 1, |C^t|$. Standard deviations are computed at each sample point $\left(\tilde{s_i}, \tilde{k_i}\right)$ of the average gesture $\tilde{C}$ by considering the arc-length and curvature values from all the points of the gestures in the samples set that are aligned to $\left(\tilde{s_i}, \tilde{k_i}\right)$:

$$\sigma_{s,i} = \left(\frac{\sum_{u=1,u\neq t}^{n} \sum_{\alpha_i^u = \left(s_i^t \rightarrow s_{j_r}^u\right)} \left(s_{j_r}^u - \tilde{s_i}\right)^2}{\sum_{u=1,u\neq t}^{n} \sum_{\alpha_i^u = \left(s_i^t \rightarrow s_{j_r}^u\right)} 1} \right)^{\frac{1}{2}} \tag{5.1}$$

$$\sigma_{k,i} = \left(\frac{\sum_{u=1,u\neq t}^{n} \sum_{\alpha_i^u = \left(k_i^t \rightarrow k_{j_r}^u\right)} \left(k_{j_r}^u - \tilde{k_i}\right)^2}{\sum_{u=1,u\neq t}^{n} \sum_{\alpha_i^u = \left(k_i^t \rightarrow k_{j_r}^u\right)} 1} \right)^{\frac{1}{2}} \tag{5.2}$$

The pseudocode for computing the VE model of standard deviations for gesture g from a given set of samples $C(g)$ and the average gesture $\tilde{C}$ is given below.

COMPUTE-VE-MODEL$(C, \tilde{C})$

```
1  ▷ Initial starting values
2  for i ← 1 to |C̃|
3       do
4          σ_s[i] ← 0
5          σ_k[i] ← 0
6          ñ[i] ← 0
7  ▷ Align C̃ to the gestures C^u from the training set
8  for u ← 1 to length[C̃]
9       do
10         α ← ALIGN-SPLINES(C̃, |C̃|, C[u], |C[u]|)
11         for i ← 1 to length[α]
12              do
13                 ▷ α[i] represents an association between two indexes
14                 ▷ j and r from the aligned curves C̃ and C[u]
15                 (j, r) ← α[i]
16                 σ_s[j] ← (s[u][r] − s̃[j])^2
17                 σ_k[j] ← (k[u][r] − k̃[j])^2
18                 ñ[j] ← ñ[j] + 1
19 ▷ Perform averaging and extract square root
20 for i ← 1 to |C̃|
21      do
22         σ_s[i] ← SQUARE-ROOT(σ_s[i]/ñ[i])
23         σ_k[i] ← SQUARE-ROOT(σ_k[i]/ñ[i])
24 return σ_s, σ_k
```

The algorithm returns the standard variations in arc-length $\sigma_s = (\sigma_{s,i})$ and curvature $\sigma_k = (\sigma_{k,i})$ where $i = 1, \tilde{n}$ and $\tilde{n}$ is the number of sampled points on the average gesture $\tilde{C}$. The complexity order is $O(n \cdot \tilde{n} \cdot m)$ where m represents the maximum number of segments for the splines C^u from the training set: $m = max_{u=1,n}|C^u|$.

VE in the form of standard deviations may be illustrated more clearly in the curvature-length (k, s) space where s is the normalized arc-length ranging from 0 to 1 while the curvature k takes values in the $[-\pi, \pi]$ interval. Figure 5.2 visually illustrates our VE model for the *check* gesture in the (k, s) space with standard

deviations being represented as ellipses. The model was built from 10 executions acquired from the same user, as they were presented in Figure 5.1.

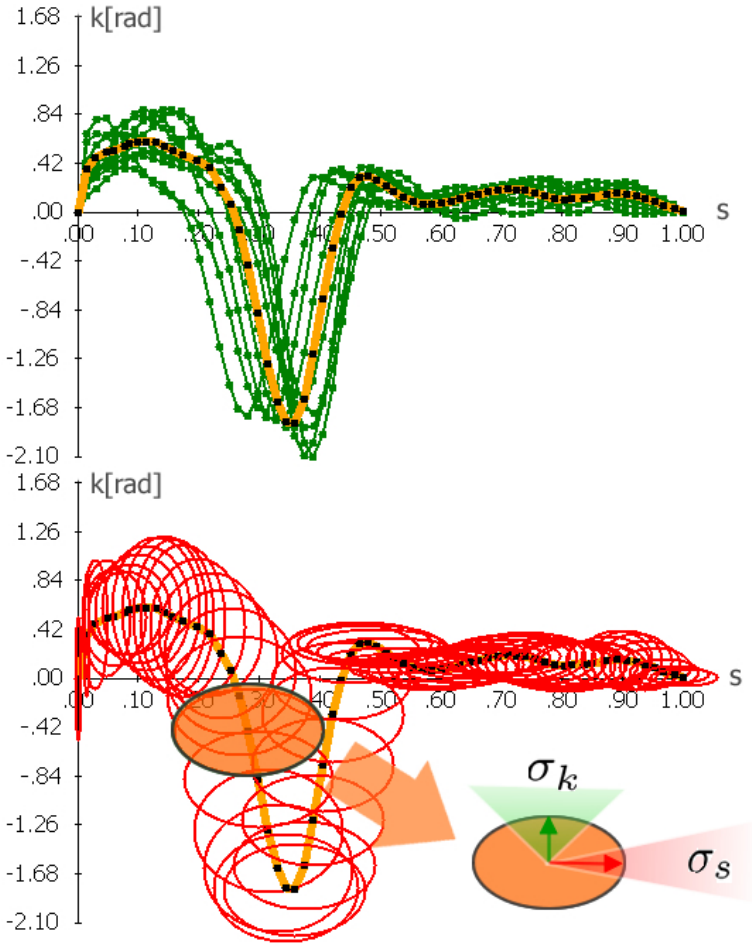


Figure 5.2: VE model in the (k, s) space for the *check* gesture. Top: 10 executions and average gesture over-imposed. Bottom: standard deviations in length σ_s and curvature σ_k measure the users' local tendency to stretch or bend their executions.

Local standard deviations in length $\sigma_{s,i}$ and curvature $\sigma_{k,i}$ may be summed together into a single value which will give a global estimate of the VE that is embedded in the set of execution samples. We introduce the Amount of Variation

in Execution (AVE) that will be further used in our validation experiments as an estimator of VE:

$$AVE = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} (\sigma_{s,i} \cdot \sigma_{k,i}) \tag{5.3}$$

where $\tilde{n}$ is the number of sampling points of the average gesture $\tilde{C}$ of the set for which we are computing VE. As $\sigma_{s,i}$ and $\sigma_{k,i}$ are bounded in the [0..1] and [0..π] intervals, AVE will take values in the range [0..π]. Due to our definition of AVE as product of standard deviations, the actual domain range is shifted toward the [0..10^{-1}] interval as actual computed values showed. As a numerical example, AVE is 0.0015 for the *check* gesture example illustrated in Figure 5.2.

5.2 Automatic Segmentation of Continuous Motion

The concept of Variability in Execution as well as the models of deviations in arc-length and curvature permit addressing the problem of detecting gesture patterns in continuous motion. In other words, a given motion trajectory may be automatically segmented into meaningful components from a set of pre-defined gestures. We make use of the VE model in order to enhance our gesture recognition algorithm and to propose a solution to automatic segmentation.

Many existing gestures-based interaction techniques isolate movements by making use of user-driven discrete events, e.g. using mouse clicks up/down, stylus up/down or users are being required to maintain a button pressed while working with the some tracking device. By taking this approach, it is the users that actually classify their movements and let the system know where the actual gesture command starts and where it ends. Recognition and interpretation is further applied on the already user-segmented gesture trajectory by using shape similarity-based methods that are well established in the pattern recognition community.

Ideal freehand gestural interaction would not require explicit start/stop events

for delimitation of gesture commands: this is clearly irrelevant to the fluidity of the interaction process. Dealing with on-the-fly gesture segmentation is, to our knowledge, still identified as an open problem, e.g. see the conclusions section in [CB05].

The challenge is related to the segmentation of the input trajectory into meaningful command patterns: given a much wider input trajectory, how can we decide under the constraints of real-time processing whether a gesture pattern is present within. The aim is to achieve freehand unrestricted interaction hence the user cannot be required to explicitly input start / stop commands by any means (mouse clicks, holding buttons pressed on a tracking device, etc). The challenge is even more difficult if the relative scale of the pattern to be searched for with respect to the wider input trajectory is not known in advance. Thorne et al. describe a method for segmenting strokes made with a stylus into smaller tokens [TBvdP04] in the context of a very interesting application for animating a virtual character. Their technique is based on corner detection and classification of the inter-corner segments as lines or arc segments with various considerations on direction and orientation. Gestures are defined as consecutive association of several tokens. We discuss a more general approach by making use of a spline-based representation for our gestures and working with curvature information in the continuous space. For the purpose of automatic multi-scale segmentation, we propose a very fast technique by computing the integral of absolute curvature and using it for fast discrimination with regards to the start position or the scale a particular pattern may appear at any given point.

Let $\Gamma(s), C(s)$ be two curves sampled in $s_i, i = 1, n$ and $s_j, j = 1, m$ respectively and we pose the problem of finding the occurrences of $C(s)$ in $\Gamma(s)$ at different scales, see Figure 5.3. The obvious solution is to use the curve comparison method presented above between $C(s)$ and several parts of $\Gamma(s)$. The problem that arises in this case is related to the scale at which $C(s)$ may be found in $\Gamma(s)$. Choosing

different scales of comparison and dividing $\Gamma(s)$ into equal lengths according to the current scale becomes computationally expensive as it gives a complexity of $O(m \times n \times |S|)$ where $|S|$ represents the number of scales to search for.

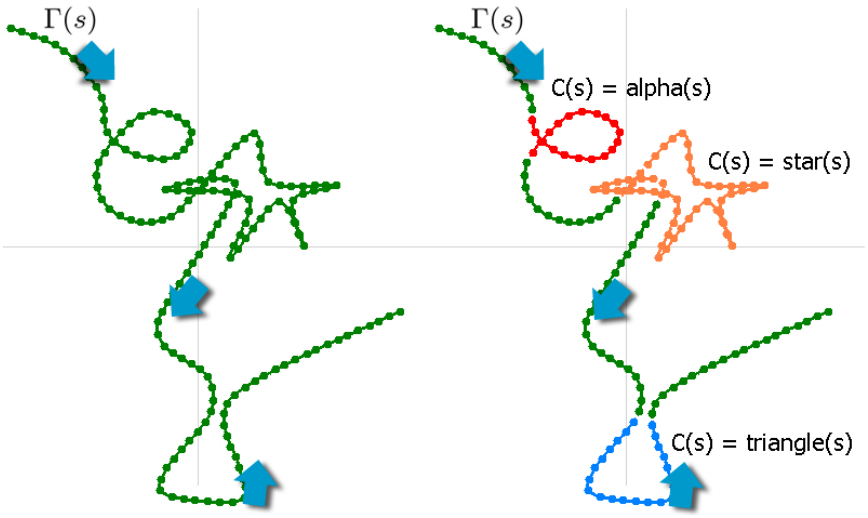


Figure 5.3: Continuous gesture trajectory (left) and identified gesture templates: alpha, star, triangle (right). No assumptions on when templates start or end or on their relative scale with rapport to the entire motion trajectory.

The challenge is to reduce the complexity of sub curve searching by eliminating issues related to the scale dependence, hence to perform a fast search that would consider multiple scales.

In order to achieve this, we use the notion of integral absolute curvature. For a given curve $C(s)$ parameterized by arc-length s with curvature $k(s)$, the integral of absolute curvature is defined as:

$$K(s) = \int_0^s |k(s)|\, ds \tag{5.4}$$

Examples of plotting the cumulative absolute curvature for two gesture types and the correspondences to the curvature function are given in the Figure 5.4.

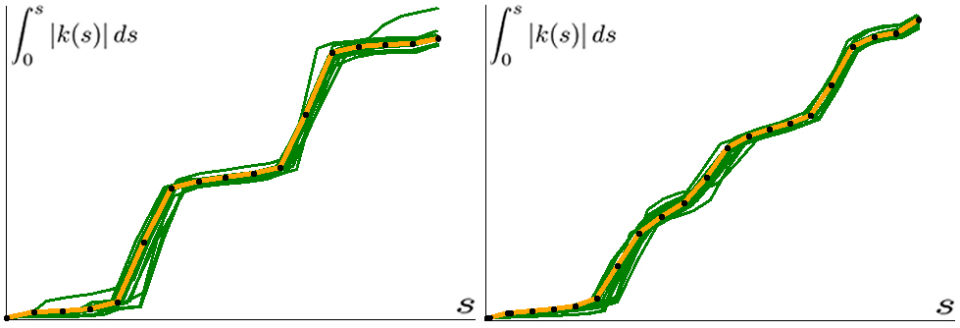


Figure 5.4: Templates for the integral of absolute curvature for two gesture types: rectangle (left) and triangle (right).

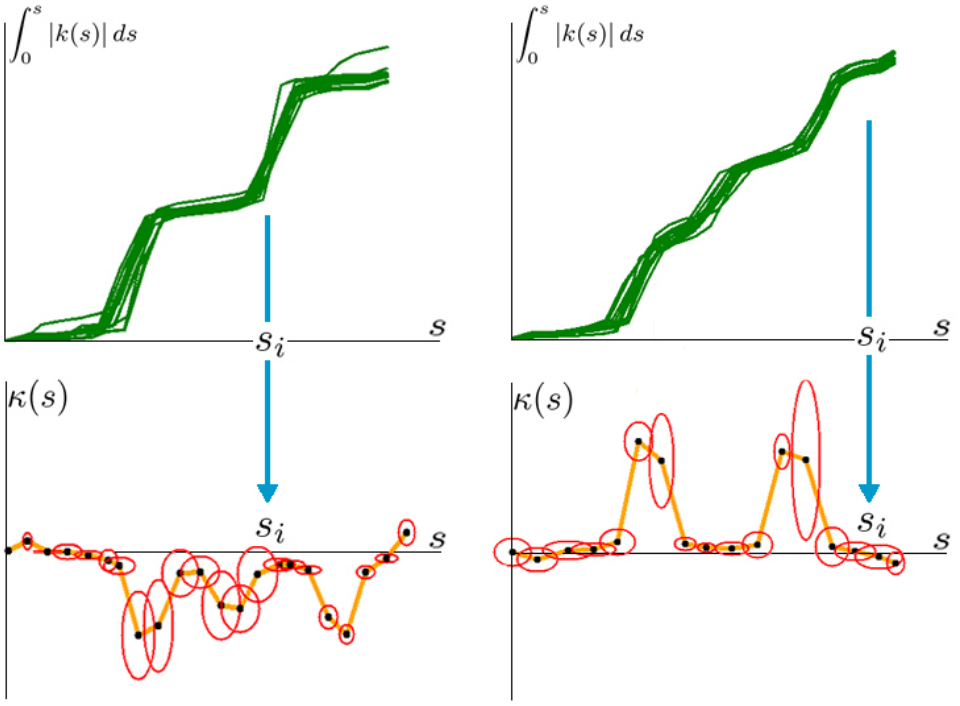


Figure 5.5: Integral of absolute curvature vs curvature for two types of gestures: rectangle (left) and triangle (right) shaped-like. Top row: multiple instances of the integral absolute curvature superimposed. Bottom row: curvature template and standard deviations. Integral of absolute curvature and curvature are related by the arc-length parameter.

The integral absolute curvature is not a curve invariant due to the fact that the curvature $k(s)$ may take negative values as well. Thus, the integral absolute curvature cannot be used by itself for curve matching like the curvature function. However, it has an important propriety: it is monotonically ascending and we can associate values of integral absolute curvature with values of arc-length s (a property that is not met by the curvature function). This enables us to start computing the integral of absolute curvature on the curve $\Gamma(s)$ between two arc-lengths $s_i \leq s_j$ (we start at the beginning of $\Gamma(s)$ with $s_i = s_0 = 0$). The value of the integral absolute curvature between arc-lengths $s_i \leq s_j$ is given by:

$$K(s_i, s_j) = K(s_j) - K(s_i) = \int_{s_i}^{s_j} |k(s)|\, ds \tag{5.5}$$

Comparing $K(s_i, s_j)$ at the current processing step with a template for the integral absolute curvature for the class of curves $C(s)$ belongs to will provide for an estimation of the arc-length s which actually gives the information regarding the relative scale of $C(s)$ in $\Gamma(s)$. This means that we must also pre compute templates for integral absolute curvatures along with our original curvature templates as discussed in the previous section, using the same training algorithm in the context of supervised learning. As the processes are similar, they can be carried out simultaneously during the training stage.

Due to the fact that the integral absolute curvature is not an invariant to be used for curve matching, we use the estimated arc-length s as an entry for the template curvature function, see Figure 5.4. If both curvatures on $C(s_j)$ and $\Gamma(s_j)$ match at s_j with respect to the standard error functions $\sigma^{(s)}$ and $\sigma^{(k)}$ then we may have a partial match starting at s_i on $\Gamma(s)$ up to s_j. We thus may proceed at incrementing s_j until we find a perfect match for the entire length of $C(s)$. The main idea is to increment s_j when a partial matching criterion is met and to advance with s_i otherwise. The algorithm is given in pseudocode below.

DETECT-GESTURE-PATTERN(Γ, C)

```
1  s_i ← 1
2  s_j ← s_i +1
3  totalK ← 0
4  while s_i < s_j
5        do
6           ▷ Update the total absolute curvature
7           totalK ← totalK + |k_Γ [s_j]|
8           ▷ Get the arc-length that corresponds to the total curvature totalK
9           ▷ for the gesture pattern C
10          s_C, index_C ← GET-ARCLENGTH-FROM-TOTAL-CURVATURE(C, totalK)
11          ▷ Does C match up to the point s_j?
12          if |k_Γ[s_j] − k_C[index_C]| ≤ α · σ^k_C[index_C]
13             then
14                   if index_C = length[C]
15                      then
16                           ▷ A match was found
17                           return s_i, s_j
18                           s_j ← s_j +1
19                      else
20                           ▷ C cannot start at s_i
21                           ▷ hence update s_i and retry the matching process
22                           totalK ← totalK − k_Γ [s_i]
23                           s_i ← s_i +1
24                           s_j ← s_i +1
25          return nil
```

The algorithm traverses the curve $\Gamma(s)$ twice (due to the fact that s_j is reseted back to $s_i + 1$ when a match cannot be found in between s_i and s_j) hence we have a complexity of $O(n^2)$ where n is the number of samples on the discrete curve $\Gamma(s)$. Note that the algorithm does not depend on m, the number of sample points of the curve $C(s)$ and it also has the ability of identifying $C(s)$ at different scales. The lowest scale at which $C(s)$ may be matched in $\Gamma(s)$ actually depends on the sampling resolution n chosen for $\Gamma(s)$. Retrieving the arc-length s from the integral curvature for $\Gamma(s)$ may be done in $O(1)$ time by performing pre-computations on

$K(s)$ and this is exactly what the function Get-ArcLength-From-Total-Curvature from the pseudocode implements.

5.3 Ergonomic Aspects of Gesture Execution

We further validate our VE model and verify two intuitive hypotheses on gesture execution for multiple gesture types and multiple subjects:

Hypothesis 1. Articulation speed at which gestures are performed affects the variability in execution. Intuitively, executions performed at low speed should determine users to focus on the details of the gesture and on the accuracy of their executions hence the amount of VE should be small. At the same time, rapid executions will present more variation.

Hypothesis 2. Gesture complexity affects the variability in execution. Simple and shorter gestures are likely to be performed with small variations while complex or time demanding gestures will show greater VE values.

Test gesture database

In order to test our hypotheses we used a database composed of 16 distinct gestures as presented in Figure 5.6. 10 subjects executed each gesture for 10 times at 3 different speeds (slow, medium and fast) giving in the end a total number of 16 x 10 x 10 x 3 = 4800 gesture samples. Gestures were acquired using a stylus pen on a Pocket PC. The full details on the database and the acquisition process are described in [WWL07].

Results and Discussion

We computed AVE values for the 10 subjects, 16 gestures and the 3 different speeds. Figures 5.7, 5.8, 5.9 display the AVE amounts for the slow, medium and fast speeds.

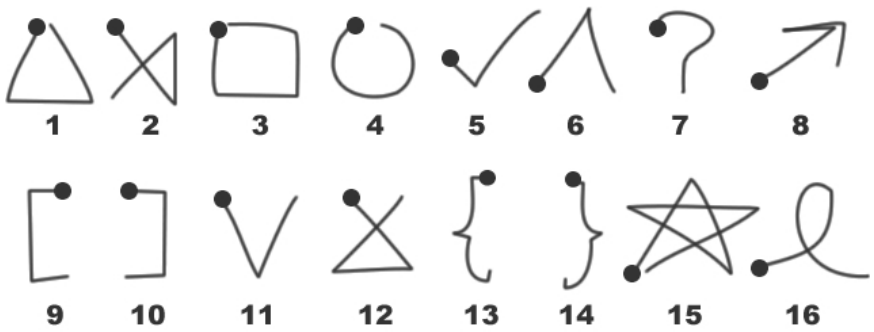
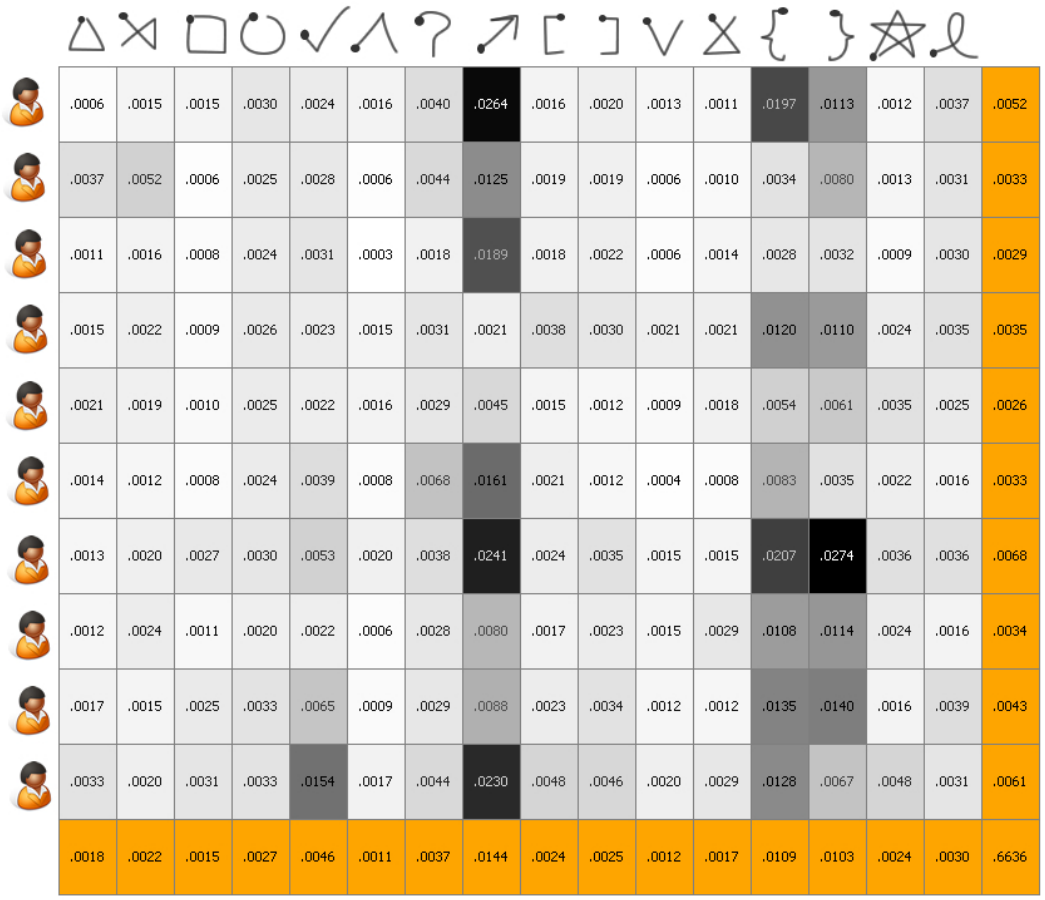


Figure 5.6: Set of 16 gesture types [WWL07]: triangle, x, rectangle, circle, check, caret, question-mark, arrow, left square bracket, right square bracket, v, delete, left curly brace, right curly brace, star, pigtail.

AVEs are presented twice as numerical values as well as using visual cues as rectangles for which the color varies from white to black. A darker colour means a greater AVE value.

Figure 5.10 displays a summary of 48 mean AVE values together with standard deviations for each gesture type and each execution speed. The mean values are computed by averaging AVEs from all the 10 subjects for a given gesture type (10 subjects x 10 executions = 100 samples were considered for each gesture type at a given speed). Figure 5.11 summarizes the data from Figure 5.10 even more by computing mean AVE values factor of speed execution only (1600 samples considered for each speed).

Several observations may be easily drawn from these charts. First of all, *Hypothesis 1* verifies which may be easily observed from the 2nd chart: the mean AVEs are 0.0041 for slow, 0.0043 for medium and 0.0057 for fast speed which translates in a variation increase of 33% from medium to fast and 39% from slow to fast. There is no significant difference between executions at slow and medium speeds (5% only) however the differences from medium to fast and slow to fast are considerably important.



	.0006	.0015	.0015	.0030	.0024	.0016	.0040	.0264	.0016	.0020	.0013	.0011	.0197	.0113	.0012	.0037	.0052
	.0037	.0052	.0006	.0025	.0028	.0006	.0044	.0125	.0019	.0019	.0006	.0010	.0034	.0080	.0013	.0031	.0033
	.0011	.0016	.0008	.0024	.0031	.0003	.0018	.0189	.0018	.0022	.0006	.0014	.0028	.0032	.0009	.0030	.0029
	.0015	.0022	.0009	.0026	.0023	.0015	.0031	.0021	.0038	.0030	.0021	.0021	.0120	.0110	.0024	.0035	.0035
	.0021	.0019	.0010	.0025	.0022	.0016	.0029	.0045	.0015	.0012	.0009	.0018	.0054	.0061	.0035	.0025	.0026
	.0014	.0012	.0008	.0024	.0039	.0008	.0068	.0161	.0021	.0012	.0004	.0008	.0083	.0035	.0022	.0016	.0033
	.0013	.0020	.0027	.0030	.0053	.0020	.0038	.0241	.0024	.0035	.0015	.0015	.0207	.0274	.0036	.0036	.0068
	.0012	.0024	.0011	.0020	.0022	.0006	.0028	.0080	.0017	.0023	.0015	.0029	.0108	.0114	.0024	.0016	.0034
	.0017	.0015	.0025	.0033	.0065	.0009	.0029	.0088	.0023	.0034	.0012	.0012	.0135	.0140	.0016	.0039	.0043
	.0033	.0020	.0031	.0033	.0154	.0017	.0044	.0230	.0048	.0046	.0020	.0029	.0128	.0067	.0048	.0031	.0061
	.0018	.0022	.0015	.0027	.0046	.0011	.0037	.0144	.0024	.0025	.0012	.0017	.0109	.0103	.0024	.0030	.6636

Figure 5.7: Mean AVE values factor of articulation speed of execution.

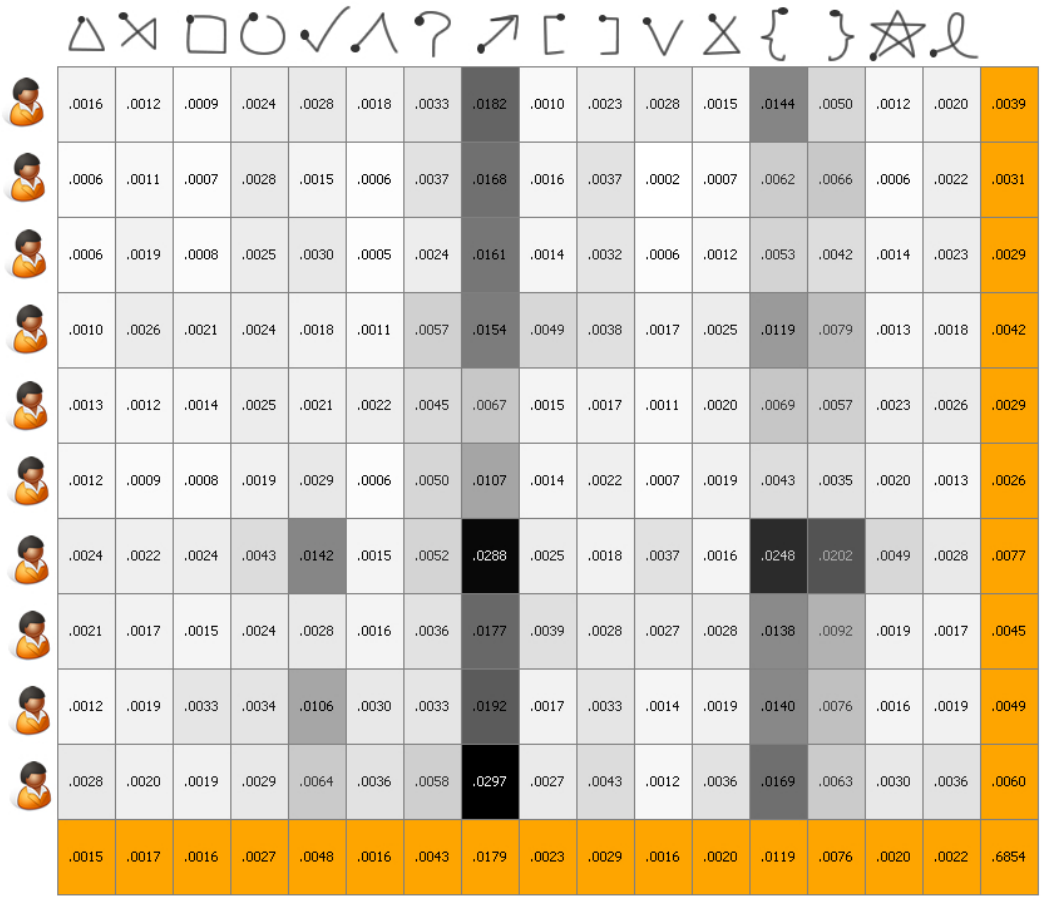


Figure 5.8: Mean AVE values factor of articulation speed of execution.

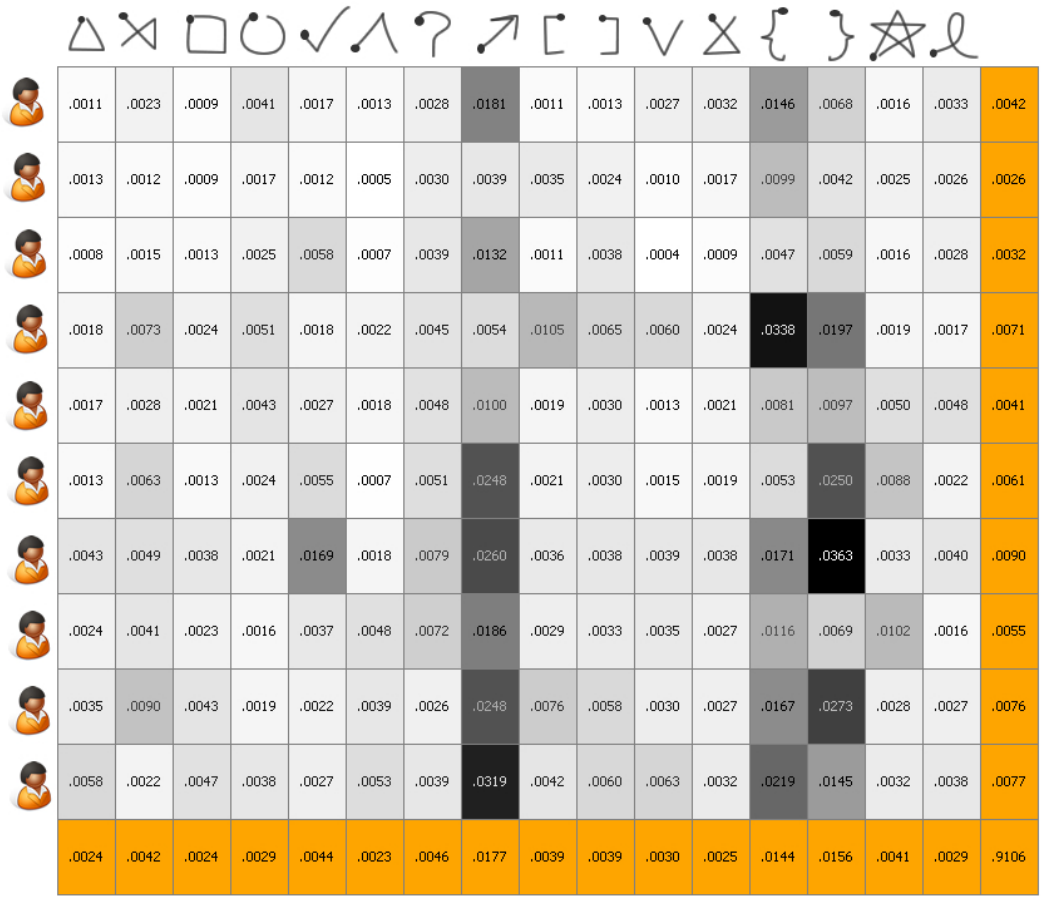


Figure 5.9: Mean AVE values factor of articulation speed of execution.

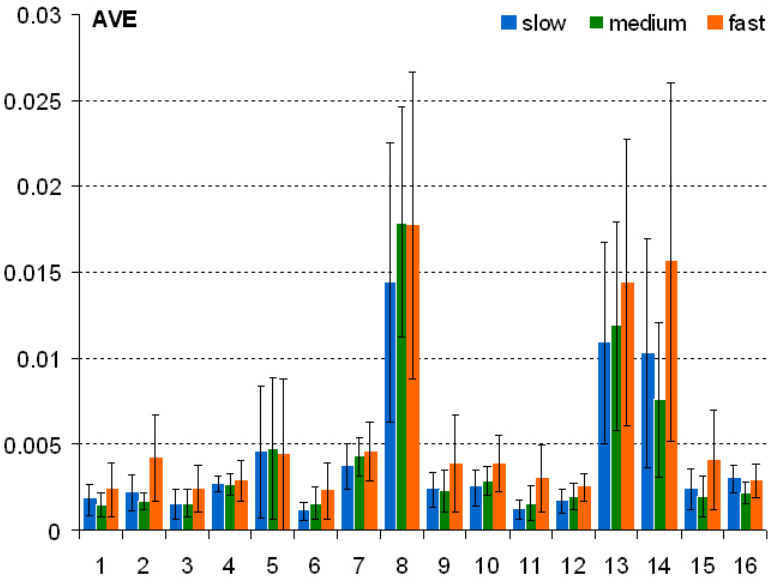


Figure 5.10: Mean AVE values and standard deviations for each of the 16 gestures types (gestures are listed in the same order as in Figure 5.6).

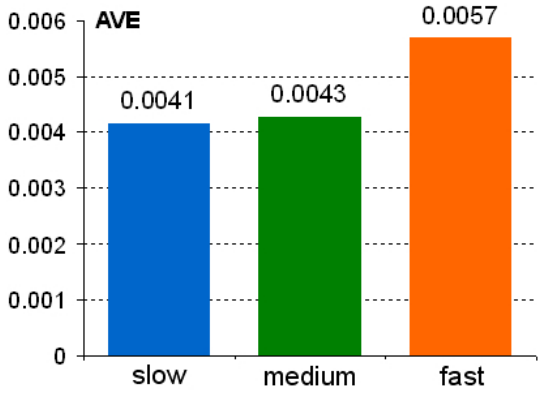


Figure 5.11: Mean AVE values factor of articulation speed of execution.

Hypothesis 2 verifies as well as the 1st chart reveals. Gesture no. 8, the *arrow*, has the biggest AVE value followed by gestures no. 13 and 14, *left curly brace* and *right curly brace*. The *arrow* gesture is the only one from the set that requires a 180 degree shift in execution. The observations are in agreement with qualitative results from [WWL07] where subjects, being questioned on the gestures they executed, commented that they disliked the curly braces. Also, [WWL07] report that the *left curly brace* was the most difficult gesture to recognize using 2 out of 3 different gesture recognition algorithms. On the other hand, simple and familiar gestures such as *triangle*, *rectangle*, *circle* or *caret* present the smallest variation amounts irrespectively of the execution speed. As a numerical example, the difference between *arrow* and *triangle* overpasses 1000%: 0.0179 vs 0.0015 (medium speed).

Another interesting observation relates to gestures that are symmetric such as *left square bracket* and *right square bracket*, *caret* and *v*, *left curly brace* and *right curly brace* for which the AVEs are close at particular speeds. For example, the difference between *left square bracket* and *right square bracket* are of 4% at slow and 0% at fast speed but 26% at medium speed.

Conclusions

The thesis deals with the gesture recognition problem by addressing specific questions while having the main focus centered on providing a flexible model for gestures as well as for the amount of variation present in gesture execution. Variability in execution (VE) has been estimated for single or multiple users and the model of variation was used in order to address the hard problem of continuous motion segmentation and performing ergonomic analysis on gesture dictionaries. The presented models, techniques and results contribute to the concerned research communities by dealing with video acquisition, similarity between gesture patterns, measuring variability in execution, automatic segmenting continuous motions or ergonomic analysis.

We start our conclusions with a few personal observations that may be drawn from the current existing research and state-of-the-art in gesture-based interaction for which the references where given in the first chapter. The contributions presented in this thesis are then listed and commented distinctly in a separate section. The chapter ends with a dedicated section that presents interesting directions of research as opened by the results of this thesis.

Looking at the last several years advances in technological development, we can say that the technology allowing for gesture acquisition is almost there. Many trackers are commercially available and provide high resolution data at high reading frequencies: flock of birds, gloves, finger rings, etc, in general worn equipment. Their only drawback relates to the fact that users have to wear them which has in

the end a negative input on the overall interaction process. Acquiring gestures by means of video cameras comes to alleviate this problem giving the users the feel of interacting naturally (with their bear hands, arms, face or entire body) without being restricted to the use of additional devices. Although computation demands are high for computer vision algorithms, reliable hand tracking results [KT05], fast and innovative implementations of face detectors [VJ01] or migration to unconventional processors [FMA05] have been reported. In addition to this, the last years showed a grow interest in multi-touch processing [Wil04, Han05] and commercial systems such as the Microsoft Surface are available. It can be noted the general tendency of moving gesture interaction toward tabletops scenarios. Bottom line, we can affirm that many acquisition technologies are available today for capturing gestures inside different types of environments although enhancements are still expected from the vision research community.

A second observation is that there is no unified theoretic model of what a gesture represents from the HCI point of view. Different works address specific problems on gesture interaction such as acquisition, recognition or ergonomic aspects and they use simplistic models or they introduce their own that will suit the purpose. As far as we know, there is no thorough mathematical definition of gesture for the purpose of issuing commands in HCI systems nor there are theoretical grounds for defining gesture vocabularies or dictionaries. A sound theoretical model for gestures and gesture dictionaries as the one we start-up in this thesis would prove beneficial for the gesture interaction community.

Many motion recognition algorithms are available reporting good accuracy results. Motion trajectory recognition is a hard problem due to the variability that comes with each execution: users will perform the same gesture differently with each execution. There is once a level of variability for the same user executing gestures and there is a much greater variability when it comes to multi-user gesture recog-

nition. As to our knowledge, there is no objective measure that would take into account the amount of variability that exists within the executions of a given gesture set although this will bring an interesting insight into the usability of a gesture vocabulary for a given application.

Another particular hard problem relates to the automatic segmentation of motion trajectories for identifying meaningful gesture patterns. This problem is usually looked over by adding extra constraints for the users in order to segment themselves the gestures they execute: for example holding a given hand posture in order to signal the start and end timestamps of the gesture execution. Addressing automatic segmentation is difficult and there is no robust solution available that would achieve this in terms of translation, rotation or scale invariance of the gesture patterns with respect to the longer trajectory.

Thesis Contributions

We presented in this thesis the following contributions, each discussed in detail by the various chapters:

1. Models for both gestures and gesture dictionaries were introduced together with thorough mathematical definitions and propositions. Examples and discussions were given for each gesture and dictionary type. We are interested in the problem of modelling the human gesture as command for interaction purposes and we introduced concepts for several gesture types according to the inner structure of posture and motion information they embed.

   Our formalizations permit easy identification of what gesture command types may be used for a given application and what is their mathematical model to be particularized according to the given interaction scenario. To our knowledge, such formalizations of gestures and dictionaries have not yet been proposed

in this complete form and may open interesting directions of development especially on top of our startup of gesture dictionary theory.

2. A model for gesture motions was proposed that makes use of elastic enhanced spline curves. Splines bring a few advantages for gesture motion processing such as: reduction of data dimensionality; more accurate processing by using smoother shapes which attenuate small execution mistakes or small acquisition errors; splines provide a mathematical representation that allows direct application of results from differential geometry, e.g. computing local parameters such as tangents provides more accurate results; affine invariance of representation; sampling at any resolution, fine or coarse.

   The spline model was further used in order to perform curves alignment as well as gesture matching within the Nearest Neighbor approach.

3. A training algorithm was proposed in the context of supervised learning for automatically computing gesture templates from a set of user acquired data samples. The algorithm makes use of the average gesture representation for a given set of samples.

4. A model for the variability encountered in gesture execution was introduced on the top of the average gesture template by considering local variations in length and curvature in the form of standard deviations. The model may serve as a basis or provide valuable criteria for researchers involved in gesture recognition and gesture-based interfaces by letting them know in advance how much variability to expect at the user execution level and thus tune the gesture recognizers appropriately. The method is also suited for HCI researchers addressing questions on human factors in gesture execution.

5. We proposed an extension of the elastic gesture matching algorithm enforced

with our VE model for the automatic segmentation of motion trajectories. This is a known hard problem due to the high complexity needed for detecting patterns of any size, at any rotation and at any point in a given larger trajectory. We presented an extension of our elastic matching algorithm for the detection of such gesture patterns in continuous motions. The algorithm is fast as per our discussions on its complexity. Also, our method is invariant to rotation, translation and, very important, to the size of the gesture patterns.

6. Verification of several hypothesis on ergonomic gesture execution was achieved using our proposed VE model. We showed quantitatively how variation in execution varies in accordance with the articulation speed of execution as well as together with the complexity of gestures. We performed our validation by connecting to the newest existing research and making use of a gesture database very recently made available for the HCI community at ACM UIST in October 2007.

Thesis outreach

Results obtained during these thesis were published in several refereed journals or refereed conference proceedings.

REFEREED JOURNALS (ISI)

1. Radu-Daniel Vatavu, Stefan-Gheorghe Pentiuc, Multi-Level Representation of Gesture as Command for Human Computer Interaction, accepted for Computing and Informatics, ISSN 1335-9150, scheduled for publication in 2008

REFEREED JOURNALS (CNCSIS B/B+)

1. Radu-Daniel Vatavu, Stefan Gheorghe Pentiuc, A Control System with Self Calibration for a Robot Arm based on Video Motion Detection, Buletinul

Institutului Politehnic Iasi: Automatica si Calculatoare, Tomul XLXI (LV), 1 (4), 2005, ISSN 1220-2169, pp. 15-21

2. Radu-Daniel Vatavu, Laurent Grisoni, Samuel Degrande, Christophe Chaillou, Stefan-Gheorghe Pentiuc, Adaptive Skin Color Detection in Unconstrained Environments using 2D Histogram Partitioning, Advances in Electrical and Computer Engineering, Volume 5 (12), Number 1(23), 2005, ISSN 1582-7445, pp. 101-106

3. Radu-Daniel Vatavu, Stefan-Gheorghe Pentiuc, Christophe Chaillou, On Natural Gestures for Interacting in Virtual Environments, Advances in Electrical and Computer Engineering, Volume 5, Number 2(24), 2005, ISSN 1582-7445, pp. 72-79

4. Stefan-Gheorghe Pentiuc, Radu-Daniel Vatavu, Remus Catalin Prodan, Leonard Iurescu, Tudor Ioan Cerlinca, Cristian Ivancescu, Mathematical Models for a Robot Arm Control System, Advances in Electrical and Computer Engineering, Volume 5 (12), Number 1(23), 2005, ISSN 1582-7445, pp. 91-95

5. Radu-Daniel Vatavu, Stefan-Gheorghe Pentiuc, Christophe Chaillou, Laurent Grisoni, Samuel Degrande, Visual Recognition of Hand Postures for Interacting with Virtual Environments, Advances in Electrical and Computer Engineering, Volume 6 (13), Number 2(26), 2006, ISSN 1582-7445, pp. 55-58

DIGITAL LIBRARIES (ACM)

1. Radu-Daniel Vatavu, Stefan Gheorghe Pentiuc, Tudor Cerlinca, Bringing Context into Play: Supporting Game Interaction through Real-Time Context Acquisition, Workshop on Multimodal Interfaces in Semantic Interaction, at ACM ICMI, Nagoya, Japan, 15 November 2007

2. Tudor Cerlinca, Stefan Gheorghe Pentiuc, Marius Cerlinca, Radu-Daniel Vatavu, Hand Posture Recognition for Human-Robot Interaction, Workshop on Multimodal Interfaces in Semantic Interaction, at ACM ICMI, Nagoya, Japan, 15 November 2007

REFEREED PROCEEDINGS

1. Radu-Daniel Vatavu, Stefan Gheorghe Pentiuc, Applied Hands-On Interfaces: Enhancing Object Manipulation inside Virtual Environments, European Conference on the Use of Modern Information and Communication Technologies, ECUMICT 2008, Gent, Blegium, 13-14 March 2008

2. Stefan Gheorghe Pentiuc, Radu-Daniel Vatavu, Ciprian-Ovidiu Ungurean, Tudor Ioan Cerlinca, Techniques for Interacting by Gestures with Information Systems, European Conference on the Use of Modern Information and Communication Technologies, ECUMICT 2008, Gent, Blegium, 13-14 March 2008

3. Radu-Daniel Vatavu, Stefan-Gheorghe Pentiuc, Ovidiu Ungureanu, Gesture Representations for Human Computer Interaction, 9th International Symposium on Automatic Control and Computer Science, Iasi, Romania, November 16 - 18, 2007

4. Radu-Daniel Vatavu, Laurent Grisoni, Stefan-Gheorghe Pentiuc, Gesture Recognition based on Elastic Deformation Energies, the 7th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW) 2007, Lisbon, Portugal, 23 - 25 May 2007, ISBN 978-972-8862-05-3

5. Radu-Daniel Vatavu, Stefan-Gheorghe Pentiuc, Interacting with Gestures: An Intelligent Virtual Environment, The 1st International Conference on Virtual Learning ICVL 2006, 27-29 October 2006, Bucharest, Romania, Proceedings

Technologies and Software Solutions. In Proc: ISBN 978-973-737-218-5, pp. 293-299

6. Stefan-Gheorghe Pentiuc, Radu-Daniel Vatavu, Tudor Cerlinca, Ovidiu Ungureanu, Methods and Algorithms for Gestures Recognition and Understanding, The 8th All-Ukrainian International Conference on Signal/Image Processing and Pattern Recognition (UkrObraz'2006), August 28-31, Kyiv, Ukraine. In Proc: ISBN 966-02-4096-1, pp. 15-18

7. Radu-Daniel Vatavu, Stefan-Gheorghe Pentiuc, Christophe Chaillou, Laurent Grisoni, Samuel Degrande, Visual Recognition of Hand Postures for Interacting with Virtual Environments, The 8th International Conference on Development and Application Systems DAS 2006, 25-27 May, Suceava, Romania, pp. 461-466

8. Radu-Daniel Vatavu, Stefan-Gheorghe Pentiuc, Motion and Color Cues for Hands Detection in Video Based Gesture Recognition, International Conference On Computers, Communications & Control, Oradea, Romania, 1-3 June 2006

Part of the results obtained during this thesis have been included in the project Gestures-based Interaction with Information and Robotics Systems, acronym INTEROB, project that was financed in the frame of the Research of Excellence National Grants (CEEX), Ref. No. 131-CEEX-II03/02.10.2006.

The future directions of research that were opened by the results in this thesis motivated the submission of the proposal Gesture Acquisition and Recognition Framework for Interacting with Computers (GRAFIC) for the call FP7-ICT-2007-1 as part of the FP7 Research Framework (proposal number 215217).

During the thesis period the author was awarded 3 research scolarships by AUF,

Association Universitaire de la Francophonie, for a total number of 23 months as well as a research of exellence Eiffel Egide scolarship by the French Ministry for Foreign Affairs for 10 months (not accepted due to superposition of multiple scolarships).

Future directions of research

Several future directions of research are interesting to be followed, as they are opened by the various ideas introduced in this thesis:

1. It would be interesting to continue on and advance the theoretical developments of the gesture dictionaries that we introduced in chapter 3. Areas of future work may include: providing a general measure for the appropriateness of the dictionary for a given application; defining and implementing operations for adding and removing gestures and how this would globally affect the dictionary appropriateness measure; could one dictionary be reused for multiple applications and at which costs, etc.

2. Our pattern matcher for detecting gesture similarities presents several weaknesses as it is not invariant with respect to the starting point or the direction of execution. For example, drawing a rectangle one direction and the other or choosing different starting points will lead to different curvature functions (that are either of opposite sign or represent translations on the arc-length axis of the trained gesture pattern) hence the recognition algorithm will fail at correctly classifying them. Future work could address recognition of executions that are invariant with respect to the starting point or the direction of execution with the direct benefit of more freedom and less constraints for the users. The same point applies for our proposed extension for the automatic segmentation of continuous motion into gesture patterns.

3. Work can be continued on the analysis of human gestures by using the amount of variation in execution technique we introduced in chapter 5. We showed how variation is influenced by the execution speed or the complexity of gestures but other directions may be interesting investigating such as: is variation in execution influence by fatigue, experience or advanced training? can a variation measure be associated to a given gesture dictionary and what is it exactly over the entire range of users working with a given application? The model may serve as a basis or provide valuable criteria for researchers involved in gesture recognition and gesture-based interfaces by letting them know in advance how much variability to expect at the user execution level and thus tune the gesture recognizers appropriately. The method is also suited for HCI researchers addressing questions on human factors in gesture execution.

Bibliography

[ACY96] R. Azencott, F. Coldefy, and L. Younes. A distance for elastic matching in object recognition. In *Proc. 13th Int. Conf. on Pattern Recognition*, pages 687–691, 1996.

[AMK99] S. Abbasi, F. Mokhtarian, and J. Kittler. Curvature scale space image in shape similarity retrieval. *Springer Journal of MultiMedia Systems*, 7(6):467–476, 1999.

[Ang01] Elli Angelopoulou. Understanding the color of human skin. In *Proc. SPIE Conf. on Human Vision and Electronic Imaging VI*, pages 243–251. SPIE Press, 2001.

[Asc07] AscensionTechnologyCorporation. http://www.ascension-tech.com, last visited: Oct. 2007.

[BB78] B. Butterworth and G. Beattie. Gesture and silence as indicators of planning in speech. In R. Campbell and P. Smith, editors, *Recent advances in psychology of language: Formal and experimental approaches*, pages 347–360. New York: Plenum, 1978.

[BBL93] Thomas Baudel and Michel Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Commun. ACM*, 36(7):28–35, 1993.

[BCD06] B.C. Bedregal, A.C.R. Costa, and G. P. Dimuro. Fuzzy rule-based hand gesture recognition. In *Proc. of IFIP Conference on Artificial Intelligence*, pages 1–10. Springer, Berlin, 2006.

[BCGJ98] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38(15):2365–2385, 1998.

[BE92] Russell Beale and Alistair D. N. Edwards. *Recognising postures and gestures using neural networks*, pages 163–172. Ellis Horwood, Upper Saddle River, NJ, USA, 1992.

[Bel03] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957 (2003).

[BF92] Russell Beale and Janet Finlay, editors. *Neural networks and pattern recognition in human-computer interaction*. Ellis Horwood, Upper Saddle River, NJ, USA, 1992.

[BG88] Phillip J. Barry and Ronald N. Goldman. A recursive evaluation algorithm for a class of catmull-rom splines. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 199–204, New York, NY, USA, 1988. ACM.

[Bis95] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[BMM97] H. Birk, T.B. Moeslund, and C.B. Madsen. Real-time recognition of hand alphabet gestures using principal component analysis. In *Proc. of 10th Scandinavian Conference on Image Analysis, Laeenranta, Finland*, 1997.

[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.

[Cad94] Claude Cadoz. Le geste canal de communication homme/machine. *Technique et Science Informatique*, 13(1):31–61, 1994.

[Car76] M. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[CAS92] I. Cohen, N. Ayache, and P. Sulger. Tracking points on deformable objects using curvature information. In *Proc. European Conf. Computer Vision*, pages 458–466, 1992.

[Cas96] J. Cassell. A framework for gesture generation and interpretation. In R. Cipolla and A. Pentland, editors, *Computer Vision in Human-Machine Interaction*, pages 191–215. Cambridge University Press, New York, 1996.

[CB99] T. L. Chartrand and J. A. Bargh. The chameleon effect: The perception-behavior link and social interaction. *Journal of Personality and Social Psychology*, 76(6):893–910, 1999.

[CB05] X. Cao and R. Balakrishnan. Evaluation of an online adaptive gesture interface with command prediction. In *Proc. of GI 2005 the Graphics Interface Conference*, pages 187–194, 2005.

[CH67] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13(1):21–27, 1967.

[CJH01] Kyung-Min Cho, Jeong-Hun Jang, and Ki-Sang Hong. Adaptive skin color filter. *Pattern Recognition*, 34(5):1067–1073, 2001.

[CKJK05] Jae Sik Chang, Eun Yi Kim, KeeChul Jung, and Hang Joon Kim. Real time hand tracking based on active contour model. *LNCS*, pages 999–1006, 2005.

[COB02] Tiberio S. Caetano, Silvia D. Olabarriaga, and Dante A. C. Barone. Performance evaluation of single and multiple-gaussian models for skin color modelling. In *Proc. 15th Brazilian Symposium on Computer Graphics and Image Processing*, pages 275–282, 2002.

[COB03] Tiberio S. Caetano, Silvia D. Olabarriaga, and Dante A. C. Barone. Do mixture models in chromaticity space improve skin detection? *Pattern Recognition*, 36(12):3019–3021, 2003.

[Com00] Houghton Mifflin Company. *American Heritage Dictionary of the English Language*. Houghton Mifflin Company, 4th edition, 2000.

[CR74] E. Catmull and R. Rom. A class of local interpolating splines. In R. E. Barnhill and R. F. Reisenfeld, editors, *Computer Aided Geometric Design*, pages 317–326. Academic Press, New York, 1974.

[CZ07] Xiang Cao and Shumin Zhai. Modeling human performance of pen stroke gestures. In *Proc. CHI '07*, pages 1495–1504. ACM Press, 2007.

[DF90] R. Deriche and O. Faugeras. 2-d curve matching using high curvature points: application to stereo vision. In *Proc. 10th Int. Conf. on Pattern Recognition*, 1990.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, 1973.

[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.

[Dor94] Brigitte Dorner. Chasing the colour glove: Visual hand tracking. Master's thesis, Simon Fraser University, 1994.

[DT02] Jiangwen Deng and H.T. Tsui. A pca/mda scheme for hand posture recognition. In *Proc. 5th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 279–284, 2002.

[EGG+03] J. Eisenstein, S. Ghandeharizadeh, L. Golubchik, C. Shahabi, D. Yan, and R. Zimmermann. Device independence and extensibility in gesture recognition. In *Proc. of IEEE Virtual Reality*, pages 207–214, 2003.

[Fak07] FakespaceLabs. http://www.fakespacelabs.com, last visited: Oct. 2007.

[FB93] Janet Finlay and Russell Beale. Neural networks and pattern recognition in human-computer interaction. *SIGCHI Bull.*, 25(2):25–35, 1993.

[FH93] S. Fels and G. Hinton. Glove-talk: A neural network interface between a data-glove and a speech synthesizer. *IEEE Transaction on Neural Networks*, 4:2–8, 1993.

[FH95] Sidney Fels and Geoffrey Hinton. Glove-talkii: an adaptive gesture-to-formant interface. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 456–463, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[FM04] James Fung and Steve Mann. Computer vision signal processing on graphics processing units. In *Proc.of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2004.

[FMA05] James Fung, Steve Mann, and Chris Aimone. Openvidia: Parallel gpu computer vision. In *Proc. of the ACM Multimedia*, pages 849–852, November 2005.

[FRF04] John C. Femiani, Anshuman Razdan, and Gerald Farin. Curve shapes: Comparison and alignment. *Transactions on Pattern Analysis and Machine Intelligence*, November 2004.

[FW94] W.T. Freeman and C.D. Weissman. Television control by hand gestures. In *Proc. of the 1st Intl. Conf. on Automatic Face and Gesture Recognition*, 1994.

[GA75] J. A. Graham and M. Argyle. A cross-cultural study of the communication of extra-verbal meaning by gestures. *International Journal of Psychology*, 10:57–67, 1975.

[GGGZ05] Timothy Gatzke, Cindy Grimm, Michael Garland, and Steve Zelinka. Curvature maps for local shape comparison. In *Proc. Int. Conf. on Shape Modeling and Applications (SMI)*, pages 33–42, 2005.

[GKP03] Sylvie Gibet, Jean-François Kamp, and Franck Poirier. Gesture analysis: Invariant laws in movement. In *Proc. Gesture Workshop*, pages 1–9, 2003.

[Han05] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM*

symposium on User interface software and technology, pages 115–118, New York, NY, USA, 2005. ACM Press.

[HK07] S. Hermann and R. Klette. A comparative study on 2d curvature estimators. In *Int.Conf.on Computing: Theory and Applications*, pages 584–589, 2007.

[HL00] J. Hong and J. Landay. Satin: A toolkit for informal ink-based applications. In *Proc. UIST '00*, pages 63–72, 2000.

[HLLM02] Jason I. Hong, James A. Landay, A. Chris Long, and Jennifer C. Mankoff. Sketch recognizers from the end-user's, the designer's, and the programmer's perspective. In *In Proc. of AAAI 2002 Spring Symposium (Sketch Understanding Workshop)*, pages 73–77, March 2002.

[HS92] J. Hershberger and J. Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. In *Proc. of 5th Symposium on Data Handling*, pages 134–143, 1992.

[HS95] A.J. Heap and F. Samaria. Real-time hand tracking and gesture recognition using smart snakes. In *Proc. of Interface to Real and Virtual Worlds*, 1995.

[IGM98] J. M. Iverson and S. Goldin-Meadow. Why people gesture when they speak. *Nature*, 396:228, 1998.

[Imm07] ImmersionCorporation. http://www.immersion.com, last visited: Oct. 2007.

[Inc02] Merriam-Webster Inc. *Merriam-Webster's Medical Dictionary*. Merriam-Webster Inc., 2002.

[Int07] Intersense. http://www.isense.com/products, last visited: Oct. 2007.

[Iso01] Poika Isokoski. Model for unistroke writing time. In *Proc. CHI '01*, pages 357–364. ACM Press, 2001.

[JJL06] Jr. Joseph J. LaViola. Sketching and gestures 101. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 2. ACM Press, 2006.

[JR98] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. Technical Report 98/11, Cambridge Research Laboratory, 1998.

[Kad96] M. Kadous. Machine recognition of auslan signs using powergloves: Towards large-lexicon recognition of sign language. In *Proc. of the Workshop on the Integration of Gesture in Language and Speech*, pages 165–174, 1996.

[Ken80] A. Kendon. Gesticulation and speech: two aspects of the process of utterance. In *The relation of verbal and nonverbal communication*. The Hague, The Netherlands: Mouton, 1980.

[Ken86] Adam Kendon. Current issues in the study of gesture. In *The Biological Foundation of Gestures: Motor and semiotic Aspects*, pages 23–47. Lawrence Erlbaum Assoc., Hillsdale, 1986.

[Ken94] A. Kendon. Do gestures communicate? a review. *Research on Language and Social Interaction*, 27:175–200, 1994.

[Ken96] Kendon. An agenda for gesture studies. *Semiotic Review of Books*, 7(3):8–12, 1996.

[KH90] G. Kurtenbach and E. Hulteen. Gestures in human-computer communications. In B. Laurel, editor, *The Art of Human Computer Interface Design*, pages 309–317. Addison-Wesley, 1990.

[KHD04] Mathias Kolsch, Tobias Hllerer, and Stephen DiVerdi. Handvu: A new machine vision library for hand tracking and gesture recognition. demo at ISWC/ISMAR, 2004.

[KMSC91] R. M. Krauss, P. Morrel-Samuels, and C. Colasante. Do conversational gestures communicate? *Journal of Personality and Social Psychology*, 61:743–754, 1991.

[Kol] Mathias Kolsch. Handvu: Hand gesture recognition. http://www.movesinstitute.org/ kolsch/HandVu/HandVu.html.

[KS05] M. Karam and M. C. Schraefel. A taxonomy of gestures in human computer interactions. Technical Report ECSTR-IAM05-009, Electronics and Computer Science, University of Southampton, 2005.

[KT04a] Mathias Kolsch and Matthew Turk. Analysis of rotational robustness of hand detection with viola and jones method. In *Proc. ICPR*, 2004.

[KT04b] Mathias Kolsch and Matthew Turk. Fast 2d hand tracking with flocks of features and multi-cue integration. In *Proc. IEEE Workshop on Real-Time Vision for Human-Computer Interaction (at CVPR)*, 2004.

[KT04c] Mathias Kolsch and Matthew Turk. Robust hand detection. In *Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition*, May 2004.

[KT05] M. Kolsch and M. Turk. Hand tracking with flocks of features. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2005*, volume 2, June 2005.

[KTH04] M. Kolsch, M. Turk, and T. Hollerer. Vision-based interfaces for mobility. In *Proc. 1st Annual Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services*, 2004.

[LaV99] J. LaViola. A survey of hand posture and gesture recognition techniques and technology. Technical Report CS-99-11, Department of Computer Science, Brown University, Providence RI, 1999.

[LaV07] Joseph LaViola. Sketching and gestures 101. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 2. ACM Press, 2007.

[Log07] Logitech. http://www.logitech.com, last visited: Oct. 2007.

[LY02] Jae Y. Lee and Suk I. Yoo. An elliptical boundary model for skin color detection. In *Proc. Int. Conf. on Imaging Science, Systems and Technology, Las Vegas USA*, 2002.

[MA02] F. Mokhtarian and S. Abbasi. Shape similarity retrieval under affine transforms. *Pattern Recognition*, 35(1):31–41, 2002.

[McN92] David McNeill. *Hand and mind: What gestures reveal about thought.* University of Chicago Press, 1992.

[MM92] F. Mokhtarian and A.K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(8):789–805, August 1992.

[MT91] Kouichi Murakami and Hitomi Taguchi. Gesture recognition using recurrent neural networks. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 237–242, New York, NY, USA, 1991. ACM.

[Mul86] Axel Mulder. Hand gestures for hci: Research on human movement behavior reviewed in the context of hand centered input. Technical Report 96-1, Simon Fraser University, 1986.

[MVK06] S. Madhvanath, D. Vijayasenan, and T. M. Kadiresan. Lipitk: A generic toolkit for online handwriting recognition. In *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition*, 2006.

[Nap93] J. R. Napier. *Hands*. Princeton, N.J.: Princeton University Press, 1993.

[New93] Gregory B. Newby. Gesture recognition using statistical similarity. In *Proc. of Virtual Reality and Persons with Disabilities*, 1993.

[NMSG03] M. Nielsen, T. Moeslund, M. Storring, and E. Granum. A procedure for developing intuitive and ergonomic gesture interfaces for hci. In *Proc. of 5th Int. Workshop on Gesture and Sign Language based Human Computer Interaction, Genova, Italy*, 2003.

[NPL86] Jean Luc Nespoulous, Paul Perron, and Andre Roch Lecours. *The Biological Foundations of Gestures: Motor and Semiotic Aspects*. Lawrence Erlbaum Associates, Hillsdale, MJ, 1986.

[Peg07] PegasusTechnologies. http://www.pegatech.com, last visited: Oct. 2007.

[Pra07] William K. Pratt. *Digital Image Processing: PIKS Scientific Inside, 4th Ed.* Wiley-Interscience, 2007.

[PSH97] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. on PAMI*, 19(7):677–695, 1997.

[Rub91] Dean Rubine. Specifing gestures by example. In *Proc. of SIGGRAPH 91*, pages 329–337. ACM Press, 1991.

[Rus98] John C. Russ. *The Image Processing Handbook, 3rd Ed.* CRC Press, 1998.

[Sch84] E. A. Schegloff. On some gestures relation tospeech. In J. M. Atkinson and J. Heritage, editors, *Structuresof social action: Studies in conversational analysis*. Cambridge University Press, 1984.

[Sen07] SensableTechnologies. http://www.sensable.com, last visited: Oct. 2007.

[SKK01] T.B. Sebastian, P.N. Klein, and B.B. Kimia. Alignment-based recognition of shape outlines. *LNCS-2059*, pages 606–618, 2001.

[SKK03] T.B. Sebastian, P.N. Klein, and B.B. Kimia. On aligning curves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(1):116–125, January 2003.

[Soa04] J.M. Soares. Contribution a la communication gestuelle dans les environments virtuels colaboratifs, phd dissertation. Technical report, Institut National des Telecomunications and Universite de Technologie de Troyes, 2004.

[ST94] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Seattle*, 1994.

[Stu92] David J. Sturman. *Whole-hand Input*. PhD thesis, Massachusetts Institute of Technology, 1992.

[SZ94] David J. Sturman and David Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, 1994.

[Tag99] H.D. Tagare. Shape-based nonrigid correspondence with application to heart motion analysis. *IEEE Trans. Medical Imaging*, 18(7):570–578, 1999.

[TBvdP04] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: an interface for sketching character motion. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 424–431, New York, NY, USA, 2004. ACM Press.

[TSW90] C.C. Tappert, C.Y. Suen, and T. Wakahara. The state-of-the-art in online handwriting recognition. *IEEE Trans. Pattern Analysis Machine Intelligence*, 12:787–808, August 1990.

[VA95] P. Vamplew and A. Adams. Recognition and anticipation of hand motions using a recurrent neural network. In *Proc. IEEE International Conference on Neural Networks*, pages 2904–2907, 1995.

[VGD+05] Radu-Daniel Vatavu, Laurent Grisoni, Samuel Degrande, Christophe Chaillou, and Stefan-Gheorghe Pentiuc. Adaptive skin color detection in unconstrained environments using 2d histogram partitioning. *Advances in Electrical and Computer Engineering*, 5(12):101–106, 2005.

[VGP07] Radu-Daniel Vatavu, Laurent Grisoni, and Stefan-Gheorghe Pentiuc. Gesture recognition based on elastic deformation energies. In *Proc. of the*

7th International Workshop on Gesture in Human-Computer Interaction and Simulation GW 2007, Lisbon, Portugal, 2007.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Kauai*, volume 1, pages 511–518, 2001.

[VP06] Radu-Daniel Vatavu and Stefan-Gheorghe Pentiuc. Interacting with gestures: An intelligent virtual environment. In *The 1st International Conference on Virtual Learning ICVL 2006, Bucharest, Romania*, pages 293–299, 2006.

[VP08] Radu-Daniel Vatavu and Stefan-Gheorghe Pentiuc. Multi-level representations of gesture as command for hci. *Computing and Informatics (accepted, to appear)*, 2008.

[VPC05] R.D. Vatavu, S.G. Pentiuc, and C. Chaillou. On natural gestures for interacting in virtual environments. *Advances in Electrical and Computer Engineering*, 24(5):72–79, 2005.

[VPC+06] Radu-Daniel Vatavu, Stefan-Gheorghe Pentiuc, Christophe Chaillou, Laurent Grisoni, and Samuel Degrande. Visual recognition of hand postures for interacting with virtual environments. *Advances in Electrical and Computer Engineering*, 6(13):55–58, 2006.

[VSA03] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel based skin color detection techniques. In *Proc. Graphicon, Moscow Russia*, pages 85–92, 2003.

[VT82] P. Viviani and C. Terzuolo. Trajectory determines movement dynamics. *Neuroscience*, 7(2):431–437, 1982.

[Wat93] R. Watson. A survey of gesture recognition techniques. Technical Report TCD-CS-93-11, Trinity College Dublin, 1993.

[WB03] Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 193–202, New York, NY, USA, 2003. ACM Press.

[Web02] Andrew Webb. *Statistical Pattern Recognition*. John Wiley & Sons, LTD, 2002.

[Wex95] Wexelblat. An approach to natural gesture in virtual environments. *ACM Transactions on Computer-Human Interaction TOCHI*, 2(3):179–200, 1995.

[Wil04] Andrew D. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76, New York, NY, USA, 2004. ACM Press.

[Wil05a] A. Wilson. Playanywhere: A compact tabletop computer vision system. In *Symposium on User Interface Software and Technology (UIST)*, 2005.

[Wil05b] Andrew D. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Emerging technologies*, page 25, New York, NY, USA, 2005. ACM Press.

[Wil06] Andrew D. Wilson. Robust computer vision-based detection of pinching for one and two-handed gesture input. In *UIST '06: Proceedings of the*

19th annual ACM symposium on User interface software and technology, pages 255–258, New York, NY, USA, 2006. ACM Press.

[WR06] A. Wilson and D. C. Robbins. Playtogether: Playing games across multiple interactive tabletops. In *IUI Workshop on Tangible Play: Research and Design for Tangible and Tabletop Games*, 2006.

[WS99] J. Weissmann and R. Salomon. Gesture recognition for virtual reality applications using datagloves and neural networks. In *Proc. of Int. Joint Conf. on Neural Networks, Washington DC, USA*, pages 2043–2046, 1999.

[WS07] A. Wilson and R. Sarin. Bluetable: Connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In *Graphics Interface*, 2007.

[WWL07] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168, New York, NY, USA, 2007. ACM Press.

[ZCF+04] Liang-Guo Zhang, Yiqiang Chen, Gaolin Fang, Xilin Chen, and Wen Gao. A vision-based sign language recognition system using tied-mixture density hmm. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 198–204, New York, NY, USA, 2004. ACM Press.

[ZLB+87] Thomas G. Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A hand gesture interface device. In *CHI '87: Pro-*

ceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface, pages 189–192, New York, NY, USA, 1987. ACM Press.

Appendix A

Computing the Catmull-Rom spline coefficients

The Catmull-Rom spline is completely defined by specifying its control points p_i and the associated tangents t_i. Each segment of the spline between two consecutive control points, p_i and p_{i+1} is a cubic polynomial of the form:

$$\lambda(u) = \sum_{k=0}^{3} c_k u^k \tag{A.1}$$

where u is a local parameter that varies between $[0, 1]$. The coefficients $c_k, k = 0, 3$ may be computed for each segment using end continuity conditions:

$$\begin{cases} p_i & = \lambda(0) & = c_0 \\ p_{i+1} & = \lambda(1) & = c_0 + c_1 + c_2 + c_3 \\ t_i & = \lambda'(0) & = c_1 \\ t_{i+1} & = \lambda'(1) & = c_1 + 2c_2 + 3c_3 \end{cases} \tag{A.2}$$

Consecutively, the system becomes:

$$\begin{cases} c_0 & = p_i \\ c_0 + c_1 + c_2 + c_3 & = p_{i+1} \\ c_1 & = r_i \cdot (p_i - p_{i-1}) + (1 - r_i) \cdot (p_{i+1} - p_i) \\ c_1 + 2c_2 + 3c_3 & = r_{i+1} \cdot (p_{i+1} - p_i) + (1 - r_{i+1}) \cdot (p_{i+2} - p_{i+1}) \end{cases} \tag{A.3}$$

$$\begin{cases} c_0 &= p_i \\ c_1 &= r_i \cdot (p_i - p_{i-1}) + (1 - r_i) \cdot (p_{i+1} - p_i) \\ c_2 &= (2r_i - r_{i+1} + 1) \cdot (p_{i+1} - p_i) + r_i \cdot (p_i - p_{i-1}) - (1 - r_{i+1}) \cdot (p_{i+2} - p_{i+1}) \\ c_3 &= (r_{i+1} - r_i - 1) \cdot (p_{i+1} - p_i) + (1 - r_{i+1}) \cdot (p_{i+2} - p_{i+1}) \end{cases} \tag{A.4}$$

and further,

$$\begin{cases} c_0 &= p_i \\ c_1 &= -r_i \cdot p_{i-1} + (2r_i - 1) \cdot p_i + (1 - r_i) \cdot p_{i+1} \\ c_2 &= 2r_i \cdot p_{i-1} + (-4r_i + r_{i+1} - 1) \cdot p_i + (2r_i - 2r_{i+1} + 2) \cdot p_{i+1} + (r_{i+1} - 1) \cdot p_{i+2} \\ c_3 &= -r_i \cdot p_{i-1} + (2r_i - r_{i+1} + 1) \cdot p_i + (2r_{i+1} - r_i - 2) \cdot p_{i+1} + (1 - r_{i+1}) \cdot p_{i+2} \end{cases} \tag{A.5}$$

The system may be put under a concise matrix form:

$$C = G \cdot P \tag{A.6}$$

where C is the coefficients vector $C = [c_0\, c_1\, c_2\, c_3]^T$, P is the control points vector $P = [p_{i-1}\, p_i\, p_{i+1}\, p_{i+2}]^T$ and G is the geometry matrix as given by:

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -r_i & 2r_i - 1 & 1 - r_i & 0 \\ 2r_i & -4r_i + r_{i+1} - 1 & 2r_i - 2r_{i+1} + 2 & r_{i+1} - 1 \\ -r_i & 2r_i - r_{i+1} + 1 & 2r_{i+1} - r_i - 2 & 1 - r_{i+1} \end{bmatrix} \tag{A.7}$$

The determinant of the geometry matix G may be succesively reduced to:

$$|G| = - \begin{vmatrix} -r_i & 1 - r_i & 0 \\ 2r_i & 2r_i - 2r_{i+1} + 2 & r_{i+1} - 1 \\ -r_i & 2r_{i+1} - r_i - 2 & 1 - r_{i+1} \end{vmatrix} \tag{A.8}$$

and further to

$$|G| = r_i \cdot (1 - r_{i+1}) \cdot \begin{vmatrix} 1 & 1 - r_i & 0 \\ -2 & 2r_i - 2r_{i+1} + 2 & -1 \\ 1 & 2r_{i+1} - r_i - 2 & 1 \end{vmatrix} \tag{A.9}$$

and after evalation:

$$|G| = r_i \cdot (1 - r_{i+1}) \tag{A.10}$$

which means that the above system has alwas an unique solution C if $r_i \neq 0$ and $r_{i+1} \neq 1$.