Visual Input for Pen-Based Computers

Thesis by

Mario Enrique Munich

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2000
(Submitted January 21, 2000)

A mis padres,

a Pili,

a Mirena,

a mis hermanos,

a Lela, Nina y Pita.

Acknowledgements

Writing a thesis was just the final, albeit long and painful, step of the process of earning my Philosophy Doctorate in Electrical Engineering. I am very grateful to many people that helped me obtain my degree.

I wish to start by thanking my parents. They gave me the freedom to envision new horizons and the encouragement to pursue my dreams even if it meant to be separated and far away from each other. Without their initial momentum and their continuous support, it would have been impossible for me to get where I stand. I want to mention also my brothers *Pili* and *Rulo* that were always there for me. Although we were far apart, they always believed in me and they were always ready to help me in any way possible.

I would like to show my deepest gratitude to my dear wife *Pili*. She has been my most important support at Caltech, my guiding star in the dark nights, giving me the strength to keep going; she has filled all these years with fun, love and joy, and she has given me the most important gift on earth, the light of my eyes, my daughter Mirena Ainara.

I particularly wish to thank my advisor Pietro Perona. It was long ago that I walked into his office looking for a piece of advice and ended up convincing him that I would be a good addition to his research group, not to mention to his culinary delights. He introduced me to the wonders of Computer Vision and to the enjoyment of scientific research. His continuous support, his advice and his guidance have made an important difference in my work. I really enjoyed being part of his research group as well as interacting with him at both the professional and the personal level.

I would like to mention my great friends and lab-mates Jean-Yves Bouguet, Luis Goncalves and Enrico Di Bernardo. They made fun and enjoyable many sleepless nights of homework and paper preparation. They were also great collaborators and an excellent source of advice and ideas. They were always ready to listen and to discuss any problem, no matter what the problem was. We gave birth to a very special group of friends, a.k.a. "The Agency," along with Wayez Ahmat, Muruhan Rathinam, Chris Moser, George Barbasthasis, Dieter Koller, Gudrun Socher and Alberto Pesavento. "The Agency" accomplished many difficult and quite fun missions all around the globe under the lead of the fearless Commander Wayez.

I want to make a special mention of my roommate, Muruhan Rathinam, who showed me the marvels of vegetarian cooking. He was also extremely kind in accepting that Pili, my-wife-to-be, would live together with us. I also enjoyed all the enriching discussions that we had on topics ranging form food to politics to academics.

The vision group has been a great source of inspiration and stimulation, as well as a fun place. I enjoyed collaborating with all the past and present members of the group: Stefano Soatto, Mike Burl, Marco Tartagni, Enrico Ursella, Markus Weber, Max Welling, Alan Bond, Arrigo Benedetti, Xiaolin Feng, Yang Song, Silvio Savarese. I will truly miss the group-meetings on the beach, the group asados, and the enjoyable and fruitful atmosphere of the vision lab.

During these years, I had the pleasure of meeting so many interesting and wonderful people whose friendship was invaluable to make me feel at home at Caltech. I had the pleasure and the privilege of joining them in many activities such as the organization of "Club Latino" and "Semana Latina," uncountable asados, retreats, trips, etc. I would like to mention them in a random order and I would like to thank to everyone whose name have slipped from my memory at this moment, Douglas Varela, Karina Montilla, Fernando and Malena Paganini, Jorge Tierno, Weng-Ki Ching, Diego Dugatkin, Oscar and Maria Elena Lovera (also Nati, Brian and Sergio), Alex Backer, Eva Peral, Federico Spedalieri, Ruben Krasnapolsky, Julian Chaubell, Adrian and Patri Lew, Raul and Flavia Radovitzky, Eduardo Repetto, Javier Gonzalez, Maria Eugenia Hernandez, Pablo Parrillo, Melissa Saenz, Rogelio Addobatti, Alberto and Maria Jose Cerpa, Christine and Kurt Schenk, Martin Basch, Xavier Cartoixa, Alvaro Gonzalez, Anna Johansen, Alfredo Martinez, Roberto Zenit.

I would also like to thank the other members of my committee, Professors Yaser Abu-Mostafa, James R. Arvo, Demetri Psaltis and Dr. Michael Burl, for their help and support. I wish to thank Professor Slobodan Cuk for having believed that I would be a good Caltech student. I want to mention Professors R. D. Middlebrook, P. P. Vaidyanathan, C. Mead, and Y. Abu-Mostafa for having taught me his philosophycal view of electrical engineering rather than a simple curricular content.

Finally, I wish to thank all the people that willingly participated in my experiments. Their collaboration was extremely important in the development of the thesis.

Abstract

The development of computer technology has had a parallel evolution of the interface between humans and machines, giving rise to interface systems inspired by human communication. Vision has been demonstrated to be the sense of choice for face recognition, gesture recognition, lip reading, etc. This thesis presents the design and implementation of a camera-based, human-computer interface for acquisition of handwriting. The camera focuses on the sheet of paper and images the hand writing; computer analysis of the resulting sequence of images enables the trajectory of the pen to be tracked and the times when the pen is in contact with the paper to be detected. The recovered trajectory is shown to have sufficient spatio-temporal resolution and accuracy to enable handwritten character recognition.

Signatures can be acquired with the camera-based interface with enough resolution to perform verification. This thesis describes the performance of a visual-acquisition signature verification system, emphasizing the importance of the parameterization of the signature to achieving good classification results. The generalization error of the verification algorithm is estimated using a technique that overcomes the small number of example signatures and forgeries provided by the subjects.

The problem of establishing correspondence and measuring the similarity of a pair of planar curves, in our case a pair of signatures, arises in many applications in computer vision and pattern recognition. This thesis presents a new method for comparing planar curves and for performing matching at sub-sampling resolution. The analysis of the algorithm as well as its structural properties are described. The performance of the new technique is evaluated for the problem of signature verification and compared to the performance of the well-known Dynamic Programming Matching algorithm.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Human-Machine interfaces using vision

Over the past few decades, we have witnessed the birth and development of the computer industry. Along with the evolution of computer technology, there has been a corresponding evolution of the interfaces between humans and machines. In the early days, the user (or operator) had to activate certain switches in a particular sequence in order to achieve a desired action. Later, the input was provided with punch cards and the output was extracted with paper tapes. As machines evolved, they were connected to keyboards, CRT's and monitors to provide "real-time" text input and output. More recently, advances in memory, video and microprocessor technology have allowed computers to enter in the era of graphical interfaces involving a whole set of pointing devices (like mice, digitizing tablets, joysticks, touch-screens and track-balls) and graphical user interfaces.

Technological advances in miniaturization of electronic devices in the last few years have opened a new market for portable computers. Laptop and notebook computers as well as Personal Digital Assistants (PDA's) like the popular "Palm Pilot" are in wide use nowadays. Cellular phone manufacturers are adding computing power in their new generation of phones, e.g., Sprint started offering Internet access using a cellular phone in mid-1999, Qualcom is advertising a device that is a PDA and a cellular phone at the same time, etc. The trend indicates that more computing power will be packed inside smaller and easier to carry devices. The ultimate portability would be achieved with so-called "wearable computers." These computers would be worn, much as eyeglasses or clothing are worn, and would interact with the user based on the context of the situation. Part of the success of such computers would depend on the user feeling comfortable and getting accustomed to wear them. One successful example of people adapting to wear a "computer" is provided by mechanical and electronic watches. These devices have made their way into the everyday life of millions of people who depend on them and wear them without even noticing. Another question that will affect the success of wearable computers is how simple is the interaction with the device? The human-machine interface as it presently exists, i.e., based on keyboards and screens, is becoming a miniaturization bottleneck since the resolution of the human eye limits the size of the screen, and the dimensions of the fingers limit the minimum size of keyboards and mice. Therefore, there is a need for humans to communicate with computers in alternative ways, that are more intuitive and involve smaller hardware (ideally smaller than 1 cm^3). There are already some attempts to overcome this interface miniaturization bottleneck. Sony is advertising a computer in the form of a monocle headset, in which a mini-monitor detaches from the headset and covers one eye. The computer screen is displayed on this monitor while a joystick-type device is used to provide input. Many PDA's use a stylus instead of a keyboard as input device, reducing the computer's size at some expense in the throughput of the input element. Data is acquired by the computer by tracking the motion of the stylus on a flat touch-sensitive display which provides immediate graphical feedback to the user. These PDA's accept input in the form of characters and pen gestures and their performance heavily relies on a character/pen gesture recognition engine.

The development of novel and friendly human-machine interfaces is crucial for extending the use of computers to all segments of society and to all cultures (especially to the ones that do not have a suitable alphabet for entering text via a keyboard). Also, new interfaces will allow the acquisition of mathematical formulas, hand-drawn figures, sketches, etc. Humans acquire information from the exterior world by using their senses (audition, vision, olfaction, tactile, and taste). However, social interaction among humans is carried out mainly through the audio channel (audition) and the visual channel (vision). Speech and music are the main components of the audio channel, while hand, face and body gesturing, handwriting and drawing are the main components of the visual channel. Humans learn to use these communication channels from their very childhood and depend on them in their daily life. In some sense, one could think of these communication channels as a built-in resource to be used in the design of intuitive and simple to use interfaces.

Cameras and microphones are becoming ubiquitous in desktop computer systems. These devices can be made as small as $1\,\mathrm{cm}^2 \times 1\,\mathrm{mm}$ with current VLSI technology and can be easily integrated into PDA's. Techniques based on these types of inputs, in particular computer vision, have a significant role to play in the development of the interfaces of the future. Vision has been demonstrated to be the sense of choice for human face detection [9, 10, 11] and recognition [75, 17], facial expression interpretation [46, 22, 15], lip reading [25, 47, 43, 6], head orientation detection [14], eye gaze tracking [14, 29, 16], three-dimensional finger pointing [13], hand tracking [53] and gesture interpretation [61, 73, 55], and body pose tracking [32, 4, 5, 33].

Face detection will allow machines to determine whether a user is present in the field of view of the camera while face recognition will permit machines to identify each user without the need for explicit passwords. This will be particularly useful in systems designed for cooperative work, where multiple users share an electronic and physical workspace, or for access control to a particular resource like an automatic teller machine, restricted areas of a company, a credit card purchase, etc. Facial expression interpretation will play a big role in "affective computing" [56], allowing machines to detect the emotions of a user and respond appropriately. For example, by providing more assistance when the user looks puzzled or frustrated, or by playing a favorite tune when the user is sad, etc. Head tracking and detection, and facial expression could provide a very useful computer interface for the physically impaired, some of whom can only communicate using head gestures. Eye gaze, in the same manner as finger pointing, may be useful as a pointing device to substitute for the mouse. Gaze aside, the head orientation can be used to guide a remote, synthesized "clone" face for low bandwidth video conferencing. Lip reading provides an additional channel of information for speech recognition that will facilitate the development of robust speech recognition-based computer interfaces. Lip reading interfaces may also prove effective as interfaces for the speech impaired. Models developed for representing facial expression and lip movement may also be used to build multimedia documents which are able to represent emotions. Body pose tracking will facilitate the development of immersing virtual reality systems [27, 44]. Users may move freely within environments, reach out and touch objects, interact with avatars, virtual humans and other agents, etc. Body tracking will also allow building models of human motion to be used for realistic character animation [31].

Handwriting on paper is one very important way of communication among humans. People send postcards or letters in order to keep in touch with friends and family. They use tiny post-it notes to remember certain duties or large pads of flip-chart paper to save ideas during brainstorming meetings. They read books, newspapers and memos. Chances are that you are reading these words on paper. Therefore, despite the claims about the "paper-less" office, in which information on paper is replaced by information on the screen of a computer, we find that paper is used as much, if not more than before. However, electronic documents provide a number of properties that are lacking in paper documents, such as spelling correction, electronic mail, keyword searching, numerical calculation, time-stamping, and language translation.

An alternative to the replacement of one type of document by the other is to augment the physical world with computers. The Digital Desk [80, 79] developed at Xerox PARC merges physical objects (paper and pencils) with their electronic counterparts using computer vision and video projection. A computer screen is projected onto a physical desk using a video projector, while a camera is set up to watch the workspace such that the surface of the projected image and the surface of the image area coincide. A tablet digitizer or a finger tracked by the camera [17, 18] is used to input mouse-type of information into the system allowing one to select or highlight words on paper documents, cut and paste portions of text, draw figures, etc. The Liveboard [23] developed by Xerox is similar in concept to the digital desk. This device is the replacement for the pads of flip-chart paper used in meetings. A computer screen is projected onto a whiteboard and a cordless pen is used as input. The same image could be displayed onto boards placed at different locations and the input from each of these boards overlaid on all of them, allowing in this way for remote collaboration. The Digital Desk and the Liveboard are steps towards the integration of paper documents into the computing environment and towards the development of handwriting-based human-computer interfaces.

The development of an interface that automatically acquires handwriting using a video camera and computer vision techniques has not been addressed in the literature. This interface would be very small since it would only require a camera as input device, reducing the clutter usually added to desktop computers by tablet digitizers and full-page scanners. This interface would allow the user to write on a piece of paper with a normal pen, eliminating the requirement of writing with strange stylus on a slippery plastic surface. A complete pen-based interaction with the computer could be devised with this interface since in addition to handwriting mouse-type commands, passwords in the form of signatures, drawings, etc., could be acquired.

In the first part of this thesis, we present the design and implementation of a human-computer interface for acquisition of handwriting using a single camera. The input interface consists of a camera, a piece of paper, and a normal pen. The camera focuses on the sheet of paper and images the hand writing; computer analysis of the resulting sequence of images enables the trajectory of the pen to be tracked and the times when the pen is in contact with the paper to be detected.

Tracking the position of the pen tip instead of taking a still picture of the ink trace left on the paper serves two purposes. First, more information is provided to the handwriting recognition engine since the dynamics of writing is captured along with the handwritten trajectory. Second, recognition could be performed as the user writes providing better man-machine interaction and a simpler setting for correcting recognition errors.

The second part of this thesis is devoted to developing an automatic identification system using the camera-based interface for handwriting capture. A signature or a string of characters would be captured by the interface for performing verification of the claimed identity. Signatures are regarded in the literature [41] as the result of a ballistic action, without any visual feedback involved. Hence, we use signatures in our automatic identification system since signatures would be more stable and more difficult to imitate than a sequence of characters. This system would be one component of a complete visual pen-based computer environment, where the signature could replace the password for logins into computer systems or personal Internet accounts. Measuring the similarity between signatures in order to detect forgeries involves the comparison of two two-dimensional curves. Part of the contribution of this thesis is the development of a novel technique for curve comparison that is able to work at subsample resolution.

1.2 Outline of the thesis

Chapter 2 describes the design and development of a human-machine interface for acquisition of handwriting using a single camera. The various components of the interface, its real-time implementation, and the trade-offs faced in the design are also described.

Chapter 3 presents the development of a signature verification system built around the camera-based interface for handwriting acquisition. Two different databases of signatures were collected with the visual interface and used to assess the performance of the signature verification system. We describe the application of the well-known method of Dynamic Programming Matching to compare the acquired signatures. We also discuss the effect of the parameterization of the signatures on the performance as well as the method used to evaluate the performance and estimate the generalization error of the system.

Chapter 4 introduces a novel algorithm for matching planar curves and its application to signature verification. The algorithm is based on the general method of Dynamic Programming and is able to match planar curves at subsample resolution.

Finally, in chapter 5 we summarize our contributions and findings, and discuss directions for future research.

Chapter 2 Vision System for Pen Tracking

2.1 Preliminaries

This chapter describes the design of a system that captures both the spatial and temporal aspects of handwriting using a standard consumer-quality video camera as input device. Conventional interfaces for handwriting capture include tablet digitizers and touch-sensitive screens. Tablet digitizers come in several sizes ranging from 10 cm x 12 cm to 30 cm x 45 cm with weights ranging from 0.5 kg to 4 kg. Their typical temporal sampling frequency is 200 Hz and their cost ranges from $100 to $400. Sensitive screens are manufactured using a different type of technology. The sensitive area is usually customized to fit the desired screen size and adds $200 to $300 to a monitor's price. Cameras are ubiquitous in current desktop computer systems due to the drop in pricing and advances in manufacturing technology. Thus, there would be no need to buy additional hardware for the implementation of the visual interface for handwriting.

The cutoff frequency of handwritten strokes is below 20 Hz as shown in references [78, 69, 41] (different researchers propose different cut-off values depending on the level of energy used as a threshold of significance). Typical cameras have spatial resolution of 480x640 pixels (rows x cols) and frequency of 30 Hz. Most cameras are interlaced, so the actual images have a maximum resolution of 240x640 pixels at a frequency of 60 Hz. Given that the cut-off frequency of handwriting is below 20 Hz, working at 60 Hz, we are well above the Nyquist frequency of handwriting.

Our first experiment was intended to verify that a legible handwritten trace could be obtained by tracking the position of the pen tip in a sequence of images. We videotaped a subject writing on a piece of paper and we manually identified the position of

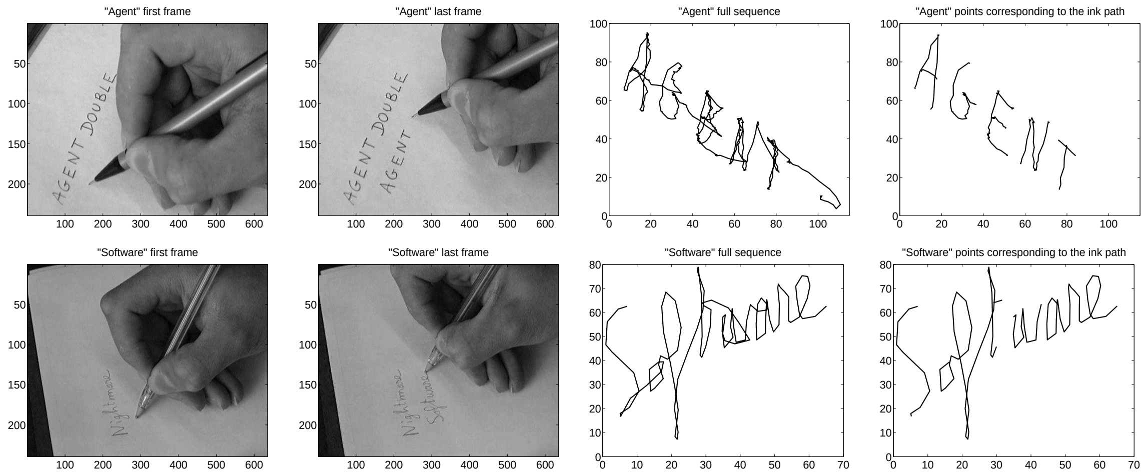


Figure 2.1: The first two columns show the first and last frames of two different sequences. The third column displays the handwritten trajectories obtained via manual tracking of the pen tip. The fourth column shows only the strokes that left an ink trace on the paper. We observe that after discarding the strokes that corresponds to movements above the paper (no pen contact with the paper), the handwritten sequences are clearly legible. All coordinates are measured in pixels.

the pen tip in each image of the sequence using a mouse. The first sequence named "Agent" was traced at 30Hz and the second sequence named "Software" was traced at 60Hz. Figure 2.1 shows the handwritten trajectories obtained with this manual tracking.

We observe that the trajectories are a bit noisy, especially the one tracked at 30Hz. The pen tip position is collected for all the images of the sequence, including frames in which the pen is actually writing on the paper and frames in which the pen is traveling above the paper. Full trajectories of the pen tip, as shown in the third column of figure 2.1, are somewhat difficult to read. However, after taking away the strokes that correspond to the pen moving above the paper and leaving only the strokes that correspond to the pen down on the paper, the trajectories are clear enough to enable one to easily read what was written. This simple experiment proves that there is sufficient information in a video sequence to reconstruct the pen trajectory. Therefore, we will describe in this chapter a system that automatically captures handwriting by following the position of the pen tip in a sequence of images provided by an off-the-shelf video camera. The experiment also revealed an important difference between this type of system and conventional handwriting capture devices: the continuous trajectory that we obtain by tracking the position of the pen tip in each of the images in the sequence must be divided into strokes corresponding to ink trace (pen down) and strokes corresponding to movement above the paper (pen up). With tablet digitizers the state of the pen (up or down) is mechanically sensed.

The remainder of this chapter is organized as follows. Section 2.2 describes the various components of the interface, section 2.3 presents the method developed for detecting pen up strokes, and section 2.4 shows the experimental results.

2.2 System description

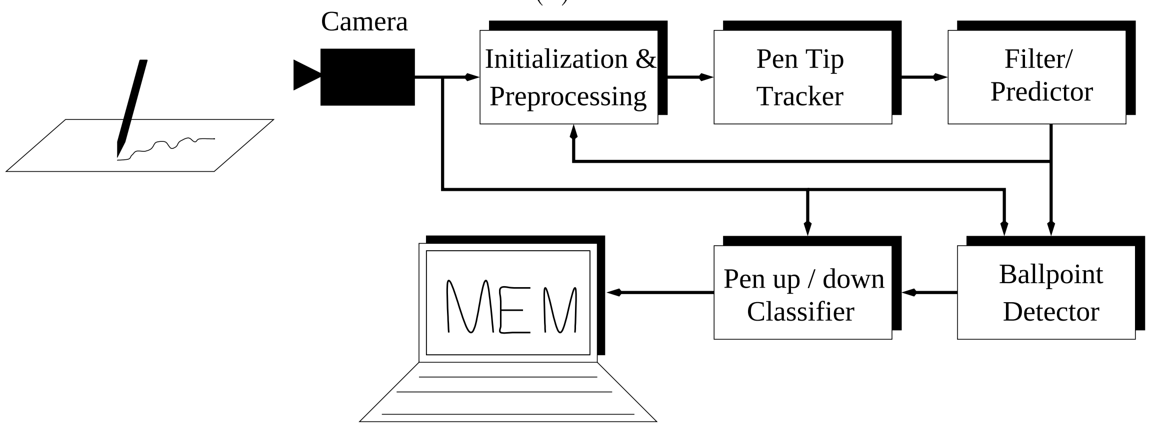
Figure 2.2 shows the block diagram of the system and the experimental setup. The image captured by the camera is shown on the screen of the computer to provide visual feedback to the user. The system does not require any calibration. The user has the flexibility of arranging the relative positions of the camera and the piece of paper in order to write comfortably as well as to provide the system with a clear sight of the pen tip.

The camera feeds a sequence of images to the preprocessing stage. This block performs initialization of the algorithm, i.e., it finds the initial position of the pen and selects a template (rectangular subregion of the image) corresponding to the pen tip. In subsequent frames, the preprocessing stage has only the function of cutting a piece of image around the predicted position of the pen tip and feeding it into the next block. The pen tip tracker has the task of finding the position of the pen tip in each frame of the sequence. The ballpoint detector finds the position of the very end of the pen tip, i.e., the place where the pen is in contact with the paper when the user is writing*. The filter is a recursive estimator that predicts the position of the tip in the next frame based on an estimate of the current position, velocity and acceleration of the pen. The filter also estimates the most likely position of the pen

*The term ballpoint is loosely used to indicate the actual ballpoint of pens and the pencil lead of pencils.

tip for missing frames. Finally, the last block of our system checks for the presence of ink on the paper at the positions where the pen's ballpoint was detected.

(a)



(b)

Figure 2.2: (a) Block Diagram of the system. The camera feeds a sequence of images to the preprocessing stage. This block initializes the algorithm and selects the template to perform the tracking of the pen tip. The tip tracker obtains the position of the pen tip in each image of the sequence. The filter predicts the position of the pen tip in next image. The ballpoint detector finds the position of the very end of the pen tip, i.e., the place where the pen is in contact with the paper when the user is writing. Finally, the last block of our system checks for the presence of ink on the paper at the positions where the pen's ballpoint was found. (b) Experimental setup. The image captured by the camera is shown on the screen of the computer to provide visual feedback to the user. The system does not require any calibration. The user has the flexibility of arranging the relative positions of the camera and the piece of paper in order to write comfortably as well as to provide the system with a clear sight of the pen tip.

2.2.1 Initialization and preprocessing

The first problem to be solved is to detect and localize the position of the pen tip in the first frame and to select the template to be used for detection in subsequent frames. There are two possible scenarios:

1.- The user writes with a pen that is familiar to the system.

2.- An unknown pen is used.

The familiar-pen case is easy to handle: the system may use a previously stored template representing the pen tip and detect its position in the image by correlation.

There are a number of methods to initialize the system when the pen is unknown. Our initialization method is a semi-automatic one that requires a small amount of user cooperation. We assume that the user is writing with a dark-colored pen on a light-colored piece of paper. We display the image captured by the camera on the screen of the computer. A rectangular box is overlaid on this image as shown in figure 2.3(a). The user is required to place the pen tip inside the displayed box, ready to start writing. The system watches for activity within this box, which is measured by image differencing between frames. When the number of pixels activated by image differencing is big enough as shown in figure 2.3(b), the system assumes that there is an object that entered the box and it then starts a waiting period until the object remains quiet. The user has, in this way, the possibility of placing the pen within the box and taking a comfortable position before starting to write. After the activity within the box has returned to low for a period of time (bigger than 200 ms), the system acquires the pen tip template, sends an audible signal to the user, and starts tracking.

Figure 2.4 shows a sketch of the pen tip, which is assumed to be roughly conical (true for most commercial pens). Hence, the projection of the pen tip onto the image plane will be a triangle. One of the borders of this triangle corresponds to the edge between the pen tip and the user's finger and the two other boundaries correspond to the edge between the pen tip and the piece of paper. Detection and extraction of the pen tip template is reduced to finding the boundary points of the pen tip, computing the corresponding centroid and cutting a portion of the image around the centroid. The edges between the pen tip and the paper have bigger contrast than the edge

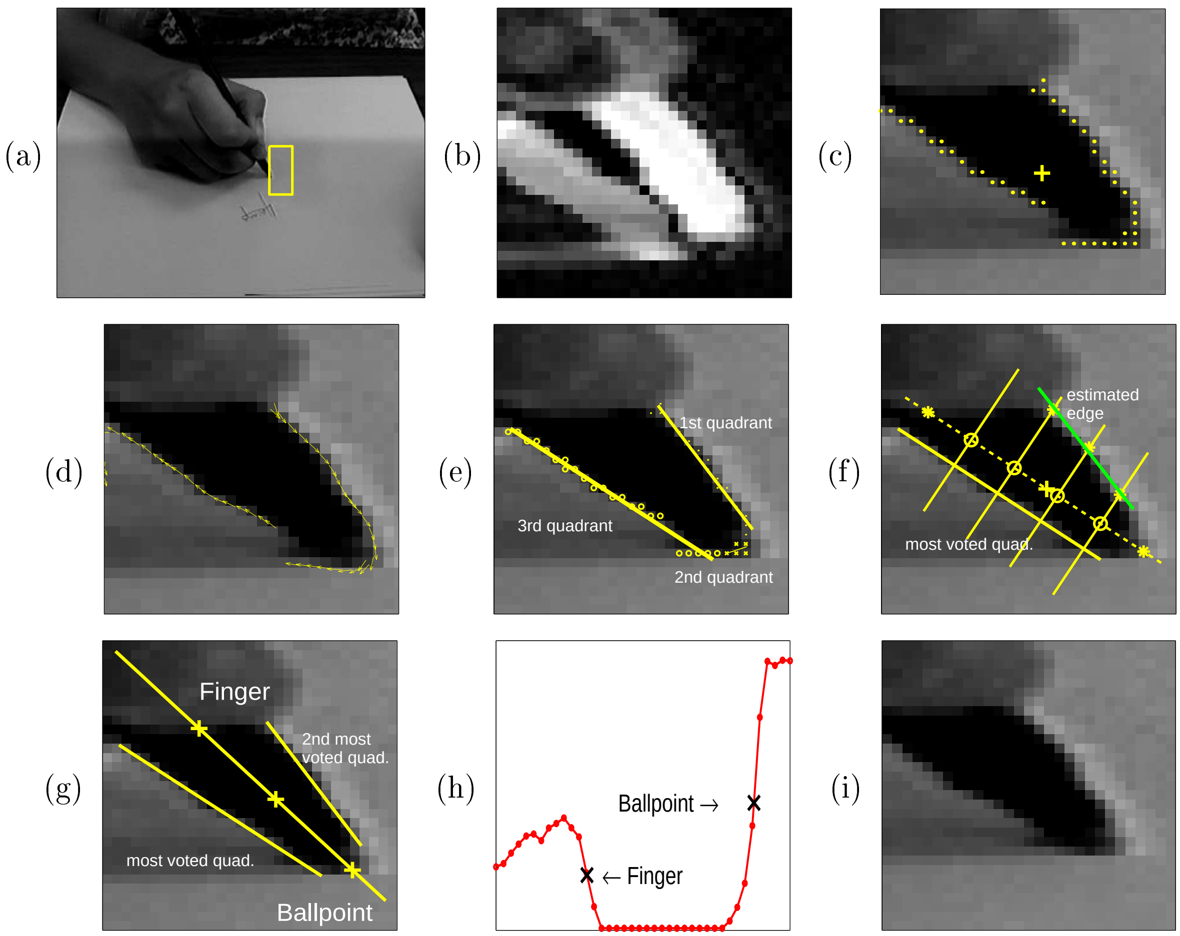


Figure 2.3: (a) Image provided by the camera with the rectangular box overlaid. The user is placing the pen inside the rectangular box. (b) Result of image differencing when the pen enters the tip acquisition area. (c) Output of Canny's edge detector used for extracting the boundary of the pen tip. The cross indicates the centroid of the boundary points. (d) Orientation of the edge elements obtained with Canny's detector. (e) Clustering of the edge elements into the four quadrants and lines indicating the mean orientation in each of the quadrants. (f) Detection of the boundary edge that was missing, using the estimated position of the centroid of the pen tip and the orientation of the other edge. (g) Boundary lines obtained by accumulating the information provided by the edge detector across different frames. Pen tip axis extracted as the mean of the boundary lines. Position of the tip, finger and centroid along the axis. (h) Profile of the image across the estimated pen tip axis, that is used to find the positions of the ballpoint and the finger by performing a 1D edge detection. (i) Template of the pen tip extracted automatically.

between the pen tip and the finger, thus, we only look for these two boundaries in the detection and extraction of the template.

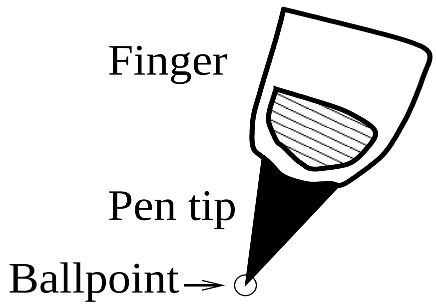


Figure 2.4: Pen tip model assumed for the initialization.

The boundaries of the pen tip are located using Canny's edge detector [12] as shown in figure 2.3(c). Sub-pixel resolution in the location of the edge elements is achieved by fitting a parabolic cylinder to the contrast surface in the neighborhood of each pixel. Since detection and extraction of the pen tip from a single frame is not very reliable due to changes in illumination, the system collects information about the pen tip for a few frames before extracting the template. The algorithm is summarized as follows:

1.- Compute the difference between current image and previous image within the rectangular box until a sufficient number of pixels have a difference value bigger than a predefined threshold (see figure 2.3(b)), then go to step 2.

2.- Compute image difference between current and previous images within the rectangular box until there is no activation for a number of frames, then go to step 3.

3.- Apply Canny's edge detector to the neighborhood of the image inside the mentioned box. Fit a parabolic cylinder to the contrast surface in the neighborhood of each pixel.

4.- Select only the pixels of the neighborhood that have sufficient contrast and whose corresponding parabolic cylinder has its axis close enough to the center of the pixel (see figure 2.3(c) and (d)).

5.- Get the centroid of these activated pixels (see figure 2.3(c)).

6.- Group the orientation of the activated pixels into the four quadrants† and get the mean orientation and the number of activated pixels per quadrant (see figure 2.3(e)).

7.- Repeat steps 3–6 for several frames and then go to step 8.

8.- Compute the mean position of the centroids computed in 5.

9.- Consider the most voted quadrant (which represents the boundary of the pen tip detected most reliably) and compute the corresponding mean orientation across frames. If the most voted quadrant does not have enough votes, abort the extraction of the pen tip and emit a sound signal to let the user know of the error condition.

10.- Consider the second most voted quadrant (which represents the second strongest detected boundary of the pen). If it does not have enough votes, recompute its position and orientation using the current image and the results of 8 and 9. Given the mean centroid position and the estimated orientation of one of the boundaries of the pen tip, the profile of the image is searched perpendicular to this orientation in order to find points with maximum contrast. These points are used to estimate the location of the other boundary of the pen tip (see figure 2.3(f)).

11.- Calculate the pen tip's orientation as the mean of the orientations obtained in steps 9 and 10 (see figure 2.3(g)). The mean orientation is computed taking into consideration the quadrant information of steps 9 and 10 in order to avoid problems with the inherent wrap-around $[0, 360^o]$ of angular quantities.

†The term quadrant refers to any of the four sections in which a plane is divided by rectangular coordinate axes lying in that plane. The first quadrant includes orientations between 0 and $\frac{\pi}{2}$, the second quadrant comprehends orientations between $\frac{\pi}{2}$ and π, the third quadrant involves orientations between π and $\frac{3\pi}{2}$, and the fourth quadrant includes orientations between $\frac{3\pi}{2}$ and 2π.

12.- Get the profile of the image along a line that passes through the centroid obtained in step 6 with the orientation calculated in step 9 (see figure 2.3(h)).

13.- Find the position of the ballpoint and the finger in this profile by performing a 1D edge detection on the image profile. Recompute the position of the centroid as the mean of the locations of the finger and the ballpoint (see figure 2.3(g-h)).

14.- Extract the template of the pen tip by selecting an area of the image of adequate size around the centroid computed in step 11 (see figure 2.3(i)).

The acquisition of the pen tip template is performed only at the beginning of the acquisition. The function of this module in subsequent frames is only to extract a region of interest centered around the predicted position of the pen tip. This region of interest is used by the following block of the system to detect the actual position of the pen tip.

2.2.2 Pen tracking

The second module of the system has the task of computing the position of the pen tip in the current frame of the sequence. The solution of this task is well known in the optimal signal detection literature [74, 28]. Assuming that the signal to be detected is known exactly except for additive white noise, the optimal detector is a matched filter, i.e., a linear filter that looks like the signal one is trying to detect. In our case, the signal consists of the pixels that represent the pen tip and the noise has two components: one component is due to noise in the acquisition of the images and the other one is due to changes in the apparent size and orientation of the pen tip during the sequence of images. The acquisition noise is the result of a combination of many factors like changes in illumination due to light flickering or automatic gain of the camera, quantization noise, changes in gain of the frame grabber, etc., where not all these factors are additive. Changes in the apparent size and orientation of the pen while the user is writing significantly distorts the image of the pen tip, as shown in figure 2.5. Clearly neither component of the noise strictly satisfies the additive

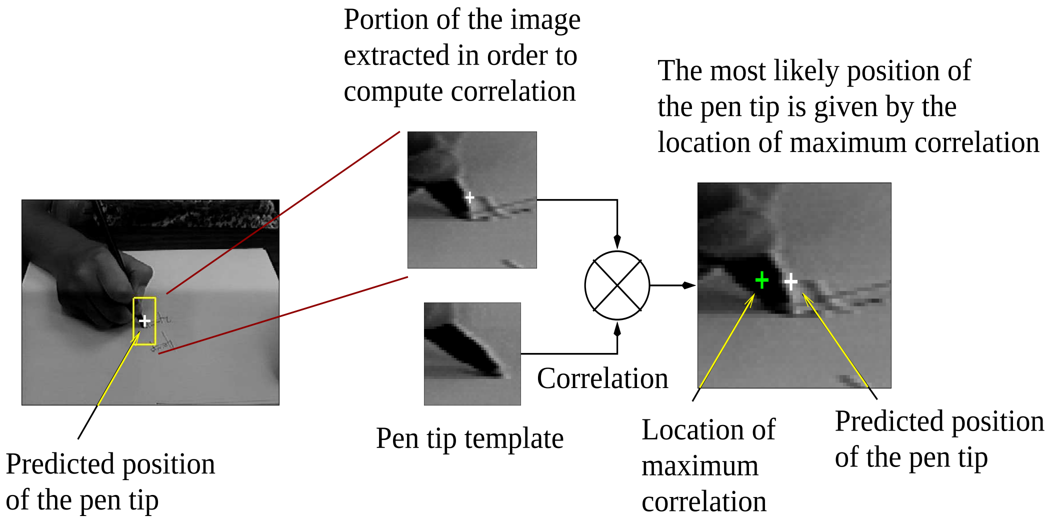


Figure 2.5: Given the predicted location of the pen tip in the current frame, the most likely position of the pen is obtained by finding the place that has maximum correlation with the previously stored template of the pen tip.

white noise assumptions of the matched filter; however, as a first approximation we will assume that the pen tip can be detected in each frame using the matched filter. The detection of the position of the pen tip is obtained in our system by locating the maximum of the normalized correlation between the pen tip template and an image neighborhood centered on the predicted position of the pen tip, as shown in figure 2.5.

The spatial resolution of the interface is defined by the localization accuracy during the computation of the maximum correlation. Sub-pixel resolution is achieved by fitting a paraboloid to the correlation surface in the neighborhood of the pixel with maximum correlation value.

The system also analyzes the values of maximum normalized correlation to detect whether the pen tip is not within the predicted image neighborhood. If the value of maximum correlation is lower than a threshold, the system emits an audible signal and continues to look for the pen tip in the same place, waiting for the user to realize that tracking has been lost and that the pen tip must be returned to the proper image neighborhood. The system waits for a few frames and if the pen tip does not return to sight, the tracking stops.

2.2.3 Filtering

The filter predicts the most likely position of the pen tip on the following frame based on the current predicted position, velocity and acceleration of the pen tip and on the measured location of the pen tip provided by the pen tip tracker. The use of the filter reduces computations since having a prediction of the position of the pen tip in the next frame allows us to decrease the size of the neighborhood used to calculate correlation. The measurements will be acquired faster and the measured trajectory will be smoothed by the noise rejection of the filter. A Kalman Filter [38, 8, 37, 30, 2] is a recursive estimation scheme that is suitable for this problem. We tested several different first- and second-order models for the movement of the pen tip on the image plane. The model that provided the best performance with the easiest tuning was a simple random walk model for the acceleration of the pen tip on the image plane. The model is given by equation 2.1.

$$\begin{cases} \dot{\mathbf{x}}(t) &= \mathbf{v}(t) \\ \dot{\mathbf{v}}(t) &= \mathbf{a}(t) \\ \dot{\mathbf{a}}(t) &= \mathbf{n}_a(t) \\ \mathbf{y}(t) &= \mathbf{x}(t) + \mathbf{n}_y(t) \end{cases} \tag{2.1}$$

where $\mathbf{x}(\mathrm{t})$, $\mathbf{v}(\mathrm{t})$ and $\mathbf{a}(\mathrm{t})$ are the two-dimensional-components of the position, velocity and acceleration of the tracked point, and $\mathbf{n}_a(t)$ and $\mathbf{n}_y(t)$ are additive zero-mean, Gaussian, white noise processes. The output of the model $\mathbf{y}(t)$ is the position of the pen tip corrupted by additive noise. Equation 2.1 describes a continuous model, but our system is discrete by nature since we perform measurements at discrete instants of time. The corresponding discrete equations for the model are as follows:

$$\begin{cases} \mathbf{x}(k+1) &= \mathbf{x}(k) + \mathbf{v}(k) + \frac{1}{2}\mathbf{a}(k) \\ \mathbf{v}(k+1) &= \mathbf{v}(k) + \mathbf{a}(k) \\ \mathbf{a}(k+1) &= \mathbf{a}(k) + \mathbf{n}_a(k) \\ \mathbf{y}(k) &= \mathbf{x}(k) + \mathbf{n}_y(k) \end{cases} \tag{2.2}$$

where we made a small abuse of notation by using the same name for the continuous and discrete noise processes (for more information on obtaining the discrete-time formulation for a continuous-time system, see chapter 3 of reference [30] or appendix C.13 of reference [2]). Calling A the transition matrix and C the output matrix of the system, the equations for the discrete model in matrix form are presented in equation 2.3 and the corresponding signal model is depicted by the block diagram of figure 2.6.

$$\begin{cases} \mathbf{X}(k+1) &= A\,\mathbf{X}(k) + \mathbf{n}_X(k) \\ \mathbf{y}(k) &= C\,\mathbf{X}(k) + \mathbf{n}_y(k) \end{cases} \tag{2.3}$$

where

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{v}(k) \\ \mathbf{a}(k) \end{bmatrix} \qquad \mathbf{n}_X(k) = \begin{bmatrix} 0 \\ 0 \\ \mathbf{n}_a(k) \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

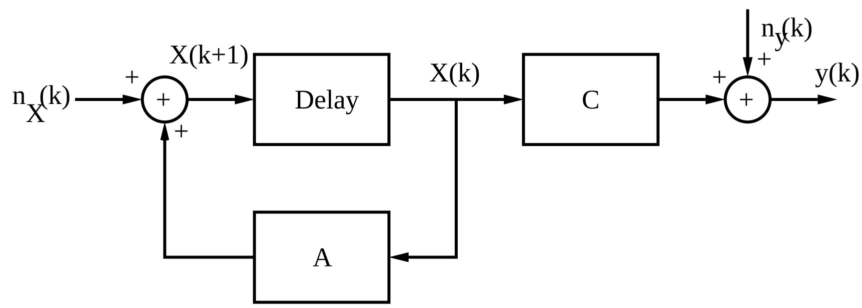


Figure 2.6: Signal model described by equation 2.3.

Before writing the equations for the recursive estimator, we must introduce some notation. Let us call $\mathbf{e}(k) = \mathbf{y}(k) - C\ \hat{\mathbf{X}}(k|k-1)$: the innovation of the filter, $R(k) = E\{\mathbf{n}_y(k)\mathbf{n}_y^T(k)\}$ the covariance matrix of the output noise, $Q(k) = E\{\mathbf{n}_X(k)\mathbf{n}_X^T(k)\}$ the covariance matrix of the state noise, $R_e(k) = E\{\mathbf{e}(k)\mathbf{e}^T(k)\}$ the covariance matrix of the innovation, $\hat{\mathbf{X}}(k|m) = E\{\mathbf{X}(k)|\mathbf{y}(0), \mathbf{y}(1), \cdots, \mathbf{y}(m)\}$ the estimate of the state $\mathbf{X}$ at time k using the measurements up to time m and $P(k|m) = E\{(\mathbf{X}(k) - \hat{\mathbf{X}}(k|m))(\mathbf{X}(k) - \hat{\mathbf{X}}(k|m))^T\}$ the corresponding covariance matrix of the estimation error. The equations for the Kalman filter are:

Measurement update equations:

$$\begin{cases} \hat{\mathbf{X}}(k|k) = \hat{\mathbf{X}}(k|k-1) + K_f(k)\ \mathbf{e}(k) \\ \mathbf{e}(k) = \mathbf{y}(k) - C\ \hat{\mathbf{X}}(k|k-1) \end{cases} \tag{2.4}$$

$$\begin{cases} R_e(k) = R(k) + C\ P(k|k-1)\ C^T \\ K_f(k) = P(k|k-1)\ C^T\ R_e^{-1}(k) \\ P(k|k) = P(k|k-1) - P(k|k-1)\ C^T\ R_e^{-1}(k)\ C\ P^T(k|k-1) \end{cases} \tag{2.5}$$

Time prediction equations:

$$\begin{cases} \hat{\mathbf{X}}(k+1|k) = A\ \hat{\mathbf{X}}(k|k) \\ P(k+1|k) = A\ P(k|k)\ A^T + Q(k) \end{cases} \tag{2.6}$$

Initialization:

$$\begin{cases} \hat{\mathbf{X}}(0|-1) &= \bar{\mathbf{X}}(0) \\ P(0|-1) &= P(0) \end{cases} \tag{2.7}$$

In the case of our system, both $R(k)$ and $Q(k)$ are assumed to be time-invariant and diagonal, i.e., the measurements are assumed to be uncorrelated and the components of the state are assumed to be uncorrelated. Given the model for the system shown in equations 2.2 and 2.3, the only components of the state with noise are the accelerations. Assuming that the vertical and horizontal accelerations of the pen tip are uncorrelated is a bit unreal since the muscular group that generates the handwritten pattern on the paper does not move the pen tip in x and y independently but rather as a whole. Also the assumption of uncorrelated measurements is unreal since we use a paraboloid fit of the correlation surface to obtain sub-pixel accuracy, and therefore, the horizontal and vertical coordinates of the maximum of correlation are linked through the fitting. However, the uncorrelated assumption provides a first order model of the actual dependences with a small number of parameters. In fact, R and Q provide the parameters that need to be tuned in order to achieve a particular convergence of the filter. We have four tuning parameters in these two matrices, two variances corresponding to $\mathbf{n}_y$ and two variances corresponding to $\mathbf{n}_a$.

2.2.4 Missing frames

The algorithm described in previous sections detects the position of the pen tip in each of the frames of the sequence. Unfortunately, some intermediate frames could be missing due to problems in the acquisition, or, in the case of the real-time implementation, due to problems with the synchronization of the PC with the frame grabber. Since it is desirable to sample the handwritten trajectory at a constant rate, there is a need for estimating the most likely position of the pen tip for the missing frames.

Various methods can be used to perform this estimation. Working in batch mode, the pen tip position for the missing frames could be estimated by spline interpolation of the acquired trajectory. Working on on-line mode, the position could be estimated by using the prediction of the Kalman filter. These two approaches represent the extreme cases of a range of possibilities that depend on the number of data points used to estimate the position of the pen tip for missing frames. One of the design premises of the interface is real-time operation; therefore, it is desirable to estimate the position of the pen tip for missing frames as the acquisition is taking place. The predicted position of the pen tip provided by the recursive estimator is the most likely location of the pen tip given the past history of the movement of the pen tip, i.e., the prediction summarizes all the available information on the state of the system up to the last measurement. Clearly this prediction does not include any information on future measurements, so a better estimation of the position of the pen tip for a missing frame would be obtained by merging the information given by the prediction and the information provided by future measurements. We estimate the position of the pen tip for the missing frames using the prediction given by the Kalman filter and the measurement obtained after the missing frame. This estimation can be performed as soon as the new measurement is captured since it only requires a few extra computations. The interface provides in this way a constant-rate stream of data that could be used by a handwriting recognition engine to perform recognition as the acquisition is taking place, increasing the possibility of interaction with the user.

Assuming that frame $k-1$ is missing, given that we have measurements at time $k-2$ and at time k, we need to estimate the most likely state and the output of the system at time $k-1$. The Kalman smoother [2, 30] is the proper scheme to solve this estimation problem. The Kalman smoother is derived from the Kalman filter by augmenting the signal model as shown in figure 2.7. The state equations for the augmented signal model are presented in equation 2.8. This new signal model generates estimates of the state that involve more measurement samples than in the conventional Kalman filter at the expense of an increase in the complexity of the estimator.

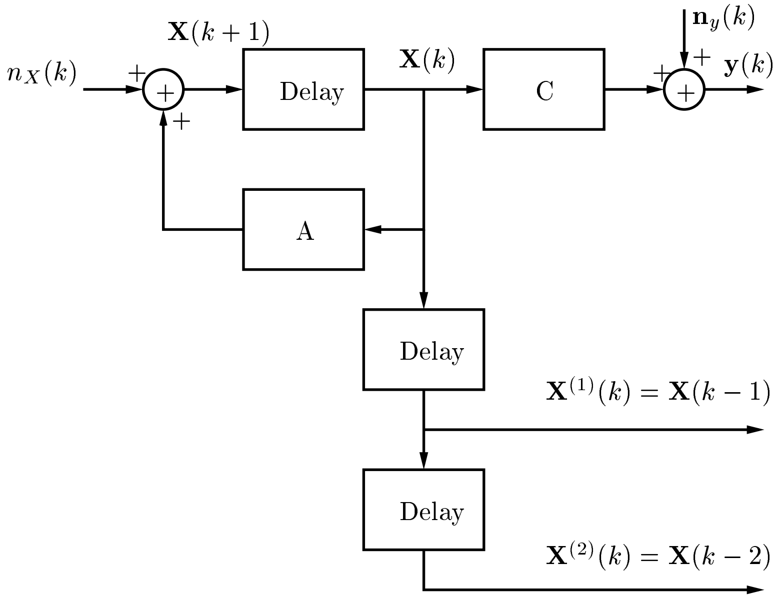


Figure 2.7: Augmented signal model (adapted from reference [2]).

$$\left\{\begin{array}{lcl}\begin{bmatrix}\mathbf{X}(k+1)\\ \mathbf{X}^{(1)}(k+1)\\ \mathbf{X}^{(2)}(k+1)\end{bmatrix} & = & \begin{bmatrix}A & 0 & 0\\ I & 0 & 0\\ 0 & I & 0\end{bmatrix}\begin{bmatrix}\mathbf{X}(k)\\ \mathbf{X}^{(1)}(k)\\ \mathbf{X}^{(2)}(k)\end{bmatrix} + \begin{bmatrix}\mathbf{n}_X(k)\\ 0\\ 0\end{bmatrix}\\ \\ \mathbf{y}(k) & = & \begin{bmatrix}C & 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{X}(k)\\ \mathbf{X}^{(1)}(k)\\ \mathbf{X}^{(2)}(k)\end{bmatrix} + \mathbf{n}_y(k)\end{array}\right. \tag{2.8}$$

From figure 2.7 and equation 2.8, it is clear that $\mathbf{X}^{(1)}(k+1) = \mathbf{X}(k)$ and $\mathbf{X}^{(2)}(k+1) = \mathbf{X}(k-1)$. Therefore, assuming that we have measurements $\{\mathbf{y}(0), \mathbf{y}(1), \cdots, \mathbf{y}(k)\}$,

$$\hat{\mathbf{X}}^{(1)}(k+1) = E\{\mathbf{X}^{(1)}(k+1)|\{\mathbf{y}(0), \mathbf{y}(1), \cdots, \mathbf{y}(k)\}\} = \hat{\mathbf{X}}(k|k)$$
$$\hat{\mathbf{X}}^{(2)}(k+1) = E\{\mathbf{X}^{(2)}(k+1)|\{\mathbf{y}(0), \mathbf{y}(1), \cdots, \mathbf{y}(k)\}\} = \hat{\mathbf{X}}(k-1|k)$$

where $\hat{\mathbf{X}}(k-1|k)$ is the estimate that we need, in the case in which $\mathbf{y}(k-1)$ is missing.

The increase in complexity of the estimator is reflected in the covariance matrix of the estimator error since there are new sub-matrices that represent the covariance between the estimation error at different points in time, as shown in the following equation:

$$\mathbf{P}(k+1|k) = E\left\{ \begin{bmatrix} (\mathbf{X}(k+1) - \hat{\mathbf{X}}(k+1|k)) \\ (\mathbf{X}^{(1)}(k+1) - \hat{\mathbf{X}}^{(1)}(k+1|k)) \\ (\mathbf{X}^{(2)}(k+1) - \hat{\mathbf{X}}^{(2)}(k+1|k)) \end{bmatrix} \begin{bmatrix} (\mathbf{X}(k+1) - \hat{\mathbf{X}}(k+1|k)) \\ (\mathbf{X}^{(1)}(k+1) - \hat{\mathbf{X}}^{(1)}(k+1|k)) \\ (\mathbf{X}^{(2)}(k+1) - \hat{\mathbf{X}}^{(2)}(k+1|k)) \end{bmatrix}^T \right\}$$

$$= \begin{bmatrix} P(k+1|k) & P^{(1)^T}(k+1|k) & P^{(2)^T}(k+1|k) \\ P^{(1)}(k+1|k) & P^{(1,1)}(k+1|k) & P^{(2,1)^T}(k+1|k) \\ P^{(2)}(k+1|k) & P^{(2,1)}(k+1|k) & P^{(2,2)}(k+1|k) \end{bmatrix}$$

where we have identified $P^{(i)}(k+1|k) = E\{(\mathbf{X}(k+1) - \hat{\mathbf{X}}(k+1|k))(\mathbf{X}^{(i)}(k+1) - \hat{\mathbf{X}}^{(i)}(k+1|k))^T\}$ and $P^{(i,i)}(k+1|k) = E\{(\mathbf{X}^{(i)}(k+1) - \hat{\mathbf{X}}^{(i)}(k+1|k))(\mathbf{X}^{(i)}(k+1) - \hat{\mathbf{X}}^{(i)}(k+1|k))^T\}$. Given that $\hat{\mathbf{X}}^{(i+1)}(k+1|k) = \hat{\mathbf{X}}(k-i|k)$, the diagonal terms of $\mathbf{P}(k+1|k)$ are $P^{(i+1,i+1)}(k+1|k) = P(k-i|k)$. Applying the Kalman filter equations to the augmented system and decoupling the different terms, we have:

$$\begin{cases} \hat{\mathbf{X}}(k+1|k) &= A\,\hat{\mathbf{X}}(k|k-1) + K_p(k)\,\mathbf{e}(k) \\ \hat{\mathbf{X}}^{(1)}(k+1|k) &= \hat{\mathbf{X}}(k|k-1) + K_p^{(1)}(k)\,\mathbf{e}(k) \\ \hat{\mathbf{X}}^{(2)}(k+1|k) &= \hat{\mathbf{X}}^{(1)}(k|k-1) + K_p^{(2)}(k)\,\mathbf{e}(k) \\ \mathbf{e}(k) &= \mathbf{y}(k) - C\,\hat{\mathbf{X}}(k|k-1) \end{cases} \tag{2.9}$$

$$\begin{cases} R_e(k) &= R(k) + C\,P(k|k-1)\,C^T \\ K_p(k) &= A\,P(k|k-1)\,C^T\,R_e^{-1}(k) \\ K_p^{(1)}(k) &= P(k|k-1)\,C^T\,R_e^{-1}(k) \\ K_p^{(2)}(k) &= P^{(1)}(k|k-1)\,C^T\,R_e^{-1}(k) \end{cases} \tag{2.10}$$

$$\left\{\begin{array}{rcl} P(k+1|k) & = & A\ P(k|k-1)\ A^T + Q(k) - K_p(k)\ R_e(k)\ K_p^T(k) \\ P^{(1)}(k+1|k) & = & P(k|k-1)\ (A - K_p(k)\ C)^T \\ P^{(2)}(k+1|k) & = & P^{(1)}(k|k-1)\ (A - K_p(k)\ C)^T \\ P^{(1,1)}(k+1|k) & = & P(k|k-1) - P(k|k-1)\ C^T\ {K_p^{(1)}}^T(k) \\ P^{(2,2)}(k+1|k) & = & P^{(1,1)}(k|k-1) - P^{(1)}(k|k-1)\ C^T\ {K_p^{(2)}}^T(k) \end{array}\right. \tag{2.11}$$

The equation that we are looking for is the one corresponding to $\hat{\mathbf{X}}(k-1|k) = \hat{\mathbf{X}}^{(2)}(k+1|k)$, given that we have no measurement at time $k-1$. From equation 2.10, it is clear that $\hat{\mathbf{X}}(k-1|k) = \hat{\mathbf{X}}(k-1|k-1) + K_p^{(2)}(k)\ (\mathbf{y}(k) - C\ \hat{\mathbf{X}}(k|k-1))$, but the problem is that we cannot get $\hat{\mathbf{X}}(k-1|k-1)$ since we do not have a measurement at $k-1$. Using the measurement update equation 2.4 for the Kalman filter, $\hat{\mathbf{X}}(k-1|k-1) = \hat{\mathbf{X}}(k-1|k-2) + P(k-1|k-2)\ C^T\ R_e^{-1}(k-1)(\mathbf{y}(k-1) - C\ \hat{\mathbf{X}}(k-1|k-2))$. One way to overcome this problem is by assuming that there is a measurement at $k-1$ that has infinite variance. Therefore, the innovation covariance $R_e(k-1)$ goes to infinity (or, better, its inverse $R_e^{-1}(k-1)$ goes to zero), and $\hat{\mathbf{X}}(k-1|k-1) = \hat{\mathbf{X}}(k-1|k-2)$. Also, $\hat{\mathbf{X}}(k|k-1) = A\ \hat{\mathbf{X}}(k-1|k-1) = A\ \hat{\mathbf{X}}(k-1|k-2)$. In fact, given that we do not have any measurement at $k-1$, the most likely state of the system at $k-1$ is provided by the prediction from time $k-2$ and the predicted state at time k is given by applying the transition matrix onto the most likely state at time $k-1$. Finally, the equations for computing $\hat{\mathbf{X}}(k-1|k)$ are as follows:

$$\left\{\begin{array}{rcl} \hat{\mathbf{X}}(k-1|k) & = & \hat{\mathbf{X}}(k-1|k-2) + K_p^{(2)}(k)(\mathbf{y}(k) - C\ A\ \hat{\mathbf{X}}(k-1|k-2)) \\ K_p^{(2)}(k) & = & P(k-1|k-2)(A - K_p(k-1)\ C)^T\ C^T\ (R(k) + C\ P(k|k-1)\ C^T)^{-1} \\ K_p(k-1) & = & A\ P(n-1|n-2)\ C^T\ (R(k-1) + C\ P(k-1|k-2)\ C^T)^{-1} \\ P(k|k-1) & = & A\ P(k-1|k-2)\ A^T + Q(k-1) - \\ & & K_p(k-1)\ (R(k-1) + C\ P(k-1|k-2)\ C^T)\ K_p^T(k-1) \end{array}\right. \tag{2.12}$$

After estimating the state of the system for the missing frame at time $k-1$, the measurement update step of the Kalman filter for the current frame k is performed using equations 2.4 and 2.5 with the only caveat that $\hat{\mathbf{X}}(k|k-1) = A\,\hat{\mathbf{X}}(k-1|k-2)$ and $P(k|k-1)$ is the one computed with equation 2.12.

2.2.5 Ballpoint detection

The previous block of the system finds the most likely position of the centroid of the pen tip, a point that will be close to the center of gravity of the triangular model of the pen tip. The position of the ballpoint is obtained using a similar algorithm as the one used in the initialization. The main difference from the initialization algorithm is that the pen is now moving, so we need to compute one ballpoint position for each frame and only a part of the steps of the initialization algorithm are used.

Using Canny's edge detector, we find the position and the orientation of the boundary edges of the pen tip, i.e., the sides of this triangle. Using the values of the orientations of the boundaries in the previous frame, the distance from the ballpoint and the finger to the centroid of the pen tip as well as the current position of the centroid, we calculate the expected current position of the boundaries of the pen. A few points on these boundaries (in the case of the experiments, we use five points) are chosen as the centers of the edge detection windows. The edge detector is only applied to small windows in order to save computations and to speed up the processing of the current frame. We look for points in each window that have maximum contrast. The edges are found by interpolating lines through these points. Then, the axis of the pen tip is computed as the mean line defined by the pen boundary edges. The profile of the image through the pen's axis is extracted in order to find the position of the writing end. The position of the finger along the axis of the pen tip is also extracted in order to complete the representation of the projection of the pen tip on the image plane.

Figure 2.8 shows the image of the pen tip, its boundary edges, its axis, the image profile, and the positions of the ballpoint and the finger. The positions of the ball-

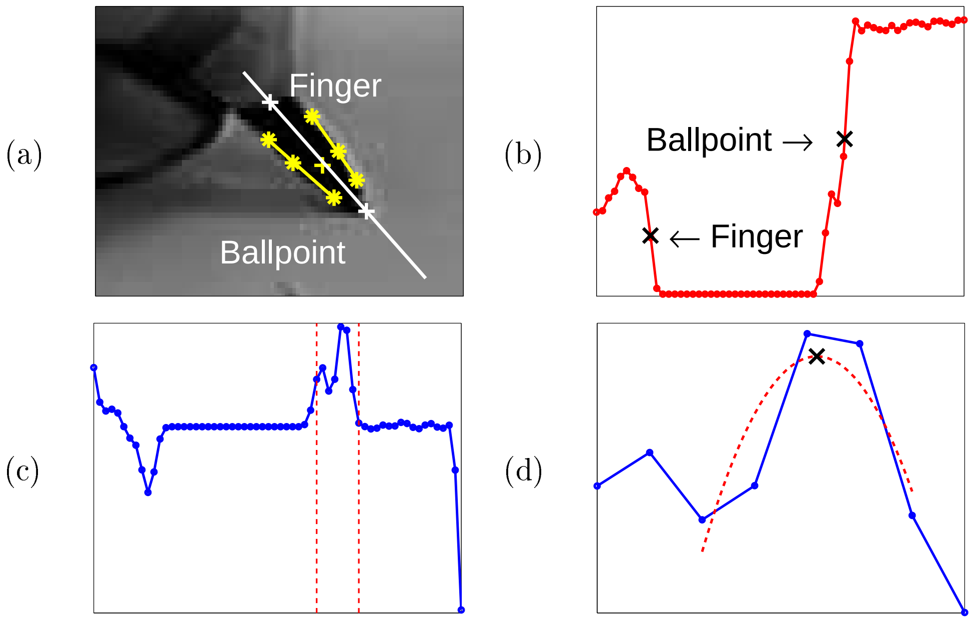


Figure 2.8: (a) Image of the pen tip displaying the various elements used to detect the ballpoint. The cross '+' in the center of the image shows the centroid of the pen tip obtained with correlation. The points marked with a star '*' show the places where the pen boundaries were found using edge detection. The lines on the sides of the pen tip are the boundary edges and the line in the middle is the pen tip axis. The other two crosses 'x' show the estimated positions of the ballpoint and of the finger. (b) Image profile along the axis of the pen tip and the corresponding positions of the ballpoint and of the finger. (c) Result of correlating the image profile with a derivative of a Gaussian function. (d) Blow-up of the region between the dotted vertical lines in (c). Parabolic fit of the peak identifies the position of the ballpoint. The vertex of the parabola is plotted with a cross 'x' and it corresponds to the estimated sub-pixel position of the ballpoint.

point and the finger are computed with sub-pixel accuracy by correlating the image profile with a derivative of a Gaussian function (1D edge detection) and then fitting a parabola to the corresponding peak values. Figure 2.8 (c) shows the result of correlating the image profile with the derivative of a Gaussian function and Figure 2.8 (d) displays the parabolic fit to the corresponding peak and the estimated position of the ballpoint.

2.2.6 Stopping acquisition

We have mentioned that the system automatically stops if the value of maximum correlation is very low, since this would imply that the pen tip has moved outside the search window (or that there was such a change in illumination that the pen tip no longer matches the template). The user can exploit this behavior to stop the acquisition by taking the pen tip away from the search window. There is another stopping possibility offered to the user. The system checks whether the pen tip has moved at all between consecutive frames and counts the number of consecutive frames in which there is no movement. If this number reaches a predefined threshold, the system stops the acquisition. Thus, if the user wants to finish the acquisition at the end of a desired piece of handwriting, he/she can hold the pen tip still and the system will stop the acquisition.

2.3 Pen up detection

The trajectories obtained by the tracking blocks of the system are not suitable for performing handwriting recognition using standard techniques since most of the recognition systems to date assume that their input is only formed by pen down strokes. Our interface has only one camera from which we cannot detect whether the pen is touching the paper or not, as mentioned in section 2.1. The addition of another camera that would form a stereo system with the first would allow one to compute the distance between the pen and the paper. However, a stereo system would bring more problems than benefits. We would need another camera, increasing the cost of the interface; we would need to process two images instead of one and also establish correspondence between them, doubling the required number of computations; finally, we would need to set up a calibration procedure to be performed each time that the interface was going to be used, imposing a further burden on the user. Hence, the detection of the times when the pen is lifted and, therefore, not writing, is accomplished in our system by using the additional information given by the ink path on

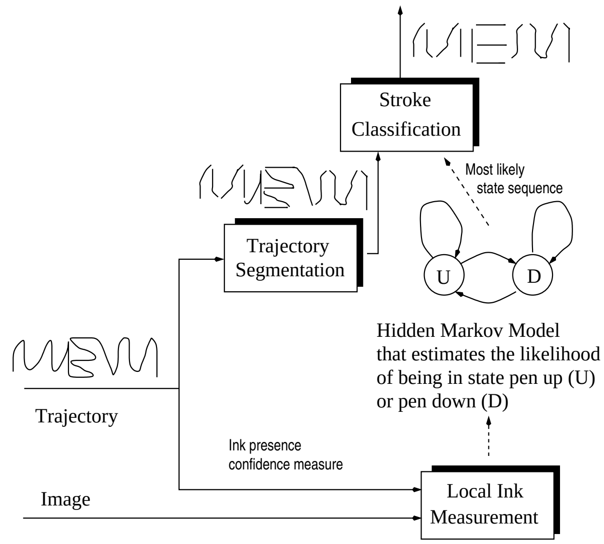


Figure 2.9: Block diagram of the pen up/down classification subsystem. We detect when the pen is up or down using a bottom-up approach. At the local level, the brightness of each point in the trajectory is measured and compared with the brightness in its surroundings. A Hidden Markov Model (HMM) estimates the likelihood of being at state Pen Up or state Pen Down for each individual point given the measured brightness. The full trajectory is segmented into strokes and each stroke is classified as pen up or pen down aggregating the likelihoods provided by the HMM.

the paper.

The system checks for the presence of ink in the places where the pen's ballpoint was found. The detection of ink using image brightness is tricky. The image brightness at any given place varies with illumination, the writer's hand position, and the camera gain. Moreover, the image contrast could change from frame to frame due to light flickering and shadows. So, the detection of ink must be done using local measurements for each point. Figure 2.9 shows a block diagram of this subsystem.

The existence of ink on the paper is checked locally for each ballpoint position. A measure of confidence of the presence of ink is obtained from local measurements of brightness. A Hidden Markov Model is used in order to model the transition of the confidence measure between the states of pen up and pen down. Using the local confidence measure and the estimated HMM state sequence, the system classifies each particular point of the trajectory as pen up or pen down. However, the measure of ink presence is quite difficult and prone to errors, so it is better to divide the complete handwritten trajectory into strokes and aggregate the point-wise classification into a stroke-wise classification. The following sections describe each of the blocks presented in figure 2.9 in more detail.

2.3.1 Local ink detection

Given a particular position of the ballpoint, we need to find out whether there is an ink trace on the paper at this place or not. The local detection of the ink trace on the paper is quite difficult, as we illustrate with the example of figure 2.10. The first plot of the figure shows the recovered sequence, the second plot displays an image of the ink trace left on the paper and the third plot presents the overlay of the sequence onto the image of the ink trace, where the little dots represent the sample points. The image of the ink trace is a portion of the last frame of the sequence (we can see part of the pen tip on the left side of the image). The ink trace appears upside-down on the image since the camera is aimed to the hand of the writer from the front side (then, the camera sees the writing being done from right to left and from bottom to top). We note that the ink trace takes only a few pixels of the whole image and that there is quite a distortion on the strokes due to the pixelization of the image. We also observe that the overlaid sequence fits quite well to the ink trace except at the beginning of the sequence (shown on the right side of the image). This happens because there might have been a displacement of the paper generated by one of the strokes (probably the long horizontal stroke between samples 20 and 40). We measure the brightness profile of the image along a line perpendicular to the direction of movement that passes through each sample point. The values of brightness measured are presented as an image in the last plot of figure 2.10. The profile is measured at the position of the ballpoint and on five pixels on each side of the ballpoint, along the mentioned perpendicular. We note that the ink trace is not always found at the ballpoint position (pixel number 6 on the vertical axes). We can see the ink trace being a few pixels off the ballpoint pixel at the beginning of the sequence (samples 1-20), then stabilizing on the ballpoint (samples 20-35) until the pen tip appears on the profile (samples 35-40) and later disappearing because of a pen up stroke (samples 40-55).

We can get several observations from this simple example. The local ink measurement should be performed as soon as possible in order to have a good fit of the sample points on top of the ink trace. However, the measurement has to be done after the pen tip moves away, otherwise, the pen tip will obstruct the paper and the ink trace. The value of brightness corresponding to the ink trace varies quite a lot within the same image (and it is expected to vary even more across images), so we need to detect the ink trace in a robust and local way.

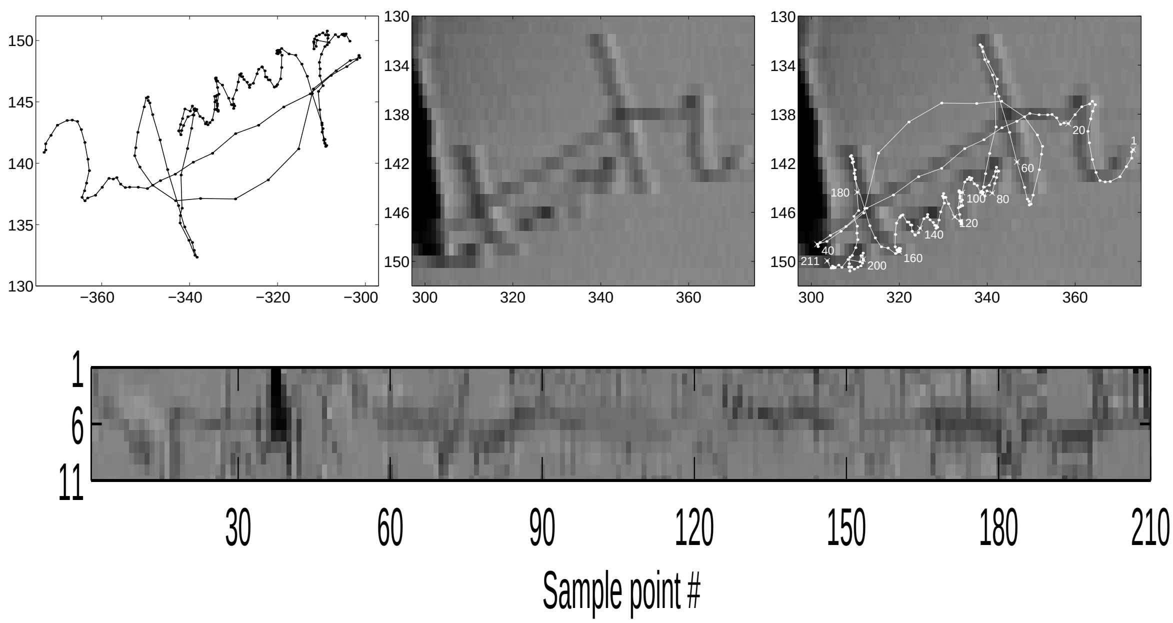


Figure 2.10: Example that illustrates the difficulties in detecting the ink trace. The first plot shows one sequence acquired with the visual tracker. The dots indicate the position of the sample points. The second plot displays a portion of the last frame of the sequence showing the corresponding ink trace deposited on the paper. The third plot shows the recovered sequence overlaid on the image of the ink trace. The sample points land over the ink trace most of the time, and the exception is at the beginning of the sequence (shown on the right side of the image). The last plot shows the profile of the image brightness along lines that passes through each sample point and are perpendicular to the direction of motion. The profile of image brightness is measured at eleven sample points using the interpolation method described in [42]. The profile is shifted so that the detected ballpoint position appears in sample 6 (row 6 of the plot).

Figure 2.11 shows how the local ink detection is performed. The center cross of figure 2.11(a) shows the position of the ballpoint. We observe that there is an ink trace at the ballpoint's position. The ink trace is not straight due to the pixelization effect produced by the digitalization of the image. The local detection of the ink trace is quite difficult since the ink trace is only 1 to 3 pixels wide; therefore, if the position of the ballpoint is off by 1 pixel, the local detection would be wrong. We perform the detection locally by comparing the brightness of the pixel where the ballpoint was found with the brightness of the pixels on a surrounding circle centered at the ballpoint position. This circle is shown in figure 2.11(a). The brightness at each position marked with a plus '+' is obtained by interpolation using the method described in [42]. Figure 2.11(b) shows the histogram of the brightness values measured on the circle and the brightness measured at the ballpoint's position (vertical line).

We assume the brightness of ink-less pixels to be a Gaussian-distributed random variable. We estimate the mean and variance of this probability density function using the brightness values measured on the circle, assuming that all these positions correspond to ink-less pixels. The ink presence confidence measure is computed as the probability of the brightness found at the ballpoint given the paper without ink (this probability correspond to the area below the Gaussian pdf between $-\infty$ and the brightness value found at the ballpoint). If there is ink present at the ballpoint pixel, this measure is low, close to zero. Otherwise, the measure is high, close to one. The selection of this particular confidence measure is very convenient since it provides automatic scaling between zero and one.

The brightness measurements cannot be obtained until the pen tip has left the measurement area; otherwise, the ink trace will be covered by the pen tip and/or the hand of the user. The system assumes a simple cone-shaped model for the area of the image covered by the pen and the hand of the user. The ballpoint is located at the vertex of the cone, the axis of the pen tip defines the axis of the cone, the position of the finger has to be inside the cone, and the aperture of the cone is chosen to be 90 degrees. This simple model allows the system to determine if the user is left handed or right handed and whether a particular ballpoint position is within the cone. The

(a) (b)

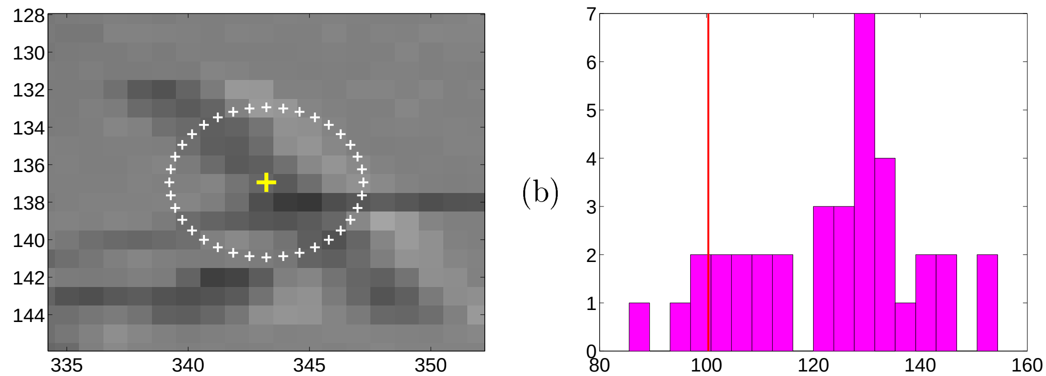


Figure 2.11: (a)The center cross corresponds to the position of the pen's ballpoint. The other points show the surrounding pixels where the brightness is measured. (b) Histogram of the measured brightness. The vertical line shows the value of brightness corresponding to the ballpoint's position.

system waits until the cone is sufficiently far away from the area of interest before doing any brightness measurements.

2.3.2 Local pen up/down modeling

The ink presence confidence measure could be used to decide whether a particular sample point corresponds to pen up or pen down. However, making hard decisions based on a single measurement is very likely to fail due to noise and errors in brightness measurements. A soft-decision approach that estimates the probability of each individual point being a pen up or a pen down is more robust. A further improvement is provided by modeling the probability of transition between these two states (pen up or pen down), given the current measurement and the previous state. A Hidden Markov Model (HMM) with two states, one corresponding to pen up and the other corresponding to pen down, is a suitable scheme to estimate these probabilities. The HMM *learns* the probabilities of moving from one state to the other and the probabilities of rendering a particular value of confidence from a set of examples.

Hidden Markov Models

The HMM models a doubly stochastic process governed by an underlying Markov chain with a finite number of states and a set of random functions each of which is associate with one state. At discrete instants of time, the process is in one of the states and generates an observation symbol according to the random function corresponding to the current state. The model is hidden in the sense that all that can be seen is a sequence of observations. The underlying state which generates the symbol is hidden.

In principle, the underlying Markov chain may be of any order and the observations may be multivariate random processes having some continuous probability density function. In this application, however, the Markov chain is restricted to be of order one, i.e., those for which the probability of transition to any state depends upon that state and its predecessor. Also, the observation sequence is limited to be drawn according to probability distribution functions associated with the states.

The HMM may have different structure, i.e., different topology of interconnection of nodes. Depending on the application, it is possible to constrain an HMM such that only certain desired state transitions are allowed. Figure 2.12 shows several possible topologies. For applications in speech recognition and handwriting recognition, left-to-right HMMs give the best results since they encode in the topology the causality of phoneme or letter generation. For our present application, a generalized model with two states is the most suited.

Elements of an HMM An HMM is characterized by the following elements (with the same notation used by Rabiner [63, 62]):

N : *the number of states of the model.* Although the states are hidden, for many practical applications there is some physical significance attached to the states or to sets of states of the model. We denote the individual states as $S = \{S_1, S_2, ..., S_N\}$ and the state at time t as q_t.

M : *the number of distinct observation symbols per state*, i.e., the discrete alphabet size. The observation symbols correspond to the physical output of the systems

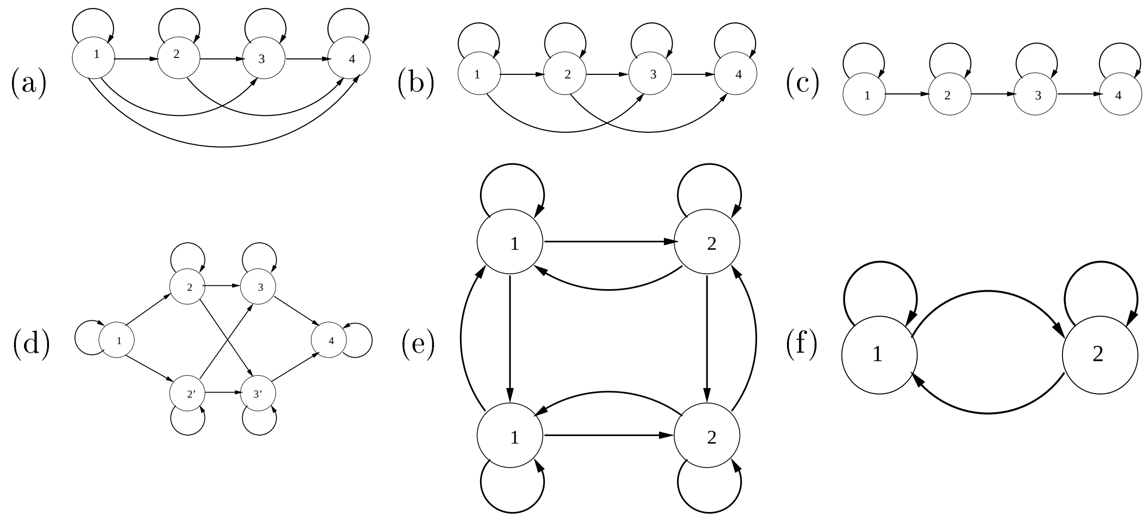


Figure 2.12: Different HMM topologies (a) Left-to-right-1, (b) Left-to-right-2, (c) Left-to-right-3, (d) parallel, (e) generalized, (f) generalized with states used in our system.

being modeled. We denote the individual symbols as $V = \{v_1, v_2, ..., v_M\}$.

A : *the state transition probability distribution*, $A = \{a_{ij}\}$

Where $a_{ij} = P(q_{t+1} = S_i \mid q_t = S_j)$, $1 \leq i, j \leq N$. For the special case where any state can reach any other state in a single step, we have $a_{ij} > 0$ for all i, j. For other types of HMMs, we would have $a_{ij} = 0$ for one or more (i, j) pairs. For example, for the left-to-right models, $a_{ij} = 0$ for $i > j$, i.e, A will be an upper triangular matrix.

B : *the observation symbol probability distribution in state* j, $B = \{b_j(k)\}$

Where $b_j(k) = P(v_k \text{ } at \text{ } t \mid q_t = S_j)$, $1 \leq j \leq N$, $1 \leq k \leq M$.

Π : *the initial state probability distribution*, $\pi = \{\pi_i\}$

Where $\pi_i = P(q_1 = S_i)$, $1 \leq i \leq N$.

A complete specification of an HMM requires the definition of two model parameters (N and M), the observation symbols, and three probability measures A, B and π. For convenience, we use the compact notation $\lambda = (A, B, \pi)$ to indicate the complete parameter set of the model since N and M are embedded in the sizes of A, B and π.

There are three common scenarios in which HMM's are useful in real-world applications:

1 : Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

This scenario corresponds to the evaluation problem, namely given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model. This problem is sometimes called the "classification" problem.

2 : Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model $\lambda = (A, B, \pi)$, how do we choose a corresponding state sequence $Q = q_1 q_2 \cdots q_T$ which is optimal in some meaningful sense (i.e., best "explains" the observations)?

This scenario is the one in which we attempt to uncover the hidden part of the model, i.e., to find the correct state sequence in some meaningful sense.

3 : How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

This scenario is the one in which we attempt to optimize the model parameters in order to best describe the generation of a given observation sequence. The observation sequence used to adjust the model parameters is called a training sequence since it is used to *train* the HMM. The training problem is the crucial one for most applications of HMMs, since it allows us to optimally adapt model parameters to observed training data, i.e., to create best models for real phenomena.

There are standard procedures described in the literature in order to solve each of the mentioned problems, namely the forward-backwards algorithm [63, 62] for scenarios 1 and 3 and Viterbi's algorithm [26, 63, 62] for scenario 2.

The HMM used in our system has the topology presented in figure 2.12(f). The observation of the model is the ink presence confidence measure, an intrinsically continuous variable as it was defined in section 2.3.1. The HMM output is a set of discrete symbols, so we need to quantize the value of the confidence measure in order to define the output symbols. The confidence measure is a probability, so it is scaled between zero and one. The interval $[0, 1]$ is divided into sixteen equal intervals in order to quantize each confidence measure value and to translate it into observation symbols. We use the forward-backward algorithm in order to train the HMM using a training set of handwritten sequences collected with the system. The training set consists of examples of cursive handwriting, block letters, numbers, drawings, signatures and mathematical formulas in order to sample the variability of the pen up/down transition for different types of writing. Figure 2.13 shows the resulting HMM after training, where the bar plots displays the output probability distribution of each state. The most likely state of the system at each point in the trajectory is estimated using Viterbi's algorithm [26, 62].

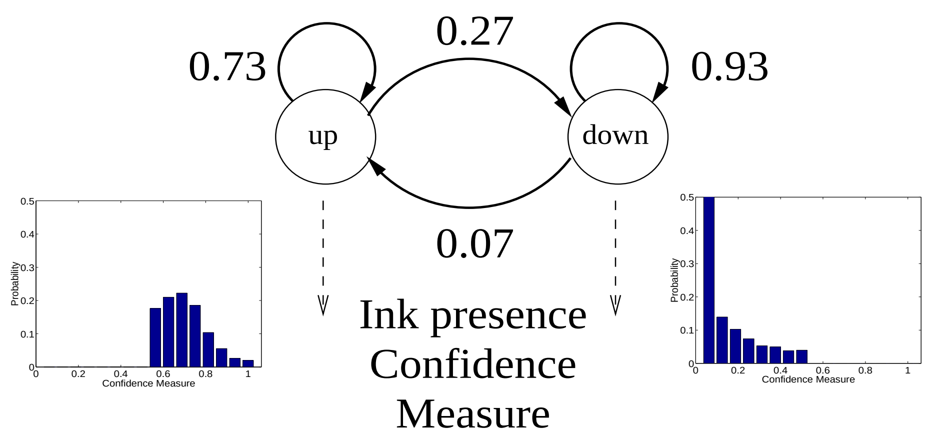


Figure 2.13: Resulting HMM that models the transitions between pen up and pen down states and the ink presence confidence measure.

2.3.3 Trajectory segmentation

The previous two sections describe local measures used to classify each *sample* of the handwritten trajectory as either pen up or pen down. The measurement of ink presence is subject to errors, so the performance may be improved by dividing the handwritten trajectory into different *strokes* and aggregating the sample-wise classification into a stroke-wise classification.

The handwritten trajectory is segmented into strokes using two features, the curvilinear velocity of the pen tip and the curvature of the trajectory. Selection of these two features was inspired by the work of Viviani [76, 77] and Plamondon [57, 59] and by the intuitive idea that on the limit points between two different handwriting strokes the velocity of the pen is very small and/or the curvature of the trajectory is very high. The set of segmentation points is the result of applying a threshold on each of the mentioned features. Figure 2.14 shows several examples of trajectories and the corresponding strokes after segmentation. The threshold in curvilinear velocity was chosen to be 0.75 pixel, so points that remain within the same pixel in two consecutive frames are discarded. The threshold in curvature was heuristically chosen to be 0.05.

2.3.4 Stroke classification

Having divided the trajectory into strokes, we proceed to classify the strokes as either up or down. We have experimented with two approaches, one based on the ink presence confidence measure and the other using the state sequence provided by the HMM. In the first approach, the mean of the ink presence confidence measures for each sample in the stroke is used as the stroke confidence measure. In the second approach, a voting scheme assesses the likelihood of a particular stroke being a pen up or pen down, this likelihood provides the stroke confidence measure. If needed, hard classification of each stroke as pen up or pen down can be obtained by applying a threshold on the stroke confidence results. The hard classification, as well as the likelihood of pen up/down, are the stroke descriptors that our interface provides to a handwriting recognition system.

2.3.5 Real-time implementation

Figure 2.15 shows a block diagram of the hardware used for implementing the system. The hardware consists of a video camera, a frame grabber, and a Pentium II 230MHz PC. The camera is a commercial Flexcam ID, manufactured by Videolabs,

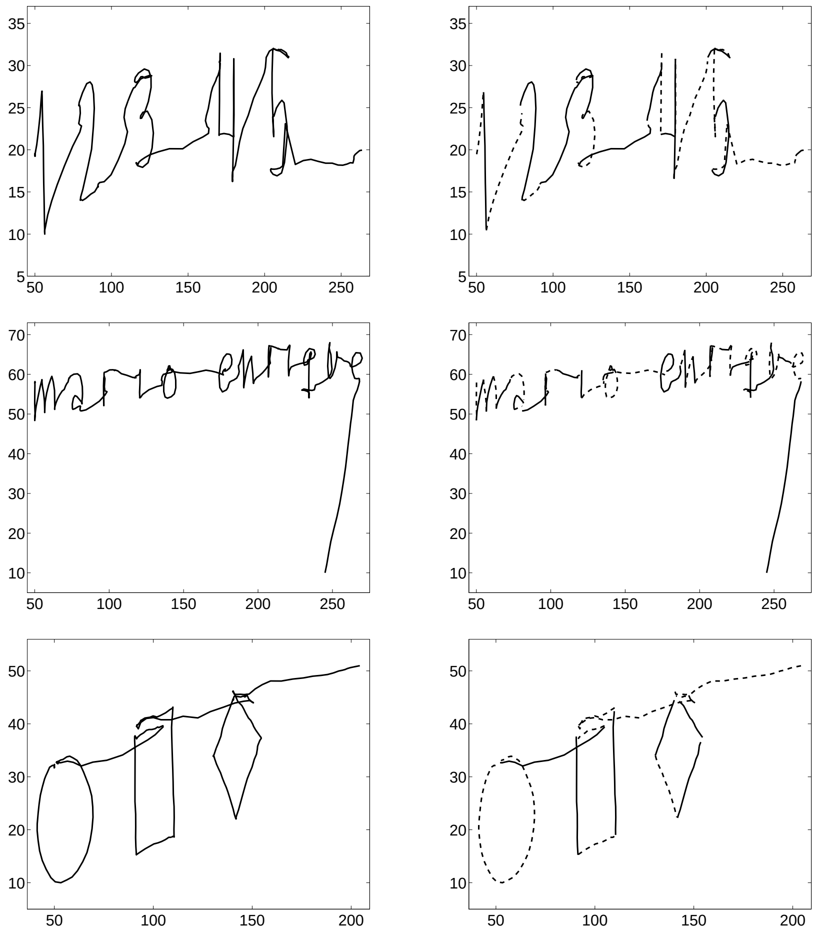


Figure 2.14: Several examples of trajectories acquired with the interface and the corresponding strokes obtained after segmentation. Successive strokes are indicated alternately with solid and dashed lines.

equipped with manual gain control. It has a resolution of 480x640 pixels per interlaced image. The frame grabber is a PXC200 manufactured by Imagination. The input camera image is digitized by the board and even and odd fields of the image are separated for future processing at 60 Hz and transferred to memory through the PCI bus. The frame grabber uses a double buffer scheme that allows the computer to process one frame while a new one is being acquired. All further computations are performed with the PC. Most of the modules of the system have been implemented in the real-time application with the exception of the pen up/down classifier that only performs sample-wise classification using local measurements instead of stroke-wise classification (see 2.3).

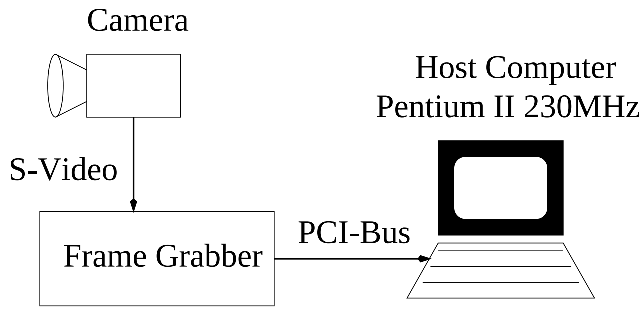


Figure 2.15: System configuration: The hardware architecture comprises a commercial camera, a frame grabber, and a Pentium II 230MHz.

Table 2.1 shows the computation time required on average by each of the modules of the system. The total processing time is 12.1 ms per frame. We observe that half of this processing time is used for displaying the image on the screen to provide visual feedback to the user. The other half of this processing time is used for actual computations. Both the transference of the new frame from the frame grabber to the memory of the PC, and the transference of the display image from the memory of the PC to the video card are performed through the PCI bus. Hence, there is a collision between two processes at the PCI bus that is the reason for the apparently important display time.

Module	**Time** (msec.)
Frame acquisition	16.67
Processing time	6.1
Pen tip tracking with correlation	3.5
Filter	0.3
Ballpoint detection & ink meas.	2.1
Additional checks in the program	0.2
Image inversion and display	6
Total computing time	12.1

Table 2.1: Computation time of each module of the system.

In figure 2.16 we show the graphical user interface (GUI) of the windows-based application that runs our program. We have three windows: one is a dialog box that allows the user to communicate with the application, the second displays the image captured by the camera in order to provide visual feedback to the user, and the third shows the acquired trajectory after having done sample-wise pen up/down classification with a hard threshold.

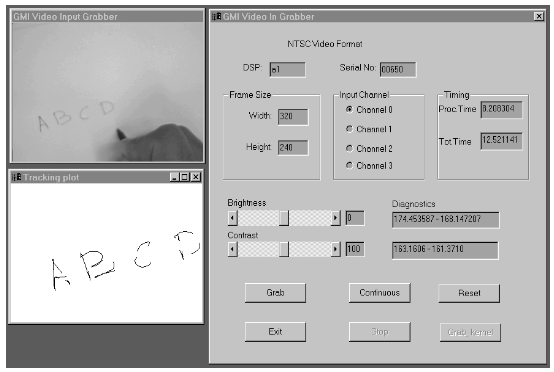


Figure 2.16: This image shows the GUI (Graphical User Interface) of the windows-based application that implements our system. The biggest window is a Dialog Box that allows the user to input parameters and run commands. A second window is used to show the image that the camera is providing to the system and the last window shows the output trajectory after having done the pen up/down classification.

2.4 Experimental results

2.4.1 System specifications

As a part of the specification of the interface, temporal and spatial resolution are key parameters that define the performance of the system. As we have mentioned before, the maximum working frequency provided by the camera is 60 Hz, so the temporal resolution of the system is at most 16.67ms. Since the total processing time per frame is 14ms, the system is able to work at maximum frame rate. However, there are frames that are missed due to a lack of synchronization between the CPU and the frame grabber. There is a block in the system that estimates the most likely state of the system in the case of missing frames. This scheme is useful if the number of missing frames is small, otherwise, the system would drift according to the dynamics of the model of equation 2.1. We have used the system for acquiring hundreds of handwritten sequences in real time, experiencing a missing frame rate of at most 1 out of every 200 frames.

The experiments of chapters 3 and 4 are based on signatures acquired in real time with our system. Signatures are written at higher speeds than normal handwriting and, therefore, a bigger image neighborhood has to be searched in order to find the pen tip. We acquired signature sequences by enlarging the search area and turning off the pen up detection block of the system. In these experiments, we experienced a missing frame rate of at most 1 out of every 400 frames. We observe that the system occasionally loses track of the pen tip when the subject produces an extremely fast stroke. This problem of losing track of the pen tip could be solved in the future by using a more powerful machine or dedicated hardware (able to process a larger search area). Nevertheless, after a few trials, the user learns how to utilize the system without exceeding its limits.

In order to evaluate the spatial resolution of the system, we performed two simple experiments. In the first experiment, we acquired a few sequences in which the pen tip was fixed at the same location, so any differences in the acquired positions were due to noise on the image acquisition and on the computation of correlation. We repeated this experiment ten times placing the pen at different positions and using different illumination. The static resolution of the system was estimated by computing the average standard deviation of the positions acquired in each of the sequences.

In the second experiment, we acquired ten sequences of a subject drawing lines of different orientations with the help of a ruler. The line were carefully drawn to be straight, so any differences from a straight line would be due to noise in the image acquisition and on the computation of correlation. We fit a line through the acquired points and computed the distance between these points and the fitting line. The dynamic resolution of the system is estimated by computing the average standard deviation of the mentioned distance in each of the sequences. Figure 2.17 shows examples of one sequence used to compute the static resolution and one sequence used to compute the dynamic resolution. Table 2.2 shows the resulting estimated resolution of the system.

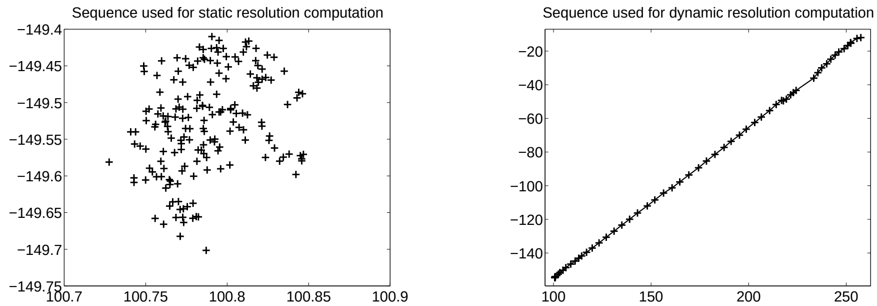


Figure 2.17: Examples of sequences used to estimate the static and dynamic resolution of the system.

	horiz. resolution (pixels)	vert. resolution (pixels)
static	0.0195	0.0627
dynamic	0.0423	0.0611

Table 2.2: Static and dynamic resolution of the system.

We note that the vertical resolution is almost the same for the two experiments but the horizontal resolution varies by a factor of two from one experiment to the other. This difference is possibly due to the subject holding the pen mostly in a vertical writing position for the static resolution experiment. In any case, we observe that the system has quite a good resolution of roughly one-twentieth of a pixel. Table 2.3 summarizes all the parameters used in the implementation of the real-time system.

Figure 2.18 shows several examples of complete handwritten sequences acquired in real time with our system, and figure 2.19 displays a more detailed part of some of these sequences. We collected examples of cursive handwriting, block letter, printed letters, drawings, signatures, and mathematical formulas. Comparing the automatically acquired examples presented in figure 2.18 with the manually obtained ones shown in figure 2.1, we observe that the level of noise in the automatically acquired examples is lower than in the manually acquired ones.

Parameter	Value
Pen tip template size	25x25 pixels
Correlation window size	15x15 pixels
Output noise covariance matrix (R)	diag(10^{-4}, 10^{-4})
State noise covariance matrix (Q)	diag(0,0,0,0,10^{-4}, 10^{-4})
Initial estimation error covariance matrix (P_0)	diag(1,1,10^{-2},10^{-2},10^{-5},10^{-5})
Initial dead time (given to the user to move the paper and find a clean area where to write)	2 sec. (120 frames)
Image difference threshold	15 (3 bits of noise)
Number of pixels required to detect movement	20
Number of pixels required to detect lack of movement	30
Time of no pen tip movement waited before acquring pen tip information	200 ms
Time used to acquire info. on the pen tip	1 sec. (30 frames)
Contrast threshold (used with Canny's edge detector)	0.7
Distance from parabolic cylinder axis to center of pixel threshold (used with Canny's edge detector)	0.5
Maximum correlation value considered as a match	0.75
Maximum velocity denoting pen not moving	0.5 pixels
Time waited before stopping	0.5 sec (30 frames)
Minimum number of points in a sequence	150 samples

Table 2.3: System parameters used in the real-time implementation.

2.4.2 Pen up detection experiments

Only the pen tracking and the local ink detection module's of the system have been implemented in the real-time application. In order to evaluate the performance of the complete pen up detection subsystem, we collected 20 sequences comprising various types of handwriting (cursive, block letters, printed letters, numbers, drawings, signatures, and mathematical formulas). We used half of these sequences for training the HMM and the other half for testing. We obtained ground truth by classifying by hand each of the points of the test sequences as a pen up or pen down. We also classified by hand each of the segments in which the test sequences were divided by the segmentation algorithm. Two types of error measurements are used to evaluate

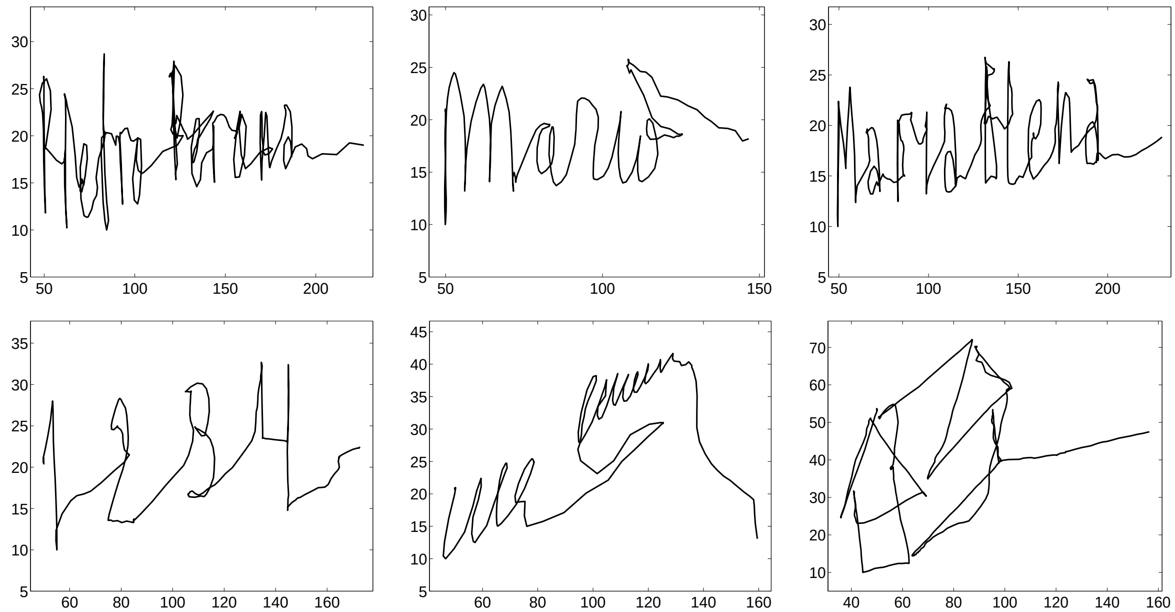


Figure 2.18: Examples of sequences captured with the real-time system. We collected examples of cursive writing, block letters, printed letters, drawings and mathematical symbols.

the performance, one is the false acceptance rate (FAR) which measures the percentage of pen up points (segments) that were classified as pen down by the system, and the false rejection rate (FRR) which provides the percentage of pen down points (segments) that were classified as pen up by our system. Figure 2.18 shows examples of sequences that we used for training the HMM.

Point-wise classification results

In this experiment, we compare the performance of point-wise pen up detection obtained by using local ink confidence and by using the HMM. We compute the FAR and FRR for the case in which the local ink confidence measurement is used to classify each point as pen up or pen down. A sample point is classified as pen down if the confidence measure is lower than 0.5; otherwise, it is classified as pen up. We also compute the FAR and FRR for the case in which each point is classified as pen up or pen down depending on the most likely sequence of states obtained with Viterbi's algorithm. Table 2.4 shows the error rates.

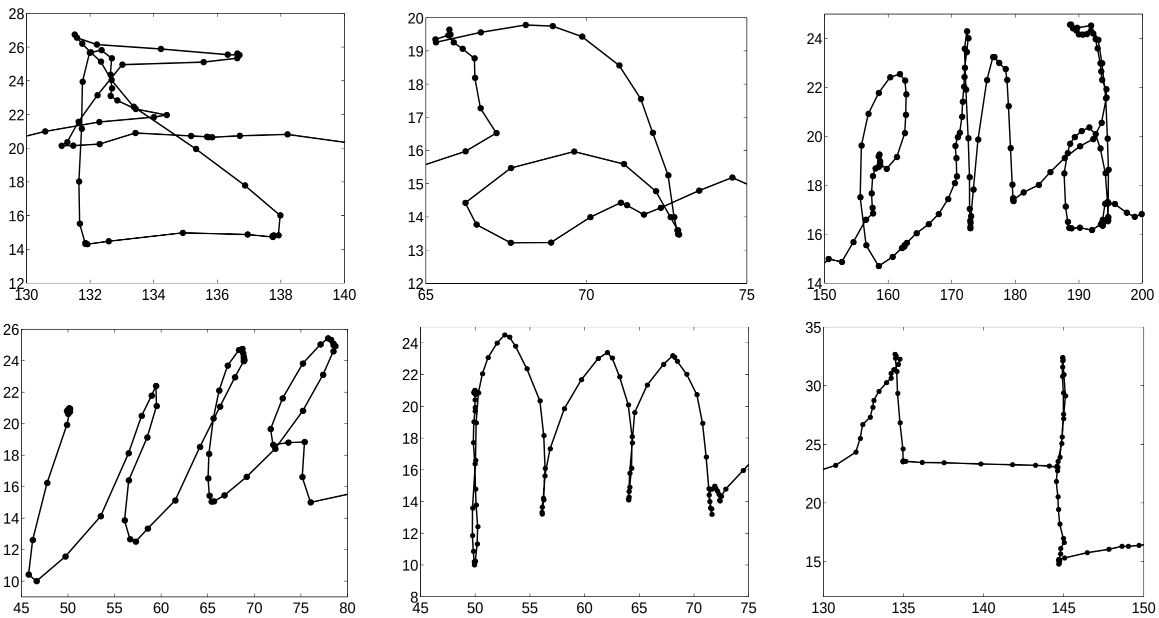


Figure 2.19: Portions of the example sequences shown in figure 2.18. The dots represent the actual samples acquired with the interface.

	local measurements (%)	HMM modeling (%)
FAR	28.6	28.6
FRR	5.33	5.33

Table 2.4: Comparison of the error rates of point-wise ink detection obtained using the local measurements and the HMM model.

Figure 2.20 depicts the application of these two approaches to three test sequences. The original sequences are plotted on the first row of the figure. The thickness of each segment in the sequences displayed on the second row is proportional to the mean of the confidence measure at the segment endpoints. On the third row, a segment of a sequence is plotted if the confidence at both endpoints is bigger than the threshold of 0.5. We see that there are several segments that appear in areas where there should be no ink trace on the paper. This misclassification is due to a bad measurement of the confidence of ink presence. On the plots on the last row, a segment is plotted if both extrema are classified as pen down by the HMM. We observe no difference in the error rates obtained with these two approaches, indicating that presumably there is no gain in using the HMM. As we pointed out before, we have to wait until the pen tip is out of sight in order to measure brightness. So, many "pen up" points that correspond to a stroke that passes above a segment of ink trace were misclassified as "pen down." This is the main reason for the apparently big value of the FAR.

Stroke classification results

In this section we evaluate the performance of pen up stroke classification. We compute the FAR and FRR for the case in which we aggregate the local ink confidence measurement in order to classify each stroke as pen up or pen down. A stroke is classified as "stroke down" if the stroke confidence measure is smaller than 0.25, otherwise, it is classified as "stroke up." We also compute the FAR and FRR for the case in which each stroke is classified as "stroke up" or "stroke down" using a voting scheme based on the most likely sequence of states of the HMM obtained with Viterbi's algorithm. A stroke is classified as "stroke down" if the resulting vote is bigger than 0.8; otherwise, it is classified as "stroke up." Table 2.5 shows the error rates.

	local measurements (%)	HMM modeling (%)
FAR	13.17	11.22
FRR	11.82	8.82

Table 2.5: Comparison of the error rates of stroke classification obtained using the ink presence confidence measure and the HMM model.

Figure 2.21 shows the results of these two approaches on the same test sequences that were classified point-wise. The segmented sequences are shown in the first row of the figure. The thickness of the strokes plotted on the second row is proportional to the stroke confidence measure. The third row shows the classification of the strokes as either stroke up or down using the previously mentioned threshold. The thickness of the strokes plotted on the fourth row is proportional to the voting results obtained with the HMM. The last row shows the hard classification of the resulting votes. We observe that the use of the HMM improves the performance of stroke classification even though it seemed to make no difference at point-wise level. We note that in most of the cases in which the stroke-up classification fails, it is due to an incorrect segmentation, like the "C" in the sequence "PEDRO MUNICH" or the crossing stroke of the "x" in the mathematical formula.

2.4.3 Discussion

This chapter has presented the design and implementation of a novel human-computer interface for handwriting. A camera is focused on the user's hand while he/she is writing with a normal pen on a piece of paper. We have shown that the handwriting trajectory is successfully recovered from its spatio-temporal representation given by the sequence of images. This trajectory is composed by handwritten strokes and pen movements between two strokes. The detection of the points in which the pen is traveling over the paper and not writing is obtained by using local measurements of the brightness of the image at the location in which the writing end of the pen was detected.

Several modules of the interface are susceptible of improvement. We used only one pen tip template for the whole sequence acquisition. This template could be automatically updated once the peak value of correlation fell below a certain threshold. Since the information about the boundaries of the pen tip, its axis, as well as the position of the ballpoint and the finger are computed for each frame by the ballpoint detection module, the automatic extraction of a new pen tip template involves no extra computational cost.

The region of interest used to detect the location of the pen tip has constant size in the current implementation of the system. The size of this region could be driven by the uncertainty on the predicted position of the pen tip, i.e., the size could depend on the covariance of the predicted location of the pen tip. Smaller regions would be required in cases of low uncertainty, reducing in this way the computational cost of performing correlation between the region of interest and the pen tip template.

The ballpoint detection is performed using the values of orientation of the pen tip and its boundaries obtained in the previous frame. We could improve the robustness of this detection by modeling the change of these orientations from frame to frame. A recursive estimation scheme could be used to predict the desired orientations, allowing to reduce the size of the windows used to perform edge detection and to decrease the number of computations.

We used a Gaussian model for the brightness of ink-less pixels. The estimation of the model parameters was performed using the brightness of points lying on a circle centered at the ballpoint position, assuming that all the circle points are ink-less points. Clearly, this model is not strictly adequate for a random variable which takes values on the interval $[0, 255]$, and the assumption is not completely valid since some circle points could correspond to the ink trace. This model could be improved by using a probability density function suitable for representing a random variable that takes values on a finite interval. However, as a first order approximation we have shown that this model provides good results in pen up/down detection.

The detection of the points in which the pen is moving above the paper is based on local measurements of brightness. A few other local measurements such as the local orientation of the ink at position of the ballpoint, the correlation of this orientation with the local direction of the pen tip's trajectory, etc., could be used in order to improve the detection rates. These local measurements of direction would decrease the FAR since a sample would be classified as "pen down" only if an ink trace with the corresponding direction is found at the sample's location. These different measurements could be naturally included in the system by increasing the dimensionality of the HMM's observation.

The set of examples used to estimate the HMM parameters and to evaluate the pen up/down performance included examples of different types of writing provided by only one subject. More example sequences provided by different subjects should be acquired in order to estimate the pen up/down performance in a writer-independent setting.

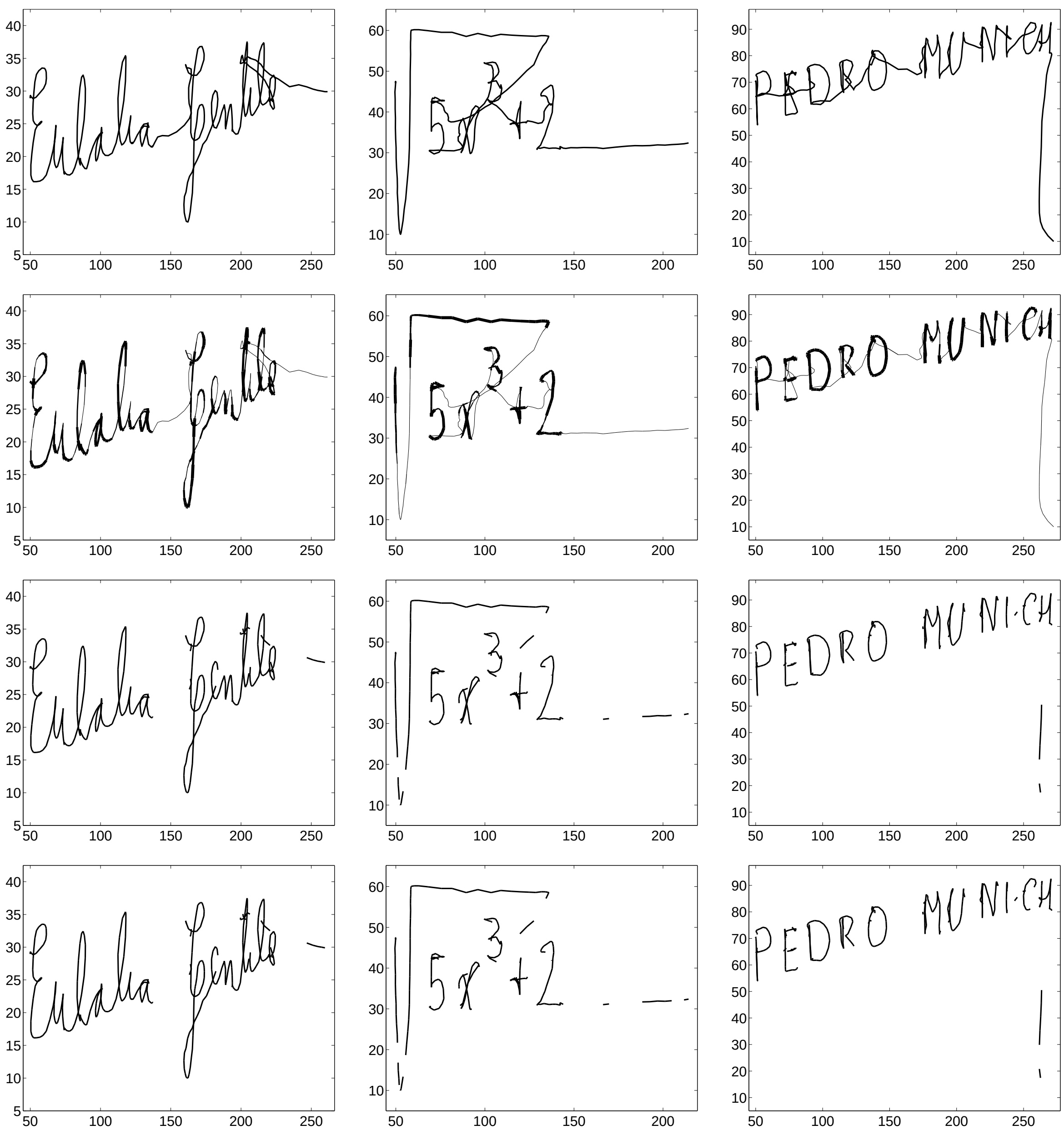


Figure 2.20: Three examples of test sequences are shown on the first row. The plots of the second row have the thickness of the segments proportional to the mean of the confidence measure of ink presence of the segment endpoints. The result of thresholding this confidence measure is shown in the third row. The fourth row shows segments whose extrema are classified as pen down by the HMM.

Figure 2.21: The first row shows the segmented trajectories, where the different segments are plotted with either solid or dashed lines. In the second and fourth rows the thickness of the strokes is proportional to the confidence measure obtained in the two cases mentioned above (aggregation of local ink measurements and voting based on HMM states). The third and fifth rows show the strokes after performing a hard classification.

Chapter 3 Signature Verification

3.1 Introduction

As we discussed previously, the scientific and technologic advances achieved during the twentieth century have brought dramatic changes in life quality. Automatic machines ranging from a simple mixer to all sorts of computer and computerized devices were introduced to the common everyday life, resulting in an incredible increase in interaction between humans and machines. Many machines are also used to control access to resources. Individuals have to show proof of identity in order to be allowed to use a particular asset. There are many examples in our everyday life in which we are required to prove that *we are who we claim we are*; we use a PIN number at a teller machine to get cash, we sign a credit card slip to make a purchase, we present appropriate credentials to gain access to a particular place, etc.

The most common personal identification methods fall within the following four categories: physical entities that are of known origin (e.g., an identification card, a passport), information known on a restricted basis (e.g., PIN number, password), specific individual activity pattern (e.g, signature, voice-print) or recognizable and unique physical characteristics of an individual (e.g., fingerprint, hand geometry, iris pattern).

The techniques in the two first categories are already widely in use in our society. However, these techniques may cause common security breaches since identification cards are easily forged and passwords or PIN numbers are easily stolen, lost or forgotten. This is the main reason for developing methods for authentication falling into the two last categories. These methods rely on biometrics, defined by the Association for Biometrics as:

> "A measurable, physical characteristic or personal behavioral trait used

to recognize the identity, or verify the claimed identity, of a person whose reference data is on file."

The biometric measurements can be subdivided into *physiological* biometrics, i.e., biometrics characterized by a physical characteristic (fingerprint, iris pattern, etc.), and *behavioral* biometrics, i.e., biometrics characterized by a behavioral trait that is learned and acquired over time (signature, keystroke dynamics, etc.).

Given a biometric sample from an individual, there are two different classification possibilities. *Authentication* or *Verification* is the one-to-one process of comparing the sample against the biometric reference template of a single enrollee whose identity is being claimed, to determine whether it matches the enrollee's template. *Identification* is the one-to-many process of comparing the sample against all of the biometric reference templates on file to determine whether it matches any of the templates and, if so, the identity of the enrollee whose template was matched.

There are many approaches to automatic identification using biometrics. Among the vision-based ones, we can mention face recognition [72, 75, 82], fingerprint recognition [36], iris scanning [19] and retina scanning. Voice recognition or signature verification are the most widely known among the non-vision based ones. Signature verification is a behavioral biometric that analyzes the way an end user signs his/her name.

In most systems, signature verification requires either the use of electronic tablets or digitizers for on-line capturing [40] or optical scanners for off-line conversion [71]. These interfaces are bulky (they need to have at least the minimum area required to sign) and involve the presence of dedicated hardware. Cameras, on the other hand, are much smaller and are becoming ubiquitous in the current computer environment. We have demonstrated in chapter 2 the feasibility of using a visual interface that can be built with video technology and computer vision techniques to capture handwriting in general and signatures in particular, and we now propose the use of this interface for signature verification.

Automatic signature verification systems involve two processing modes. In the *training* mode the user provides signature samples that are used to construct a model that statistically represents the characteristics of the signer. In the *testing* mode the user provides a new signature along with the alleged identity and the system judges the likely authenticity of the presented sample with respect to the alleged class model. Figure 3.1 summarizes the process.

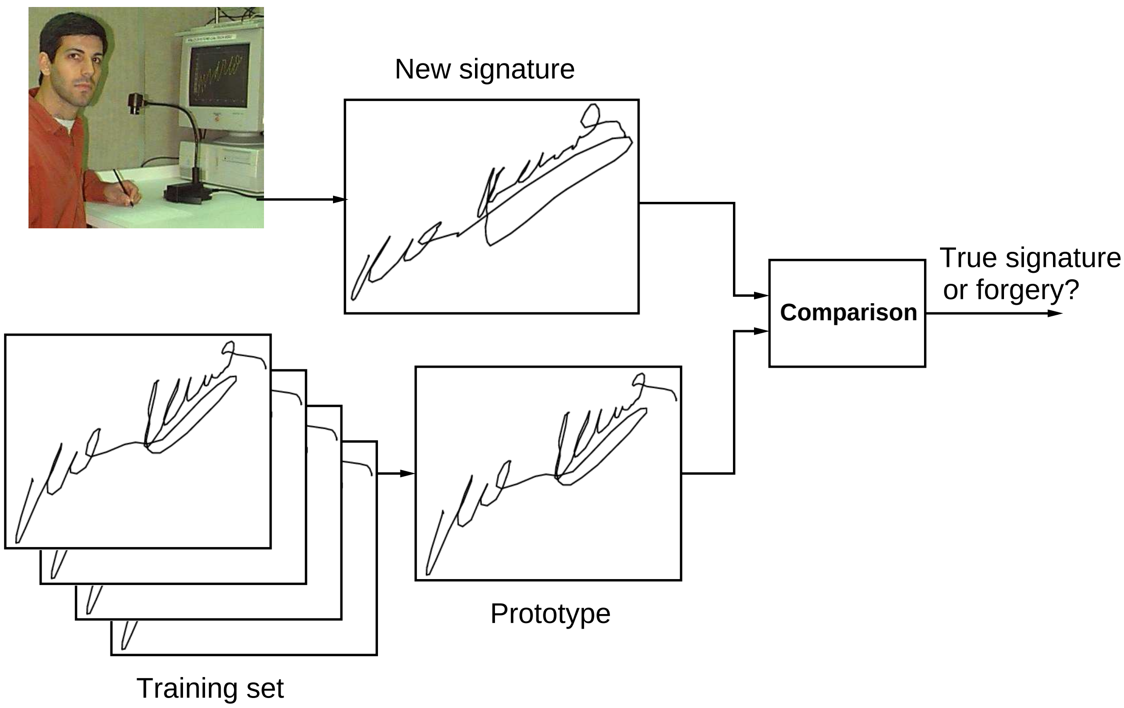


Figure 3.1: Signature verification system.

The literature on signature verification is quite extensive (see [40, 41, 58] for very comprehensive surveys) and is divided into two main areas of research, off-line and on-line systems. Off-line systems deal with a static image of the signature, i.e., the result of the action of signing, while on-line systems have available the dynamic process of generating the signature, i.e., the action of signing itself. The system described in this chapter falls within the category of on-line systems since the visual handwriting tracker captures the timing information in the generation of the signature.

On-line signature verification methods may be classified into two principal groups depending upon the features used to perform classification. On one group, complete signals (position, speed, acceleration, pressure, etc., versus time) are considered as mathematical functions whose value constitute the feature set. On the other group, global parameters (e.g., total duration, number of components, overall dimension, etc.) and local parameters (e.g., maximum values, peak curvatures, etc.) are computed from the measured signals to constitute the features. Each approach provides benefits and imposes problems on the design of the system. A successful signature verification system ideally should make use of both types of features [40, 50].

This chapter is organized as follows. Section 3.2 describes the method used to perform the comparison between signatures, section 3.3 discusses the different parameterizations of the signatures used in order to improve performance, section 3.4 shows the method used to evaluate the performance of the verification system and section 3.5 provides the experimental results.

3.2 Algorithm for signature comparison

3.2.1 Preliminaries

Signatures can be regarded as planar curves, i.e., the locus of points $C(p) = [x(p), y(p)] \in \mathbb{R}^2$, with $p \in [0, 1]$. Given two different curves, the straightforward method to measure their similarity is linear correlation. However, this method is not suitable for signature comparison. Signatures vary in length, even if they are generated by the same writer. Besides, there are random variations that create additions and deletions of portions of signals, non-linear time axis compression or expansion and gaps due to pauses or hesitations of the writer.

Different methods have been proposed in the literature in order to determine the best correspondence between two signature samples and provide a measure of their similarity. Dynamic Programming Matching (DPM) is a technique that finds for each sample in one of the signatures, the corresponding sample in the other signature that is closest to the original sample using some predefined metric. Given this correspondence, it is possible to calculate a "distance" between the signatures under comparison.

DPM belongs to the class of dynamic programming [3] algorithms that were proposed and extensively studied by Bellman and collaborators in the '50s-'60s. The use of DPM for comparison of time functions was initially proposed in the field of Speech Recognition by Sakoe and Chiba [64] and it is described in full extent in the book of Rabiner and Juang [62] with the name of Dynamic Time Warping. DPM has been successfully used for signature verification by many researchers [35, 24, 34, 45, 48, 49, 50, 54, 65, 81] and it is the technique that we use to compare signatures acquired with our visual pen tracking system.

Sato and Kogure [65] proposed to use DPM in order to align the shape of signatures, consisting only of pen-down strokes, after having normalized the data with respect to translation, rotation, trend and scale. They further used the result of DPM in order to compute the alignment of the pressure function and to compute a measure of the difference in writing motion. Finally, they perform the classification based on the residual distance between shapes after time alignment, the residual distance between pressure functions and the distance between writing motions.

Parizeau and Plamondon [54] evaluated the use of DPM for signature verification by aligning either horizontal or vertical position ($x(t)$, $y(t)$), horizontal or vertical velocity ($v_x(t)$, $v_y(t)$), or horizontal or vertical acceleration ($a_x(t)$, $a_y(t)$). In their work, they used the complete signing trajectories, i.e., pen-down and pen-up strokes.

Hastie et al. [34] obtain a statistical model of signatures that allows for variations in the speed of writing as well as affine transformations. DPM is used to find the correspondence between speed signals of pairs of signatures, and the distance measure provided by the DPM in the speed domain is used as a classification parameter. The signature with lowest distance to all others is chosen as the reference. The speed signal of the reference pattern is used to segment this signature into letters. After segmenting all the signatures by using the correspondence provided by DPM, a template signature at the letter level is extracted and used for the final comparison and classification.

Huang and Yan [35] presented the use of DPM for matching signature strokes by finding a warp path that minimizes the cost of aligning the shape, velocities and accelerations of the individual strokes all at the same time. Pen up strokes are considered in the preprocessing phase of their algorithm, in order to be merged with the pen down strokes.

Nalwa [50] parameterized the pen-down strokes of the signature along its arc length and then compute a number of characteristic functions such as coordinates of the center of mass, torque, and moments of inertia using a sliding computational window and a moving coordinate frame. He performed a simultaneous dynamic programming matching over arc length of all these characteristic functions for the two signatures under comparison. A measure of the similarity of the signatures is used for classification.

Our implementation of DPM for signature verification attempts to perform the best time alignment of the 2D shape of the signatures using a translation-invariant measure of curve similarity, i.e., we find the time warping function that has the minimum cost of aligning the planar curves that represent signatures. We note that the pen-up strokes drawn by each subject were as consistent as the pen-down strokes, as shown in figure 3.2. This observation agrees with the evidence [41] that signatures are produced as a ballistic or reflex action, without any visual feedback involved. Therefore, we use the full signing trajectory in our experiments. We do not perform any type of normalization on the signatures since we consider that users are very consistent on their style of signing; they write their signatures with a similar slant, in a similar amount of time, with similar dimensions, and with a similar motion.

3.2.2 Curve Matching using Dynamic Programming

A translation-invariant measure of curve similarity

Given two two-dimensional curves $C_1 = \{\mathbf{X}(i), i = 1, \cdots, N_x\}$ and $C_2 = \{\mathbf{Y}(j), j = 1, \cdots, N_y\}$ as in figure 3.3, and assuming that we have a warping or correspondence map $\phi = (t, s)$ between C_1 and C_2, such that a point $\mathbf{X}(t_k) \in C_1$ corresponds to a point $\mathbf{Y}(s_k) \in C_2$, for $k \in \{1, \cdots, N\}$, $t_k \in \{1, \cdots, N_x\}$, $s_k \in \{1, \cdots, N_y\}$. Let us define the elementary distance $d((t_{k-1}, s_{k-1}), (t_k, s_k))$ [67] of having $\mathbf{X}(t_k)$ in correspondence with $\mathbf{Y}(s_k)$ and $\mathbf{X}(t_{k-1})$ in correspondence with $\mathbf{Y}(s_{k-1})$ as

Figure 3.2: Signatures acquired with the visual interface and corresponding image captured by the camera after finishing the acquisition. We observe that the pen-up strokes are as consistent as the pen-down ones.

$$d((t_{k-1}, s_{k-1}), (t_k, s_k)) = \|\overrightarrow{\mathbf{X}(t_k)\mathbf{Y}(s_k)} - \overrightarrow{\mathbf{X}(t_{k-1})\mathbf{Y}(s_{k-1})}\|^2 \tag{3.1}$$

where $\|.\|^2$ is the Euclidean norm. Let us define the similarity measure between the two curves given ϕ as

$$\mathbf{D}_\phi(C_1,C_2) \triangleq \sum_{k=2}^{N} d((t_{k-1},s_{k-1}),(t_k,s_k)) = \sum_{k=2}^{N} \|\overrightarrow{\mathbf{X}(t_k)\mathbf{Y}(s_k)} - \overrightarrow{\mathbf{X}(t_{k-1})\mathbf{Y}(s_{k-1})}\|^2$$
$$= \sum_{k=2}^{N} \|\mathbf{X}(t_k) - \mathbf{Y}(s_k) - \mathbf{X}(t_{k-1}) + \mathbf{Y}(s_{k-1})\|^2$$
$$= \sum_{k=2}^{N} \| \underbrace{(\mathbf{X}(t_k) - \mathbf{X}(t_{k-1}))}_{\text{velocity } \mathbf{X}} - \underbrace{(\mathbf{Y}(s_k) - \mathbf{Y}(s_{k-1}))}_{\text{velocity } \mathbf{Y}} \|^2 \tag{3.2}$$

where the two curves are assumed to be sampled at a constant sampling rate. It is easy to see that the distance defined in equation 3.2 is invariant with respect to any translation of C_1 and C_2.

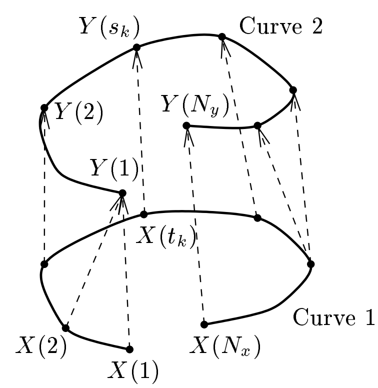


Figure 3.3: Correspondence between the two curves C_1 and C_2.

Having defined the distance between curves given the warping function $\phi = (t,s)$, the actual problem to solve is: Find the function ϕ that minimizes $\mathbf{D}_\phi(C_1,C_2)$ and report the resulting minimum distance between the curves. In other words,

$$\text{Solve} \quad \begin{cases} \mathbf{D}(C_1,C_2) = \underset{\phi}{min}\ \mathbf{D}_\phi(C_1,C_2) \\ \phi = (t,s) = \underset{\phi}{argmin}\ \mathbf{D}_\phi(C_1,C_2) \end{cases} \tag{3.3}$$

where ϕ has to satisfy a set of requirements (to be presented below).

The matching process could be visualized on the "warping plane" of figure 3.4 where t_k will be represented on the x-axis and s_k will be represented on the y-axis. The set of sample points on C_1 and C_2 defines a grid on the warping plane. The correspondence function ϕ joins different nodes of the grid and defines a curve or a path on this plane. If the warping path crosses one vertex (i, j) of the grid, it means that point $\mathbf{X}(i) \in C_1$ corresponds to $\mathbf{Y}(j) \in C_2$. This warping path is parameterized by $k \in \{1, \cdots, N\}$ as shown in figure 3.4.

The simplest case of correspondence between the two curves is the one in which the rates of production of the curves are constant and proportional to the duration of the curves. The individual warping functions are then related by a linear relationship $t_k = \frac{N_1}{N_2} s_k$. The corresponding warping path is a diagonal straight line as shown in figure 3.4. In the case of signatures, practical experience indicates that it is not possible to get two signatures that are exactly the same (in fact, if this happens, one of the signatures is certain to be a forgery) since it is quite difficult for humans to repeat precisely the same pattern at the same rate. In general, two signatures from the same subject are quite similar but present local shape deformations and local differences in rate of production. Therefore, we need to obtain two warping functions t_k and s_k that define a warping path that takes into account these local differences. The generation of signatures is implicitly a causal process, so the matching between two signatures has to preserve this temporal ordering. This constraint means that the functions t_k and s_k have to be monotonically nondecreasing as depicted in figure 3.4(b).

Dynamic programming matching (DPM)

The solution of the high-dimensional minimization presented in equation 3.3 has in general combinatorial complexity in the number of samples. However, given that the functions t_k and s_k have to be monotonically nondecreasing in order to preserve the temporal ordering on the generation of the signatures, the matching between the signatures has to be a causal process. Under these conditions, Dynamic Programming allows one to perform the full minimization as a sequence of one-dimensional minimizations by applying Bellman's *principle of optimality* [3, 20]:

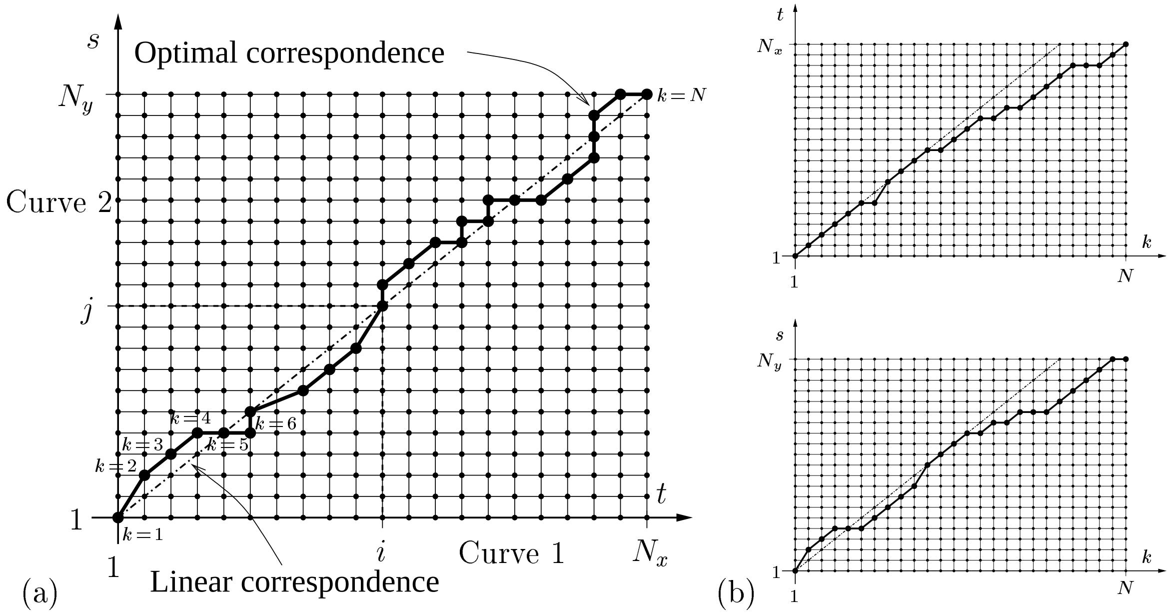


Figure 3.4: (a) Matching process displayed on the "warping plane." The dashed line shows the linear correspondence between the curves while the solid line shows the optimal matching obtained as a solution of the equation 3.6. (b) Corresponding warping functions t_k and s_k.

> "An optimal sequence of decisions in a multistage decision process problem has the property that whatever the initial stage, state, and decision are, the remaining decisions must constitute an optimal sequence of decisions for the remaining problem, with the stage and state resulting from the first decision considered as initial conditions."

Let us put the principle of optimality in mathematical terms in order to obtain an algorithm. Let us assume that $(t_1, s_1) = (1, 1)$ and $(t_N, s_N) = (N_x, N_y)$, i.e., the first and last points of C_1 and C_2 are in correspondence, then we can rewrite equation 3.3 in terms of N_x and N_y as

$$\mathbf{D}(C_1, C_2) \triangleq \mathcal{D}(N_x, N_y) = \min_{t,s} \sum_{k=2}^{N} d((t_{k-1}, s_{k-1}), (t_k, s_k)) \tag{3.4}$$

Similarly, the minimum cumulated distance along a warping path starting at $(t_1, s_1) = (1, 1)$ and ending at $(t_{N'}, s_{N'}) = (n_x, n_y)$ is

$$\mathcal{D}(n_x, n_y) = \min_{t,s,N'} \sum_{k=2}^{N'} d((t_{k-1}, s_{k-1}), (t_k, s_k)) \tag{3.5}$$

Let us assume that we have $\mathcal{D}(n'_x, n'_y)$, the solution of equation 3.5 up to node (n'_x, n'_y) in the grid of the warping plane. Using the principle of optimality, we want to express the minimization up to point (n_x, n_y) in terms of $\mathcal{D}(n'_x, n'_y)$. Let us call $\xi((n'_x, n'_y), (n_x, n_y))$ the cumulated local distance for the portion of the warping path between points (n'_x, n'_y) and (n_x, n_y), that is composed of L segments. Then, $\xi((n'_x, n'_y), (n_x, n_y)) = \sum_{m=0}^{L} d((t_{N'-m-1}, s_{N'-m-1}), (t_{N'-m}, s_{N'-m}))$ where $(t_{N'}, s_{N'}) = (n_x, n_y)$ and $(t_{N'-L}, s_{N'-L}) = (n'_x, n'_y)$. The shape and the number of segments of the portion of the warping plane between points (n'_x, n'_y) and (n_x, n_y) are defined by the local constraints imposed onto ϕ that are discussed later in this section. Figure 3.5 shows different possibilities for the portion of the path joining nodes (n'_x, n'_y) and (n_x, n_y) that are proposed in the literature. Using the principle of optimality, we can write the following recursion to solve the minimization of equation 3.3

$$\boxed{\mathcal{D}(n_x, n_y) = \min_{(n'_x, n'_y)} \{\mathcal{D}(n'_x, n'_y) + \xi((n'_x, n'_y), (n_x, n_y))\}} \tag{3.6}$$

where (n'_x, n'_y) represent all possible nodes that can be joined with node (n_x, n_y) by a portion of the path that satisfies the constraints imposed on the minimization.

Matching constraints

Unconstrained minimization in equation 3.3 may conceivably result in a near-perfect match between any two different signatures, thus making the comparison meaningless for recognition purposes. Figure 3.6 shows an extreme example of this problem. The lower curve is the same in both cases and plays the role of the reference signature. Given the correspondence between samples shown in the figure, the distance between the curves in both cases is the same.

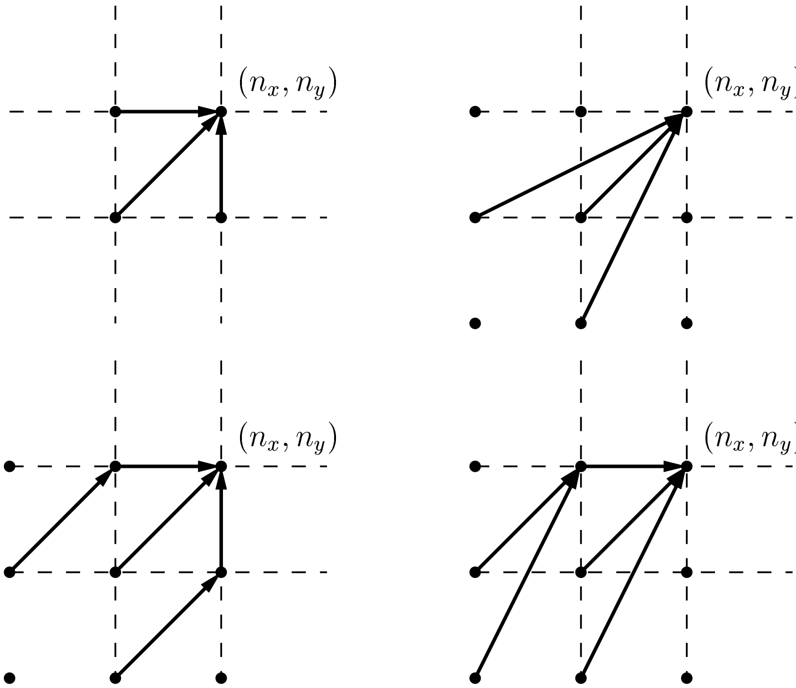


Figure 3.5: Different possible portions of warping path ending at node (n_x, n_y) proposed in the literature.

The solution of the minimization of equation 3.3 can be obtained with the dynamic programming algorithm with mathematical precision. However, for the matching process to be meaningful in terms of time normalization for different realizations of a signature, some constraints on the warping functions are necessary. These constraints have a heuristic nature since they are based upon intuition and are not motivated by analytical results. Typical time warping constraints that are considered reasonable for dynamic programming matching include endpoint constraints, monotonicity conditions, local continuity constraints, global path constraints and slope weighting

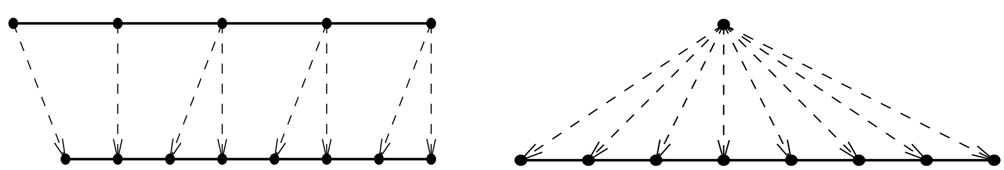


Figure 3.6: Extreme example of unconstrained minimization of equation 3.3. The lower curve is the same in both cases. Given the correspondence between samples shown in the figure, the distance between the curves is the same in both cases.

(see [62] for a more extensive treatment of the subject).

Endpoint constraints Signatures are a short piece of handwriting that have a well defined beginning and ending, so it seems reasonable to constrain ϕ such that the first and last points of C_1 and C_2 be in correspondence. These conditions have the following expression:

$$\begin{cases} \text{beginning point} & t_1 = 1 \quad\quad s_1 = 1 \\ \text{ending point} & t_N = N_1 \quad s_N = N_2 \end{cases} \tag{3.7}$$

Monotonicity conditions The generation of a signature is a sequential process whose final product is a parameterized 2D curve. The matching algorithm has to maintain the temporal order while assigning sample correspondences; it is therefore reasonable to impose the following monotonicity constraints on ϕ

$$\begin{cases} t_{k+1} \geq t_k \\ s_{k+1} \geq s_k \end{cases} \tag{3.8}$$

This constraint eliminates the possibility of reverse warping along the time axis, even within a short time interval. This constraint implies that any path on figure 3.4 cannot have negative slope.

Local continuity constraints Signatures can have random variations that produce additions and deletions of portions of the written trace as shown in figure 3.7. Local continuity constraints are designed to provide the flexibility needed to cope with these situations while, at the same time, ensuring proper time alignment with minimum potential loss of information.

These constraints define the possible ways of computing the cumulated local distance of a portion of the warping path $\xi((n'_x, n'_y), (n_x, n_y))$ that joins nodes (n'_x, n'_y) and $(n_x, n_y))$ of the warping plane of figure 3.4. The local continuity constraints can

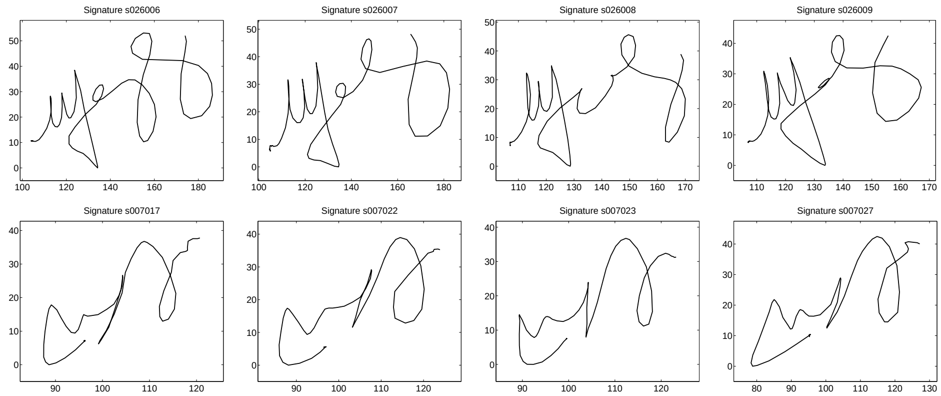


Figure 3.7: The first row shows signatures from a subject that does not sign consistently. The middle loops are added, deleted and distorted quite a bit from one signature to the other. The second row shows signatures from a much more consistent subject, although there is some distortion between signatures.

take many forms as shown in figure 3.5 (see Rabiner and Juang [62] for a detailed list of the constraints proposed in the literature). Sakoe and Chiba [64] proposed the following ones:

$$\begin{cases} t_{k+1} - t_k \geq 1 \\ s_{k+1} - s_k \geq 1 \end{cases} \tag{3.9}$$

The constraints are difficult to visualize using equations, so it is more convenient to show them in terms of incremental path changes on the warping plane. Figure 3.8 pictorially shows the local continuity constraints described by equation 3.9 and the corresponding matching between samples of curves C_1 and C_2.

Given the local continuity constraints of equation 3.9, $\xi((n'_x, n'_y), (n_x, n_y))$ is composed by only one segment and it is computed as follows:

$$\begin{aligned} \xi((i-1, j-1), (i, j)) &= d((i-1, j-1), (i, j)) \\ \xi((i, j-1), (i, j)) &= d((i, j-1), (i, j)) \\ \xi((i-1, j), (i, j)) &= d((i-1, j), (i, j)) \end{aligned}$$

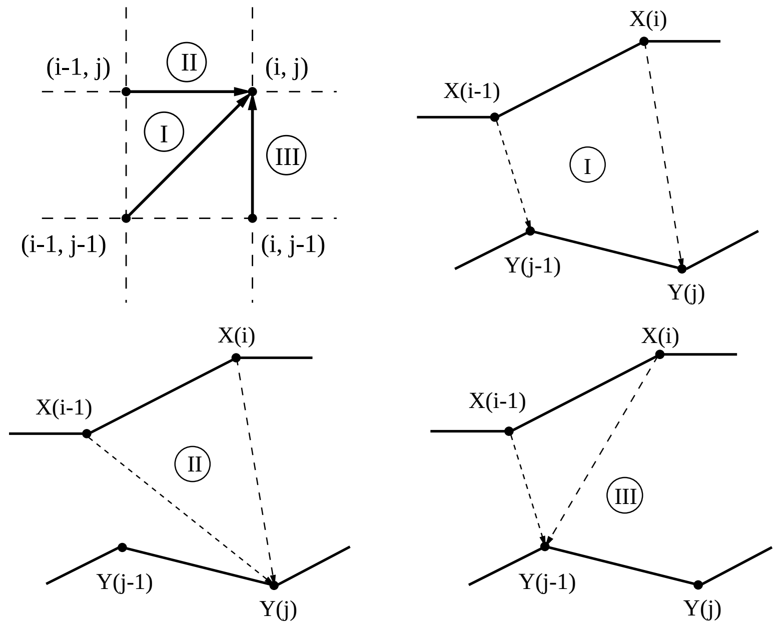


Figure 3.8: Local continuity constraint and the corresponding matching between samples of curves C_1 and C_2.

Global path constraints The local continuity constraints define a region of the warping plane that the optimal path can traverse. Using the constraint depicted in figure 3.8, all nodes of the warping plane could be reached. However, other local constraints may restrict the reachable nodes to the ones belonging to a certain region of the warping plane. For example, using the local constraint shown in the upper-right plot of figure 3.5 restrains the warping path to have a slope ranging from 0.5 to 2. Global constraints can be imposed on ϕ that also limit the allowed region of the warping plane. Constraints on the gradient of the warping function restrain the possibility of unrealistic correspondence between a very short pattern in a signature and a relatively long pattern in the other. If the maximum and minimum possible slope for the warping path are fixed to be M_{max} and $\frac{1}{M_{max}}$, the global constraints imposed on ϕ are as follows:

$$\begin{cases} 1 + \dfrac{t_k - 1}{M_{max}} \leq s_k \leq 1 + M_{max}(t_k - 1) \\ N_y + M_{max}(t_k - N_x) \leq s_k \leq N_y + \dfrac{t_k - N_x}{M_{max}} \end{cases} \tag{3.10}$$

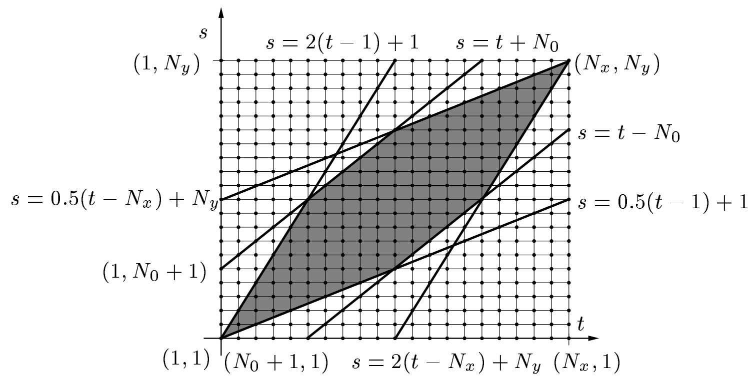


Figure 3.9: The shaded region is the allowed region of the warping plane that the warping path can traverse due to global constraints.

where the first equation specifies the range of points (i, j) of the warping plane that can be reached starting from the point $(1, 1)$ and the second equation specifies the range of points that have a legal path to the ending point (N_x, N_y). Constraints on the maximum timing difference between the two signatures are also desirable since they restrict the possibility of accepting a forgery that was obtained by tracing a real signature at a slow pace. Sakoe and Chiba [64] proposed the following one

$$|t_k - s_k| \leq N_0 \tag{3.11}$$

where N_0 is the maximum allowable absolute deviation between any two samples in each signature. Figure 3.9 shows the available region of the warping plane after applying the above global constraints ($M_{max} = 2$).

Using the recursion equation 3.6 and all the mentioned constraints, obtaining the optimal path is straightforward. For each node (i, j) within the allowed region of the warping plane, the minimum cumulated cost is computed sequentially column-wise or row-wise. The previous node that provides the minimum cost is stored in memory. Finally, the last column or row is searched for the node with minimum cost and then the optimum warping path is found by backtracking the stored nodes.

Let us summarize the Dynamic Programming Matching algorithm using all the constraints. In order to backtrack the warping path after obtaining the minimum distance, the algorithm needs to recall the parent node of each point (i, j) within the allowed region of the warping plane. Let us store the parent one of point (i, j) in $\zeta(i, j)$.

1 Initialization:

$$\mathcal{D}(1,1) = 0$$
$$\zeta(1,1) = (1,1)$$

2 Recursion: for $1 \leq i \leq N_x$, $1 \leq j \leq N_y$, such that i and j stay within the allowed grid and follow the monotonicity constraints, compute:

$$\mathcal{D}(i,j) = min \begin{cases} \mathcal{D}(i-1,j) + d((i-1,j),(i,j)) \\ \mathcal{D}(i-1,j-1) + d((i-1,j-1),(i,j)) \\ \mathcal{D}(i,j-1) + d((i,j-1),(i,j)) \end{cases} \tag{3.12}$$

$$\zeta(i,j) = argmin \begin{cases} \mathcal{D}(i-1,j) + d((i-1,j),(i,j)) \\ \mathcal{D}(i-1,j-1) + d((i-1,j-1),(i,j)) \\ \mathcal{D}(i,j-1) + d((i,j-1),(i,j)) \end{cases}$$

3 Termination:

$$\mathbf{D}(C_1, C_2) = \mathcal{D}(N_x, N_y)$$
$$\phi_1 = (N_x, N_y)$$

4 Path Backtracking:

$$\text{do } \phi_{i+1} = \zeta(\phi_i) \text{ until } \phi_{i+1} = (1,1)$$

The resulting function ϕ is ordered backwards, i.e., starting from the last matching points, so it should be reordered if desired.

Figure 3.10 shows an example of dynamic programming matching applied to compare the 2D shape of two signatures. The first column shows the horizontal coordinate $x(t)$ of both signatures and the second column shows the vertical coordinate $y(t)$ of both signatures, before and after matching. The upper plot of the third column shows the two signatures under comparison and the lower plot of the third column shows the warping path. We note that the matching is quite good regardless of the differences in the shapes of $x(t)$ and $y(t)$. The remaining mismatch between these signals accounts for the differences in shape of the signatures.

The algorithm presented for curve matching has two components, the dynamic programming minimization that is mathematically exact and the minimization constraints that are heuristically motivated. Therefore, the warping path obtained with the algorithm provides the best correspondence in terms of curve distance, under the mentioned constraints. If C_1 and C_2 are two examples of the same signature, this warping function represents the best correspondence between the curves after compensating for local shape deformations and local differences in the rate of production of the signatures. However, if C_1 and C_2 are two completely different signatures, the optimal path is not really meaningful, except as the solution of the dynamic programming process, because establishing correspondence between two completely different curves is not a well defined concept.

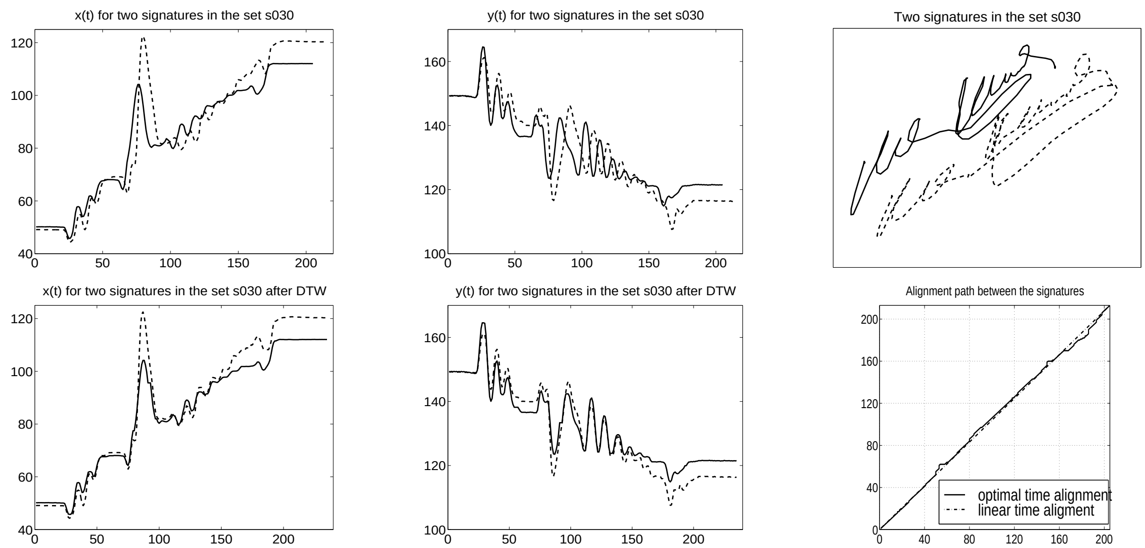


Figure 3.10: Example of dynamic programming matching applied to compare the 2D shape of two realizations of the same signature. The first column shows the horizontal coordinate $x(t)$ of both signatures before and after matching; the second column shows the vertical coordinate $y(t)$ of both signatures before and after alignment. The upper plot of the third column shows the two examples of the signature. The lower plot of the third column shows the optimal time warping path compared with a linear time matching path.

3.3 Signature parameterization

3.3.1 Preliminaries

In most of the previous work, a time-based parameterization of the functions to be compared was used, even though there is no clear reason for using this parameterization other than the convenience of being automatically provided by the capture device. To our knowledge, only Nalwa [50] used an arc-length parameterization of the signatures for computing the distinctive functions proposed in his paper. The arc-length parameterization of the signature is loosely dependent on time and on the dynamics of signing, even though it keeps the causality of the signature's generation. This weak dependence on the dynamics of signing seems contrary to the traditional idea that pen dynamics is a key element in detecting forgeries. However, the use of the arc-length parameterization is a first step towards achieving invariance with respect to Euclidean transformations of the signatures. Going one step further, we could use a parameterization that provides a certain degree of invariance with respect to affine transformations of the signatures. This parameterization has been described in the literature [7] and has been called *affine arc-length* by Pollick and Sapiro [60].

Several studies (see [60, 77, 76, 39] and references therein) show that the generation and perception of planar movements by humans have a direct relationship between the tangential velocity of the hand and the radius of curvature of the planar curve. Experimental results show that the tangential velocity decreases as the curvature increases. A mathematical fitting of these results gives rise to a power law in which the tangential velocity is proportional to the $\frac{1}{3}$ power of the radius of curvature. While the relationship between these two quantities is very intuitive, there is no clear explanation for the exact factor $\frac{1}{3}$ in the power law. Pollick and Sapiro [60] show that this power law precisely implies motion at a constant affine velocity. This means that curves with equal affine length will be drawn in equal time. The main question is why affine parameters seem to be embedded in the representation of planar motion. One possible explanation presented in [60] notes that affine transformations are obtained when a planar object is rotated and translated in space, and then projected into the eye via parallel projection. This approximated model for the human visual system is valid when the object is flat enough and away from the eye, as in the case of drawing and planar point motions. These observations are the main motivation for using affine arc-length in our experiments.

3.3.2 Euclidean and affine arc-length

Let's define the relations used to re-parameterize the signatures on Euclidean and affine arc-lengths. A planar curve may be defined as the locus of points $C(p) = [x(p), y(p)] \in \mathbb{R}^2$, with $p \in [0,1]$. Different parameterizations p define the same curve but give rise to different velocities along the curve $\frac{\partial C}{\partial p}$. Given an increasing function $q(p) : [0,1] \to [0,1]$, the curve defined by $C(q) = C(q(p))$ will be the same as the one defined $C(p)$, even though the velocities along the curve will be different $\frac{\partial C}{\partial p} \neq \frac{\partial C}{\partial q}$.

One of the most well-known parameterizations is the *Euclidean arc-length* ν defined such as the curve is traveled with constant velocity, i.e., $\|\frac{\partial C}{\partial \nu}\| = 1$. Given our curve C, parameterized with an arbitrary parameterization p, in order to re-parameterize it in Euclidean arc-length, we use the relation

$$\nu(p) = \int_0^p \|\frac{\partial C(t)}{\partial t}\| \, dt \tag{3.13}$$

where it is easy to see that since $C(\nu) = C(\nu(p))$, then, $\frac{\partial C}{\partial \nu} = \frac{\partial C}{\partial p}\frac{\partial p}{\partial \nu}$, and $\|\frac{\partial C}{\partial \nu}\| = 1$. Given a curve with Euclidean arc-length parameterization and two points ν_0 and ν_1 on the curve, the Euclidean length between them is

$$l_e(\nu_0, \nu_1) \triangleq \int_{\nu_0}^{\nu_1} d\nu \tag{3.14}$$

The Euclidean arc-length parameterization defines a length that is invariant with respect to rotations and translations (Euclidean transformations).

If we allow for affine transformations rather than Euclidean ones, the Euclidean length ν is not invariant any more. A new parameterization s on *affine arc-length* is defined such that the resultant affine length is invariant with respect to affine transformations. As in the case of the Euclidean arc-length, given the curve with an arbitrary parameterization p, the re-parameterization in affine arc-length s is

$$s(p) = \int_0^p |\frac{\partial C}{\partial t} \times \frac{\partial^2 C}{\partial t^2}|^{\frac{1}{3}} \, dt \tag{3.15}$$

Based on this parameterization, the affine length between two points s_0 and s_1 of the curves is

$$l_e(s_0, s_1) \triangleq \int_{s_0}^{s_1} d\nu \tag{3.16}$$

which can be shown to be invariant with respect to affine transformations [7, 60]. The curve $C(s) = C(s(p))$ parameterized on affine arc-length satisfies the following relation

$$\left|\frac{\partial C}{\partial s} \times \frac{\partial^2 C}{\partial s^2}\right| = 1 \tag{3.17}$$

which means that the area of the parallelogram defined by the vectors $\frac{\partial C}{\partial s}$ and $\frac{\partial^2 C}{\partial s^2}$ is constant.

Figure 3.11 shows an example of a signature acquired with the visual tracking system and the corresponding re-parameterizations on Euclidean and affine arc-length.

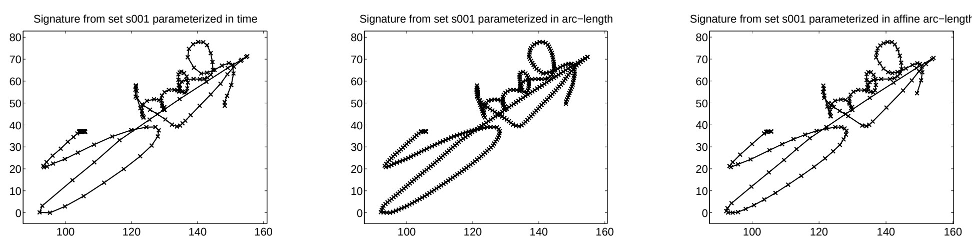


Figure 3.11: The first plot shows a signature acquired with our system and therefore parameterized in time. The second and third plots display the same signature parameterized in Euclidean and affine arc-length respectively.

3.4 Evaluation of the performance of the verification system

3.4.1 Error rates

Signature verification can be thought of as a two-class pattern recognition problem, one class consisting of genuine signatures and the other consisting of forgeries. A great deal of variability can be observed in signatures from the same individual according to country, age, time, habits, psychological or mental state, and physical and practical conditions. The only certainty in this domain is that when two signatures are identical, one of them is a forgery.

The performance of a verification system is generally evaluated according to the error representation of a two-class pattern recognition problem, that is, with Type I and Type II error rates. The Type I error rate (or False Rejection Rate (FRR)), measures the number of genuine signatures classified as forgeries as a function of the classification threshold. The Type II error rate (or False Acceptance Rate (FAR)), evaluates the number of false signatures classified as real ones as a function of the classification threshold.

Clearly, we can trade-off one type of error for the other type of error. As an extreme example, if we accept every signature, we will have a 0% FRR and a 100% FAR, and if we reject every signature, we will have a 100% FRR and a 0% FAR. The curve of FAR as a function of FRR, using the classification threshold as a parameter, is called the *error trade-off curve*. It provides the behavior of the algorithm for any operating regime and it is the best descriptor of the performance of the system. In practice, this curve is often simplified into a scalar, the *equal error rate*, i.e., the error rate at which the percentage of false accepts equal the percentage of false rejects. This equal error rate provides an estimate of the statistical performance of the algorithm, i.e., it provides an estimate of its generalization error. Figure 3.12 shows the curves of FRR and FAR as a function of the classification threshold and the corresponding error trade-off curve.

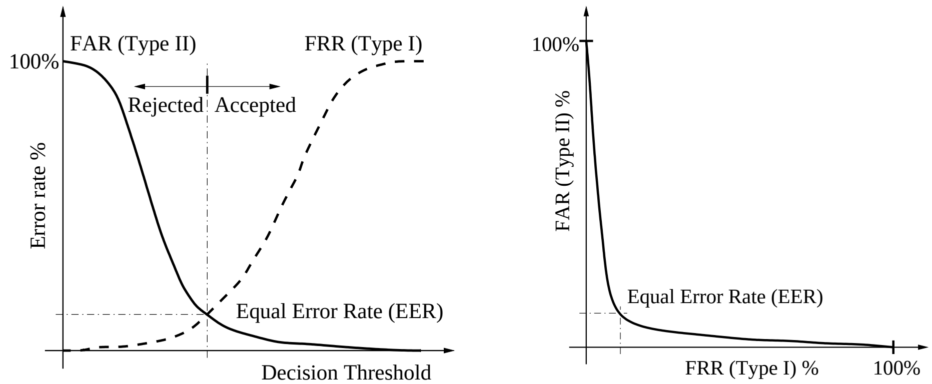


Figure 3.12: Curves of FRR and FAR as a function of the classification threshold and the corresponding error trade-off curve.

Depending on the testing conditions and on the availability of data, a signature verification system can be validated with different types of forgeries as discussed in the literature [58]. The two most common types of forgeries are the following:

Random Forgery : where the forger uses his own signature instead of the signature to be tested.

Skilled Forgery : where the forger tries and practices imitating as closely as possible the static and dynamic information of a signature.

The comparison among different signature verification systems that have been presented in the literature is quite difficult since it is hard to replicate the conditions of the experiments. Signatures are a behavioral biometric, so the generation of a signature is influenced by the state of mind of the signer. In other words, if the signer is well motivated, he would make an effort to provide good samples of his signature. Otherwise, if the subject does not concentrate on signing properly, the set of signatures could present quite a bit of unusual variability. The same can be said for the case of forgers. The use of random forgeries for evaluating a signature verification system provides artificially lower error rates and should be used as an initial sanity check, since a system that does not work well when tested with random forgeries most probably will not work properly with skilled forgeries.

3.4.2 Duplicated examples

One common problem of many on-line systems for signature verification is the lack of examples needed to build a reliable model for a signature and to assess the performance of the algorithm. This problem is inherent to the application since it is not feasible to ask a subject for all possible examples of his signature. Thus, we have to build a model of the signature that performs well in practice and we have to infer the generalization error of the algorithm, all with very few examples. If we know that the model that we are building should be invariant with respect to some transformation of the examples, we could increase the number of examples in both the training and test set by using *Duplicate Examples* as described by Y. Abu-Mostafa [1]. In our particular case, one possible example of this transformation is time origin translation since our system should be insensitive to the particular instant of time in which we started acquiring the signature. Another possible transformation is given by global affine deformation of the signatures, provided that the acceptable range of the parameters of this affine deformation could be estimated from the examples. This affine deformation of the signatures arises by modification of the position of the camera from acquisition to acquisition. Figure 3.13 shows a signature captured with the tracking system and a set of virtual examples obtained by time origin shifting and affine deformation.

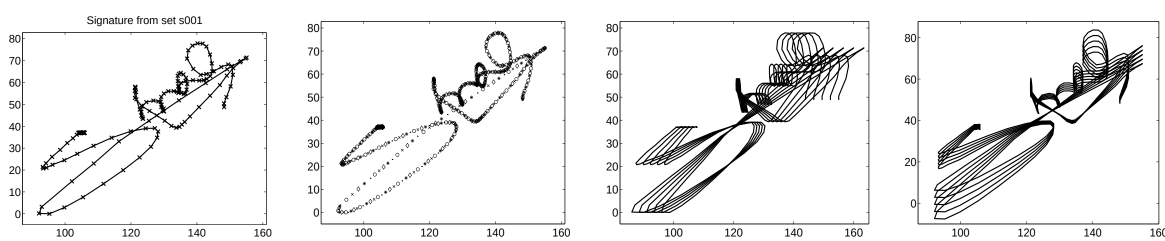


Figure 3.13: The first plot shows the original signature captured with the visual tracker. The second plot displays the position of the new samples when performing the time origin shifting. The third and fourth plots show the result of applying an affine scaling to the original signature, in the horizontal and vertical coordinates respectively. The maximum and minimum value of scaling to be applied is estimated from the training set.

3.5 Experiments

3.5.1 Data collection

We collected two set of signatures in order to evaluate the performance of the signature verification system. The first data set consists of signatures from 56 subjects, 18 of them were women and 4 were left handed. Each of them was asked to provide 25 signatures, 10 to be used as the training set and the other 15 to be used as the test set. The second data set consists of signatures from 49 different subjects (no intersection with the first set); 14 of them were women and 6 were left handed. All subjects were between twenty years old and sixty years old. Each of them was asked to provide 30 signatures, 10 to be used as the training set and the other 20 to be used as the test set. The first set was used throughout the development of the system so the resulting performance on set 1 could have been compromised by this fact. In other words, the resulting algorithm could have over-fitted the first signature set. The second set was kept unused until the system was fully developed and the performance on set 2 was computed only once in order to get the results presented in this chapter. Figure 3.14 shows one signature from each of the subjects in the databases. Figure 3.15 shows all the signatures for subjects s030 and s066 of the first database.

The data was collected in three sessions that took place on different days in order to get a sample of the variability of the subject's signatures while avoiding the distortion produced by the boredom of the repetitive task of signing. We should point out that the camera was not placed at a fixed position and height; it was changed from subject to subject and from session to session.

We also asked five of the signers to provide forgeries for each of the subjects in the database, as the ones shown in figure 3.18. Each set of forgeries for a particular subject was collected in one session. The naive forger was shown the ink trace of a set of real signatures and given enough time to practice the signature to be forged until feeling comfortable writing it. The set of forgeries was collected in two groups of 5 signatures each, giving the forger some rest in between. The visual tracker was set up such that the signer could not remain still in the same place for more than a few

Figure 3.14: One signature from each of the subjects in the database.

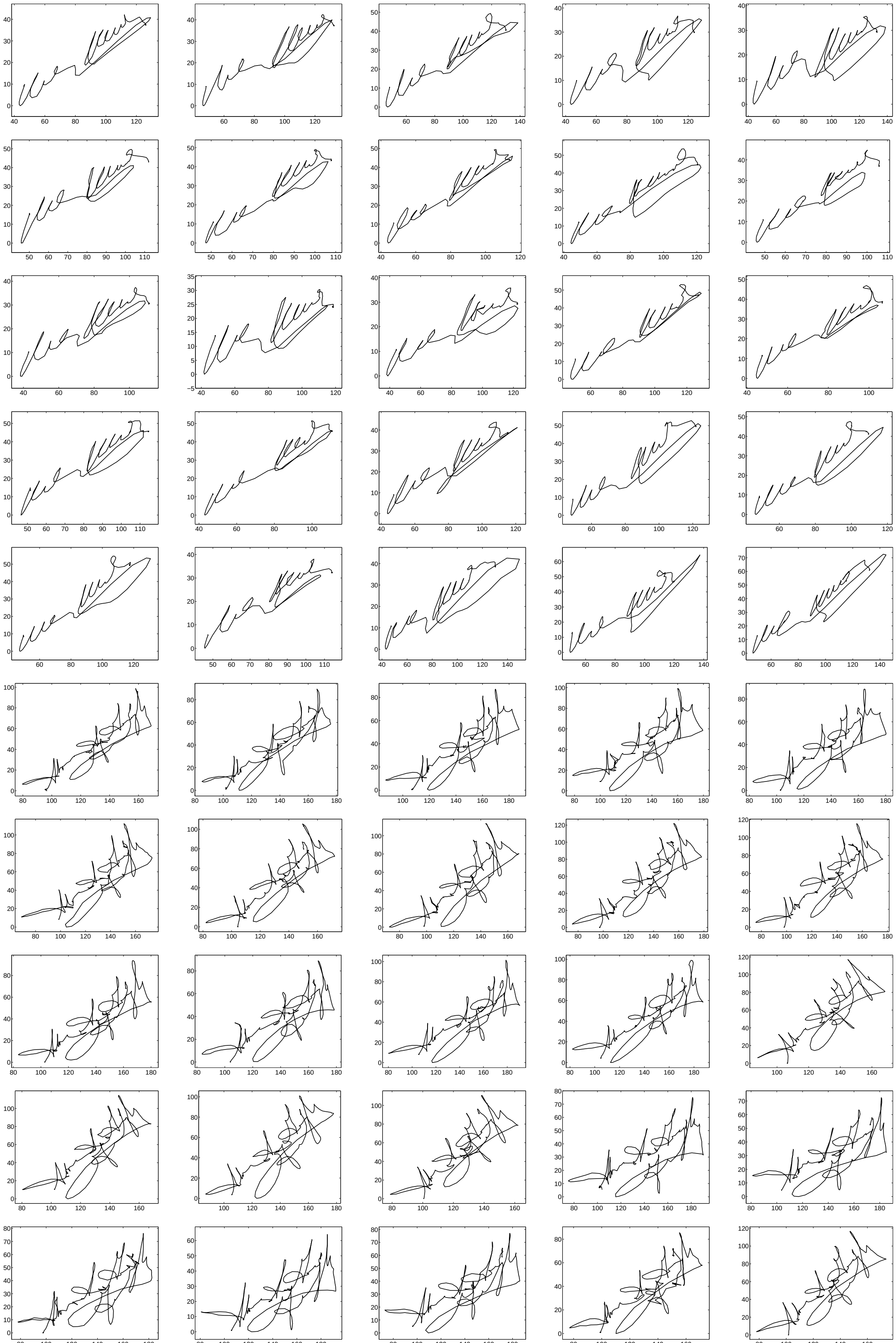


Figure 3.15: All signatures from subjects s030 and s066.

hundred milliseconds, not allowing the forger to copy the signatures at a very slow speed but rather forcing him/her to produce it at normal signing pace. The forger knew that the system was acquiring the full signing trajectory and he/she was given feedback on the success of his/her attempt. Figure 3.16 shows four signatures and one skilled forgery from eleven different subjects in the databases. The reader can make a guess on which signatures are actually forgeries. The answer is on the last page of the chapter.

3.5.2 Preprocessing

When acquiring the signatures, the position of the camera was only constrained to be such that the signatures would be captured properly and such that the signer would be comfortable. Since different signers have diverse writing manners and different pen holding style, the camera was moved from session to session. Therefore, signatures from the same subject captured in different sessions could be rotated with respect to each other. The distance defined for performing DPM is not invariant to rotation, so we need to normalize each signature with respect to rotation. In order to do so, we computed the axis of maximum and minimum inertia of the signature and then rotated the signature so that the axis of maximum inertia coincided with the horizontal axis. Experiment 1 and 2 compare the performance of the verification system with and without normalization for rotation. This normalization is not always successful since it assumes that the signatures have a clearly defined axis of maximum inertia. Figure 3.17 shows examples of a subject for which the rotation normalization works quite well and another subject (the only one in our two sets of signatures) for which the rotation normalization fails.

Subject s004 is the only one in our two sets of signatures for whom the rotation normalization fails. The individual equal error rates for this subject are 6% for intentional forgeries and 10% for random forgeries. These error rates are quite high due to the fact that several authentic signatures are rejected since the rotation normalization makes them quite different from the training set prototype. Apart from rotation

Figure 3.16: There are four true signatures and one skilled forgery in each row. Do you want to make a guess? Solutions in the last page of the chapter.

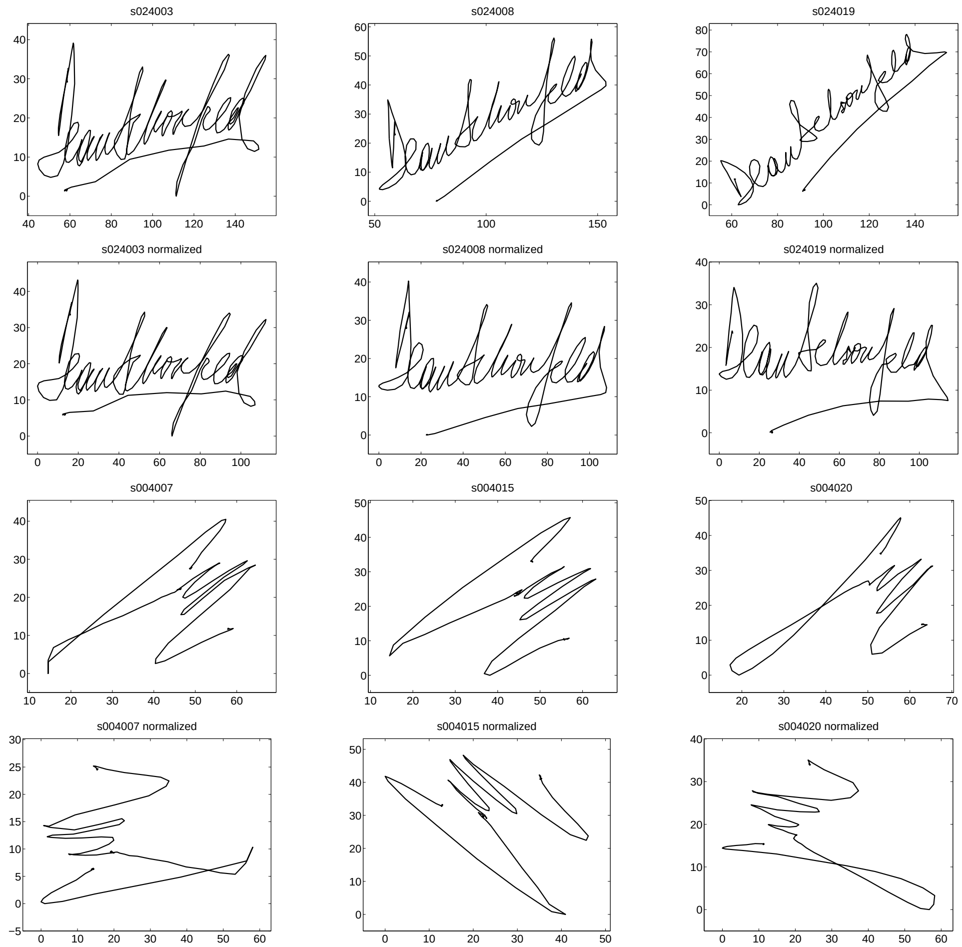


Figure 3.17: The signatures in the first and third rows are the original ones captured with the visual tracker and the signatures on the second and fourth rows are the corresponding ones after rotation normalization. The normalization works quite well for subject s024's signatures and fails for subject s004's signatures.

normalization, no other preprocessing was performed on the signatures.

3.5.3 Distance measures

The distance defined in equation 3.2 is the parameter used for classification in experiments 1, 2 and 3. However, once DPM between two signatures is performed, we have correspondence between the samples of the signatures and many other measures of similarities could be employed. In experiment 4, we describe the performance of the system using three different similarity measures that are boiled down to a single parameter by the use of the harmonic mean, as proposed by Nalwa [50]. Given two distances called d_1 and d_2, the *weighted harmonic mean* of d_1 and d_2 is defined as follows:

$$\frac{1}{d} = \frac{1}{\alpha_1 d_1} + \frac{1}{\alpha_2 d_2} \tag{3.18}$$

where α_1 and α_2 are the weighting factors. The harmonic mean is similar to performing an OR operation between the distances (assuming that a low distance corresponds to a logic 1 and a high distance corresponds to a logic 0). In practice, the harmonic mean is a simple way of "averaging" distances calculated using two models of which at least one of them is applicable, but not both models are necessarily applicable. Using the definition of weighted harmonic mean in equation 3.18, it is possible to generalize to more than two distances as $\frac{1}{d} = \sum_i \frac{1}{\alpha_i d_i}$. The weighting factors normalize the different distances to comparable values. In our system, we use the reciprocal of median absolute deviation of the corresponding distances computed in the training set as the normalizing factors.

3.5.4 Training

The time required by a biometric system to make an authentication decision could be critical for some applications. For a typical access control application, the system needs to make a decision in real-time. For forensic applications, however, the verification time may not be such a strict requirement. The authentication decision consists of determining whether a new signature belongs to the claimed class or not. This class is represented by the training set. There are two possible approaches in making this decision. One possibility is measuring the similarity between the new signature and each of the signatures in the training set and make a decision based on the set of all similarity measures. Another possible approach that is less time consuming at verification time, would be the one of extracting a reference signature from the training set and make a decision based on the similarity between the new signature and this reference one. Given the our particular application of the verification system, i.e., access control in a fully pen-based system, we chose to take the latter approach in the development of the verification algorithm.

During training the system must *learn* a representation of the training set that will yield minimum generalization error. The dynamic programming matching algorithm provides the correspondence function between two signatures, so computing the mean signature of the two original ones along the warping path provides a more robust representative for the class since the inherent noise in capturing the signatures is averaged. In the case in which there are more than two examples in the training set, there is no clear way of aligning all of them at the same time. In principle, one could think of performing the simultaneous alignment of all the examples, working on an N-dimensional tensor instead of a matrix. The disadvantage of this approach is that it is difficult to define the elementary distance associated to the arc joining two nodes of this tensor. We propose a sub-optimal training procedure. We perform only pairwise alignment in order to find all the pairwise mean signatures out of all the possible pairs of elements in the training set. The mean signature that yields minimum alignment cost with all the remaining signatures in the training set is the one chosen to perform the final matching. All signatures are placed in correspondence with this particular pairwise mean signature. The prototype that represents the training set is computed as the mean of the aligned signatures.

Given the matching constraints mentioned in previous sections, we observe that the warping path could be non-invertible. In other words, we could have many samples of one signature in the training set that are in correspondence with only one sample of the reference signature, and vice-versa. Given this particular characteristic of the matching process, it is not possible to achieve a full correspondence among all signatures in the training set, making the computation of the prototype a bit difficult. In our system, we take all the samples from all the signatures in the training set that are in correspondence with each particular sample of the reference. The mean of all these samples provides the corresponding sample of the prototype and the standard deviation of all these samples gives the weights to be used later when computing the weighted correlation measure between signatures. This particular way of extracting the prototype from the training set is not optimal since it could happen that several different samples from the same signature are used to compute one sample of the prototype, and also, it could happen that the same sample from a signature is used to compute several samples of the prototype. Figure 3.18 shows several prototypes extracted using this method. Some of these prototypes are quite noisy in part due to the particular method used to compute the prototype and in part due to the variability of the signatures in the training set as well as the quantization effect produced by the time sampling of the signatures.

The prototype and the weighting function summarizes the local statistics of the matching process among signatures in the training set. The individual residual distances between each of the signatures in the training set and the prototype signature are collected in order to estimate the global statistics of the alignment process. We extract the median and the median absolute deviation of these distances in order to use them for classification. In figure 3.18 we show several examples of signatures collected for our database, their corresponding training prototypes and one of the forgeries.

3.5.5 Testing

The error rates are evaluated first, for each individual subject and, second, for the whole dataset. Each subject's test set allows us to compute the FRR. We computed the FAR using random and skilled forgeries where the signatures from all other other subjects were used as random forgeries.

Nelson et al. [51, 52] proposed a complete set of global measurements to be used in order to discard gross forgeries before performing any matching. Only time duration is used in our system in order to do a screening of gross forgeries and speed up the experiments. The time duration of the signatures in the training set is stored along with the prototype. The time duration of all signatures under test is checked to be within three standard deviations of the mean duration for the corresponding subject

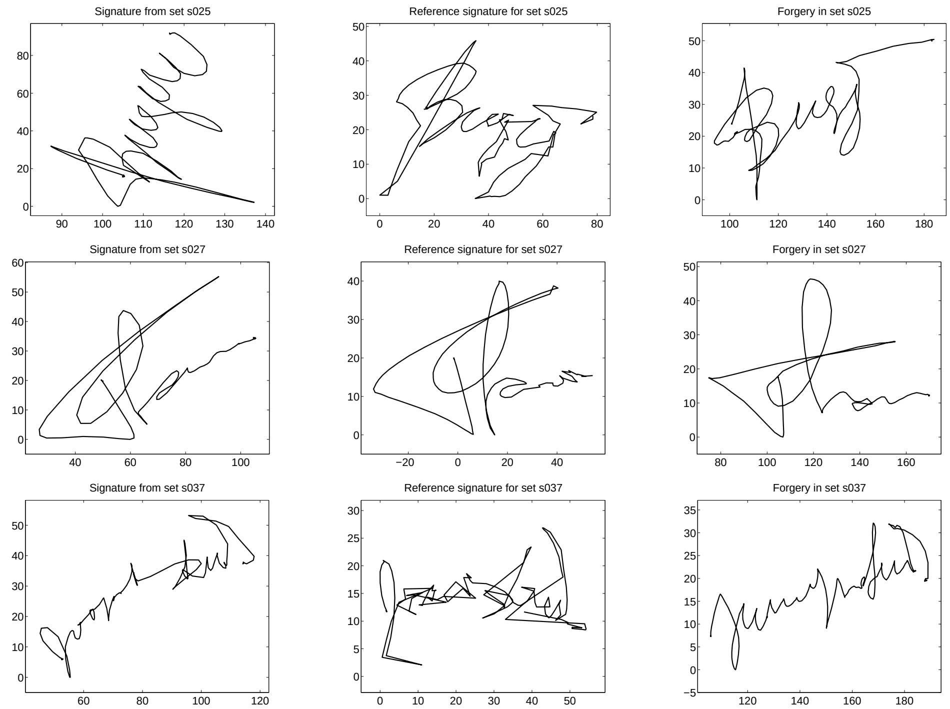


Figure 3.18: Several examples of signatures in our database. On the first column we display signatures captured with the visual tracker, on the second column we show the corresponding prototype signature, and on the third column we display one of the intentional forgeries.

(in fact, we work with robust statistics, so the median and the median absolute deviation are used instead of the mean and the standard deviation). Signatures that fall outside this bound are rejected as forgeries while signatures within bounds are matched to the prototype using DPM. This gross screening reduces the number of DPM comparisons to be performed by 40% approximately and does not generate any false rejection.

Once the matching is performed and the corresponding distances measured, the equal error rate per subject is calculated by intersecting the FAR and FRR curves, considering them to be piecewise linear. In order to show the performance of the system, we need to compute the error trade-off curves for the full set of signatures. The distances obtained per subject have to be normalized in order to obtain the full error trade-off curves. This normalization seems to be performed in the different systems described in the literature, but it is not clear how it is done. In our case, we use the global statistics per subject collected during training in order normalize the test distances. Each subject's test distances are normalized in two steps. The test distances are divided by the median absolute deviation of the subject's distances in the training set, and then the value of the threshold corresponding to the subject's equal error rate is subtracted from the result. After the normalization, all subject's test distances are merged together in order to compute the overall error trade-off curves.

As described in section 3.4.2, we use duplicate examples in order to estimate the generalization error of the algorithm. Since the distance defined for DPM is translation-invariant and the signatures are normalized for rotation, the duplicated examples are generated using only time origin shifting and affine scaling. The performance of the system using duplicate examples is shown in experiment 3.

3.5.6 Experiment 1: Performance using different parameterizations of the signature

Figure 3.19 compares the performance of the system for the different parameterizations without rotation normalization of the signatures. We only show the portion of the error trade-off curve that is most informative. The curves present a staircase pattern since the number of test examples is very small.

3.5.7 Experiment 2: Performance using different parameterizations of the signature

Figure 3.20 compares the performance of the system with rotation normalization of the signatures for the different parameterizations. The equal error rate values are smaller than the ones presented in experiment 1, i.e., the rotation normalization improved the performance of the system since most signatures have a clearly defined axis of maximum inertia.

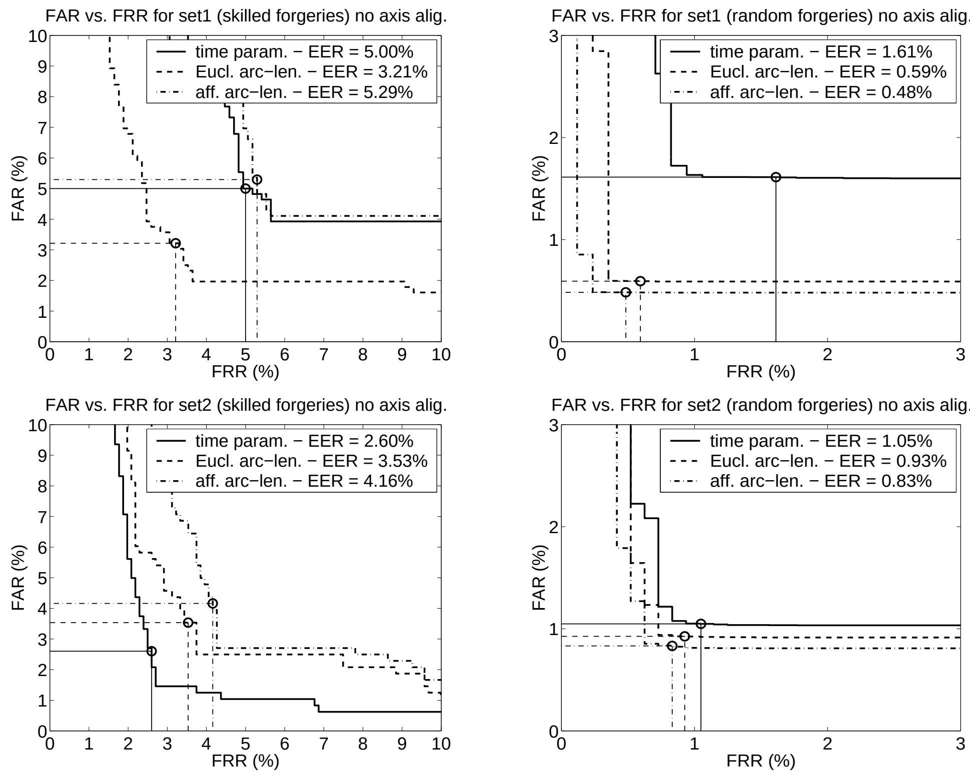


Figure 3.19: Performance of the system without rotation normalization. The first row corresponds to the first signature set and the second row corresponds to the second signature set. The circle shows the equal error rate condition.

3.5.8 Experiment 3: Performance using duplicate examples

We generated duplicate examples both for training and for computing the FRR. We used two transformations in order to produce the duplicate examples. One was time origin shifting, i.e., we re-sampled the signatures using linear interpolation, as if the time origin would have shifted from its original position to a point inside the inter-sample interval. The other was small scaling in x and y, where the range of scaling factors in each coordinate was estimated from the training examples. We generated 19 examples for each signature or forgery provided by the subjects. We did not use duplicate examples to generate random forgeries since we have enough to estimate the FAR reliably. Figure 3.21 shows the performance of the system with rotation normalization, for each of the parameterizations. We observe that the error rates are bigger than the error rates of experiment 2. This increase in the error rates is

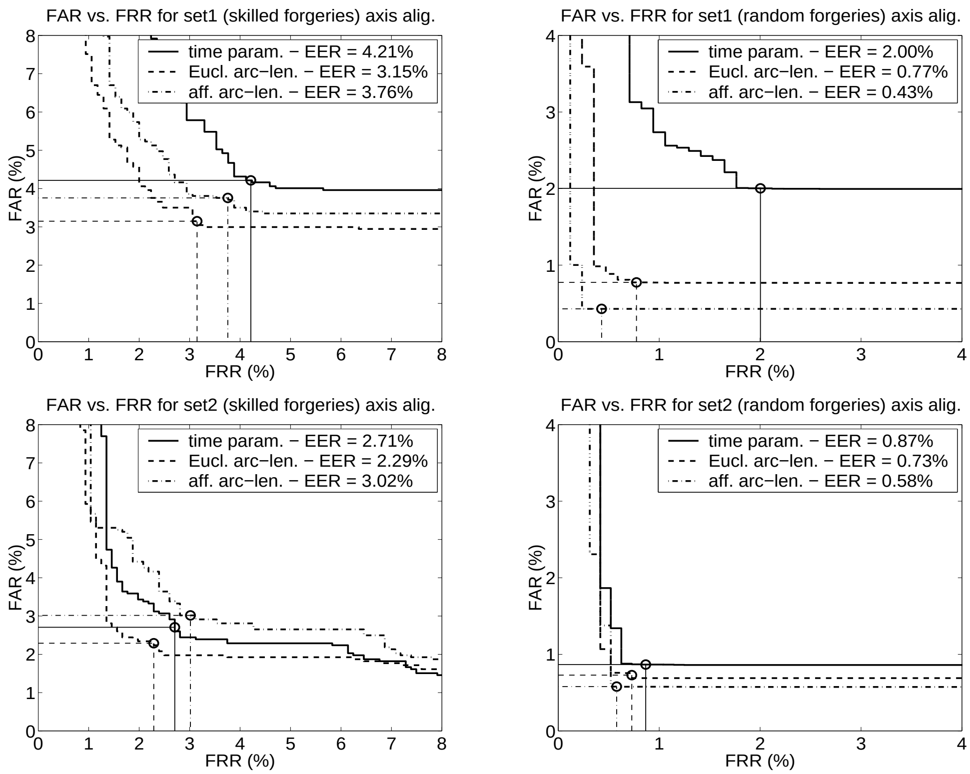


Figure 3.20: Performance of the system with rotation normalization. The first row corresponds to the first signature set and the second row corresponds to the second signature set. The circle shows the equal error rate condition.

expected since we have a bigger set of examples that provides a better statistical characterization of the problem. We should point out that the error rates are of the same order as in experiment 2, indicating that we did not introduce any wildly distorted example and that the algorithm seems to perform acceptably well.

3.5.9 Experiment 4: Performance using different distance measures

We use three different similarity measures in this experiment. The first one is the resulting distance after DPM of the prototype and the test signature. The second one is the weighted correlation between the prototype and the test signature. The weighting function represents the stability of each point of the prototype when aligned with each of the signatures in the training set, and it is computed as the reciprocal

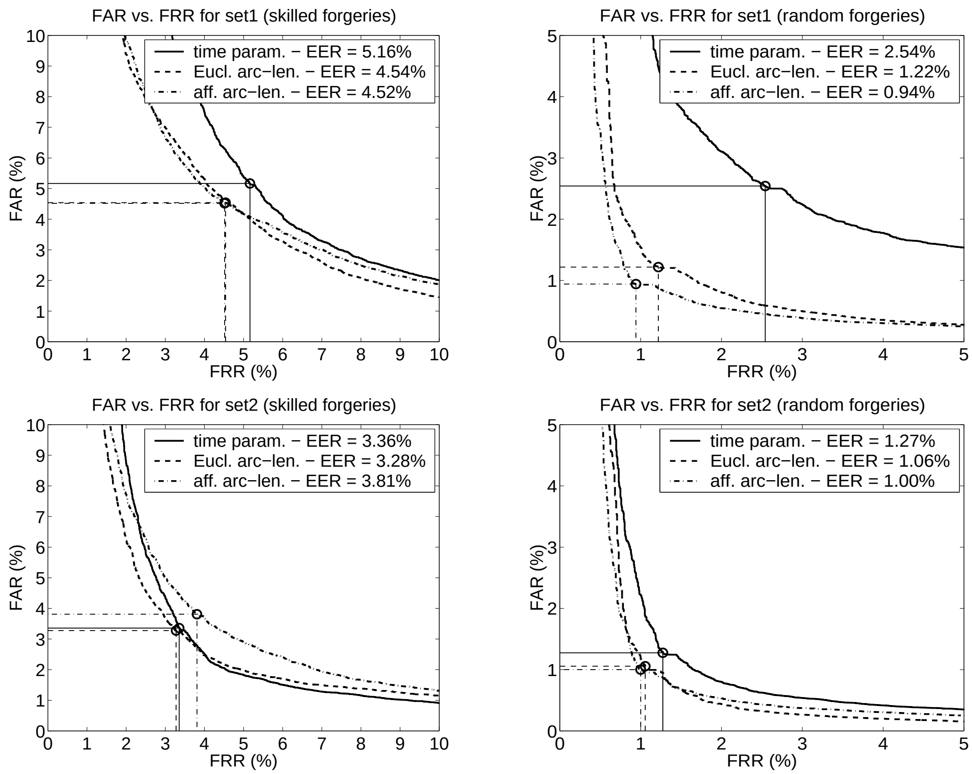


Figure 3.21: Performance of the system with duplicated examples. The first row corresponds to the first signature set and the second row corresponds to the second signature set. The circle shows the equal error rate condition.

of the standard deviation of the points of the training set that correspond to a point of the prototype. The third distance is the correlation between the prototype and the test signature after having performed a Procrustes transformation [70, 21] on them. This Procrustes transformation provides the optimal, in the least squares sense, translation, rotation and scaling between the prototype and the test signature, given the correspondence between their samples. The harmonic mean of these different distances is used as the classification parameter and the performance is shown in figure 3.22, for each of the parameterizations. We observe that the best performance is achieved by using affine arc-length parameterization of the signatures. The individual equal error rates achieved using the harmonic mean as the classification parameters is presented in figure 3.23. Given the error rates obtained with random forgeries, we could conclude that our verification method could be used for recognition since it has quite good discrimination capabilities. Figure 3.24 shows the error rate curves for each of the distances used in the experiment, for affine arc-length parameterization. We observe that the use of the harmonic mean of the various distances provides quite an improvement in performance.

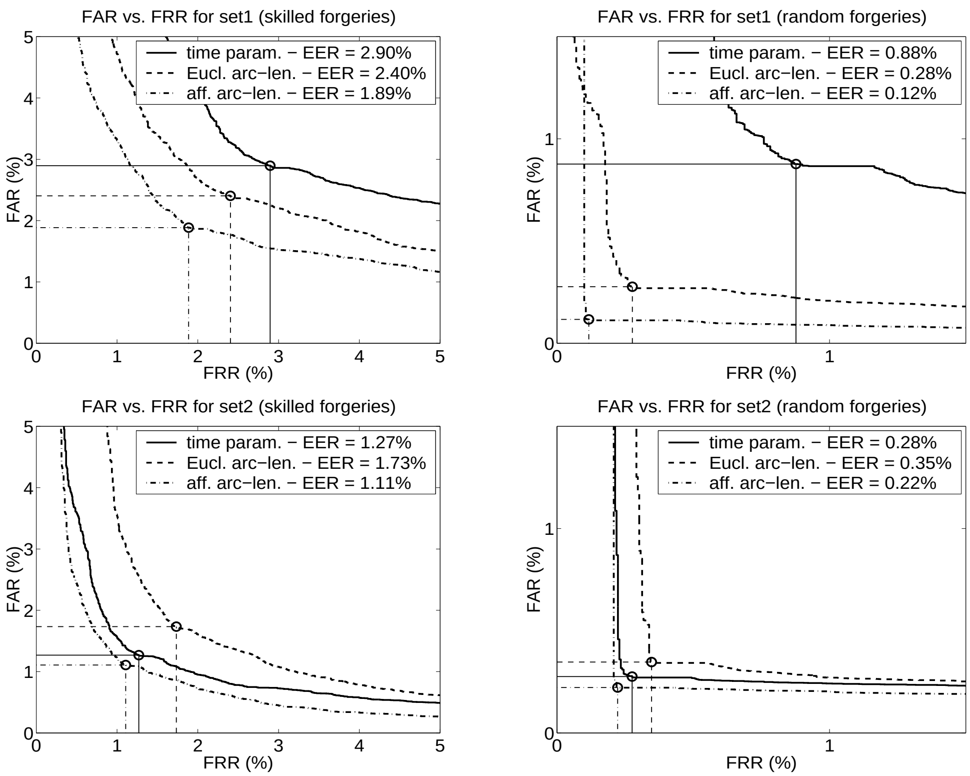


Figure 3.22: Performance of the system with multiple distances. The first row corresponds to the first signature set and the second row corresponds to the second signature set. The circle shows the equal error rate condition.

3.5.10 Discussion

The error rates presented in figure 3.22 show that the second set of forgeries have lower error rates than the first set. This difference could be due to the fact that the second set has fewer subjects than the first one, or it could also be due to less-motivated forgers in the case of set 2, or perhaps less-motivated signers in set 1. Of course, these kinds of speculations arise because we are dealing with a behavioral biometric and the only way of actually determining the performance of the system would be by

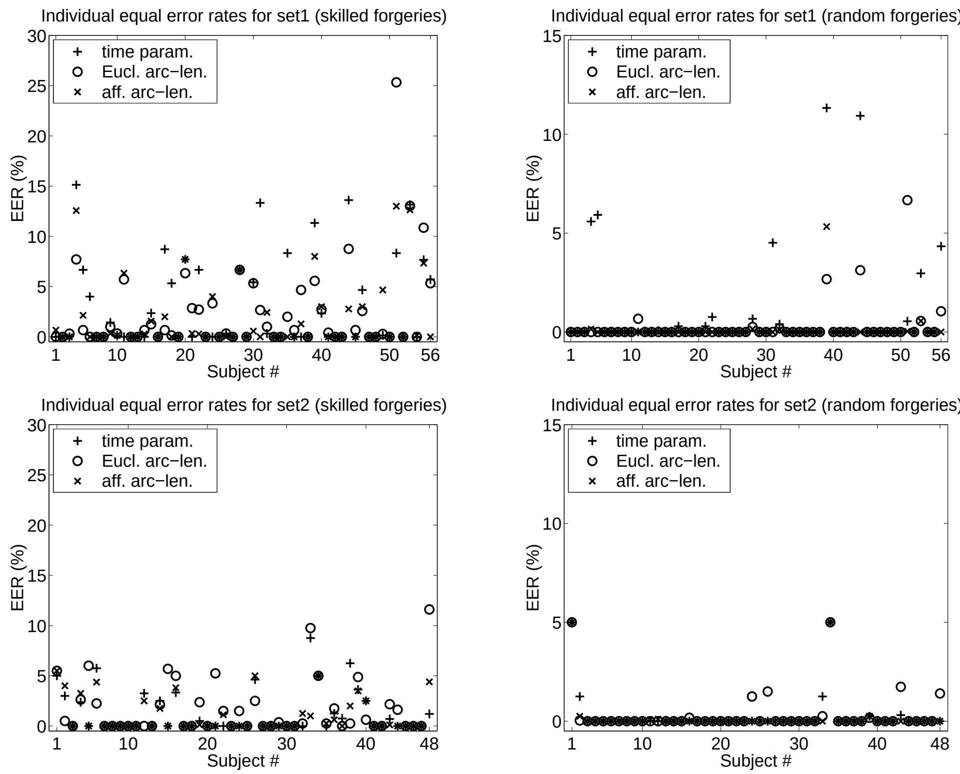


Figure 3.23: Individual equal error rates achieved using multiple distances. The first row corresponds to the first signature set and the second row corresponds to the second signature set.

running a long experiment involving hundreds, if not thousands, of subjects providing a couple of signatures per day over the course of several weeks. In any case, the error rates achieved by our system are comparable to the best performances presented in the literature [40, 41, 58, 35, 24, 34, 45, 50, 54, 65, 81].

Figure 3.25 shows examples from the two data sets for which the algorithm has an equal error rate greater than 5%. In this figure we show one of the original signatures, the prototype signature, a signature that is falsely rejected and a falsely accepted skilled forgery. We observe that higher error rates correspond to signatures that are very simple, and therefore, easy to forge.

Solution of the game of figure 3.16 There is only one forgery per row of the figure. We identify the row with a number and the column with a letter, i.e., 3e corresponds to the figure on the third row and the fifth column. The forgeries are

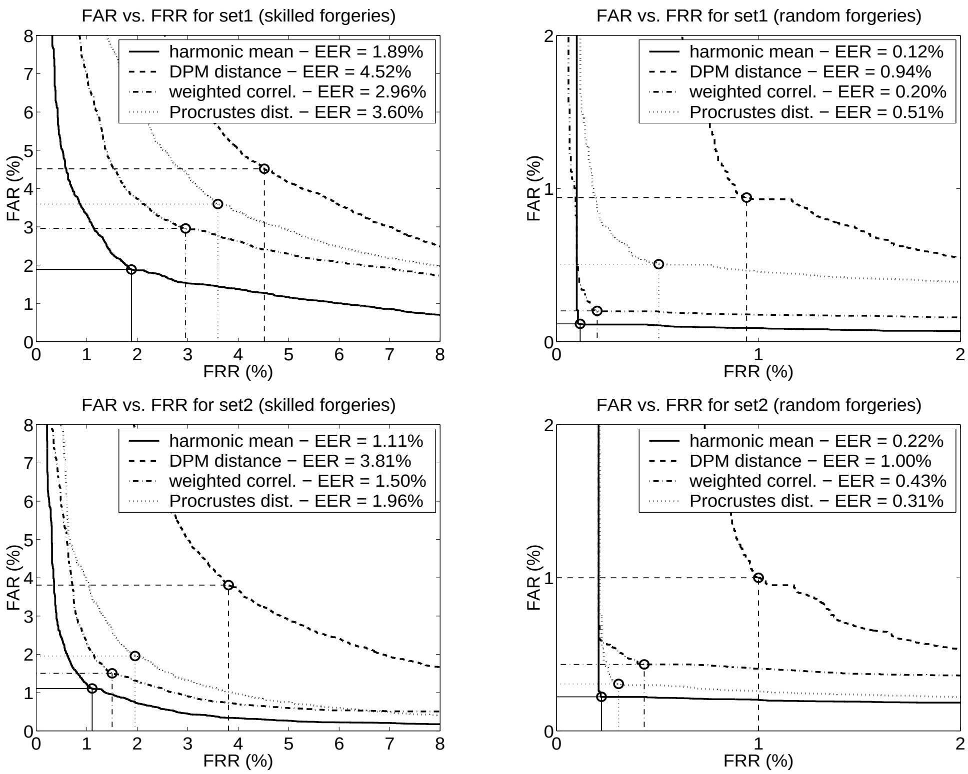


Figure 3.24: Error rate curves for the different distances used in the experiment, for affine arc-length parameterization.

located in the following position: 1c, 2e, 3a, 4b, 5a, 6d, 7c, 8e, 9c, 10a, 11c.

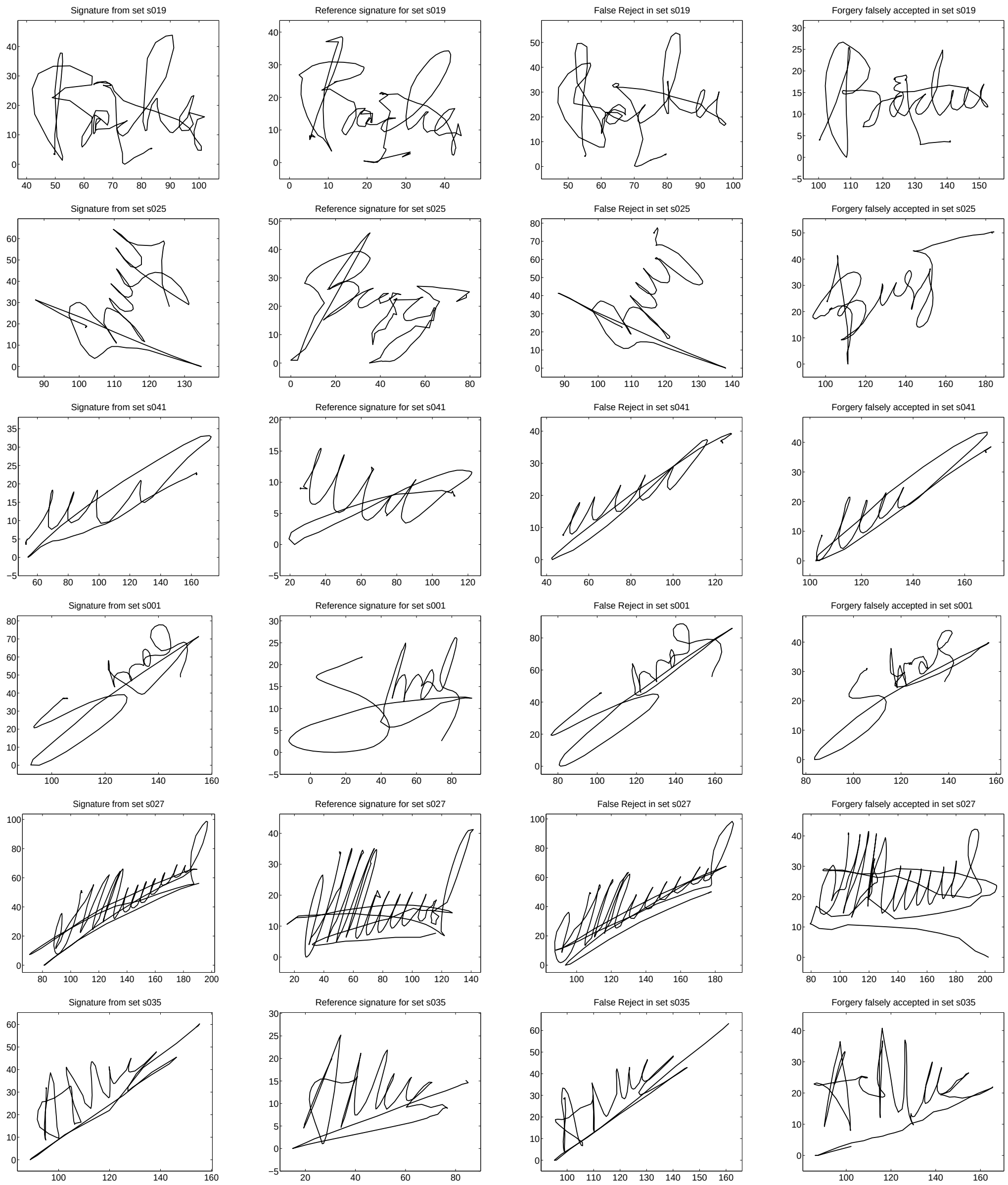


Figure 3.25: Cases for which the algorithm has biggest error. The three first sequences are from set 1 and the three last are from set 2. We show a signature from the set, the prototype signature extracted from the training set, a falsely rejected signature, and a falsely accepted skilled forgery.

Chapter 4 Subsample Curve Matching

4.1 Introduction

Chapter 3 described the application of the visual handwriting acquisition interface to the development of a signature verification system. The comparison between signatures was performed using Dynamic Programming Matching (DPM). Although this method provides reasonably good performance, it has some disadvantages. Where the sampling is sparse the matching distance is susceptible to large errors because the algorithm matches only discrete samples rather than continuous curves. One possible solution for this problem is to oversample the curves using, for example, spline interpolation before matching them. This oversampling would provide the desired resolution; however, it would also increase the computational cost of the matching proportional to the square of the oversampling factor, and it is not clear how to choose this oversampling factor in a principled way. Also, depending on the local constraints imposed on DPM, the resulting correspondence function between the curves might be non-invertible. In the case of extracting a prototype from a training set, it is desirable to put all the examples in full correspondence to extract the mean representative from the set. The correspondence map being non-invertible makes it impossible to establish full correspondence between the examples and makes it difficult to extract the prototype.

This chapter describes an algorithm based on the general method of Dynamic Programming [3] that overcomes the mentioned disadvantages of DPM by using a continuous formulation. The algorithm is allowed to find correspondence not only from sample points in one curve to sample points in the other curve but also from sample points in one curve to inter-sample points in the other and vice-versa, as shown in figure 4.1.

To our knowledge, the only existing previous work that has a similar formulation

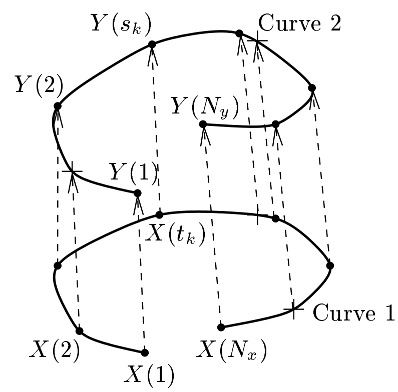


Figure 4.1: Correspondence between the two curves C_1 and C_2. The dots indicate actual samples and the crosses indicate inter-sample points.

is the one of Serra and Berthod [67, 68, 66]. They worked on matching curves or contours extracted from sequences of images or from stereo image pairs. They also proposed a continuous dynamic programming technique in order to obtain sub-pixel matching of the contours, and, therefore, better estimation of the three-dimensional structure of the scene. Our algorithm is based on a similar choice of distance between curves (see equation 3.2) as the one described in [67], but it is quite unrelated to the one presented in [68, 66]. The algorithms developed by Serra and Berthod rely on the use of several heuristic approximations to limit the complexity of the algorithm. In our case, we are able to derive several properties that exploit the structure of the problem and enable the spatial complexity of the algorithm to be decreased.

This chapter is organized as follows. Section 4.2 describes the continuous algorithm for matching planar curves and section 4.3 presents the results of experiments.

4.2 Continuous Dynamic Programming Matching

Continuous Dynamic Programming Matching (CDPM) is the continuous generalization of Dynamic Programming Matching (DPM). Figure 4.2(a) and (c) shows an example of curve matching using DPM and the corresponding matching map on the warping plane, where each sample on one of the curves is only allowed to match another sample on the other curve. The continuous generalization of DPM allows each sample point in one of the curves to match a point in-between two samples in the other curve as shown in figure 4.2(b). In other words, the warping path is allowed to go through points between the vertices of the grid as shown in figure 4.2(d). The recursion equation will be the same as the one of equation 3.6, with the condition that if t_k takes values on $\{1, \cdots, N_x\}$, then s_k is allowed to take non-integer values, and vice-versa (see figure 4.2(b)).

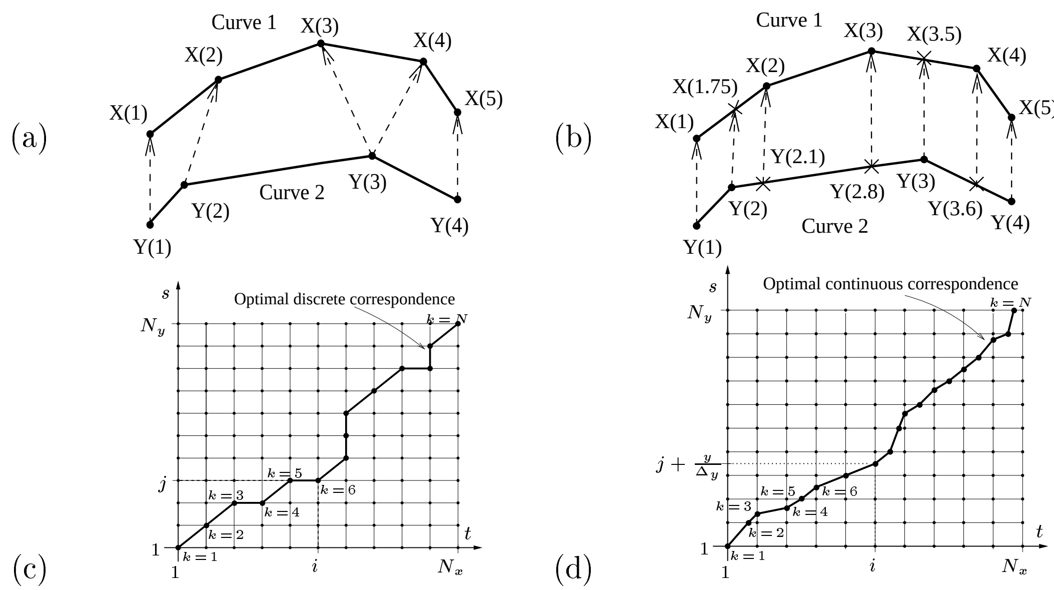


Figure 4.2: (a) Matching of two curves using DPM. (b) Matching of two curves using CDPM. The crosses show matching points that are not samples. (c) and (d) corresponding warping planes and matching functions.

The generation of these intermediate matching points assumes a particular interpolation model for the curves. We assume a linear interpolation model between sample points since it allows us to derive the correspondence equations in closed and simple form. Figure 4.3 shows the parameterization of the curves and the notation used in the derivation of CDPM. Arc-length parameterization is the most convenient way to describe the curves using the linear interpolation model. The curvilinear coordinate is denoted by x and y in correspondence with the notation X for points of C_1 and Y for points of C_2. The equations for the coordinates of points belonging to the piece-wise linear segments of C_1 and C_2 will be the following:

$$C_1: \begin{cases} u_x = u_x(i-1) + x\frac{\Delta u_{x_i}}{\Delta_{x_i}} \\ v_x = v_x(i-1) + x\frac{\Delta v_{x_i}}{\Delta_{x_i}} \end{cases} \qquad C_2: \begin{cases} u_y = u_y(j-1) + y\frac{\Delta u_{y_j}}{\Delta_{y_j}} \\ v_y = v_y(j-1) + y\frac{\Delta v_{y_j}}{\Delta_{y_j}} \end{cases}$$

$$x \in [0, \Delta_{x_i}] \qquad y \in [0, \Delta_{y_j}]$$

$$\Delta u_{x_i} = u_x(i) - u_x(i-1) \qquad \Delta u_{y_j} = u_y(j) - u_y(j-1)$$

$$\Delta v_{x_i} = v_x(i) - v_x(i-1) \qquad \Delta v_{y_j} = v_y(j) - v_y(j-1)$$

$$\Delta_{x_i} = \sqrt{\Delta u_{x_i}{}^2 + \Delta v_{x_i}{}^2} \qquad \Delta_{y_j} = \sqrt{\Delta u_{y_j}{}^2 + \Delta v_{y_j}{}^2}$$

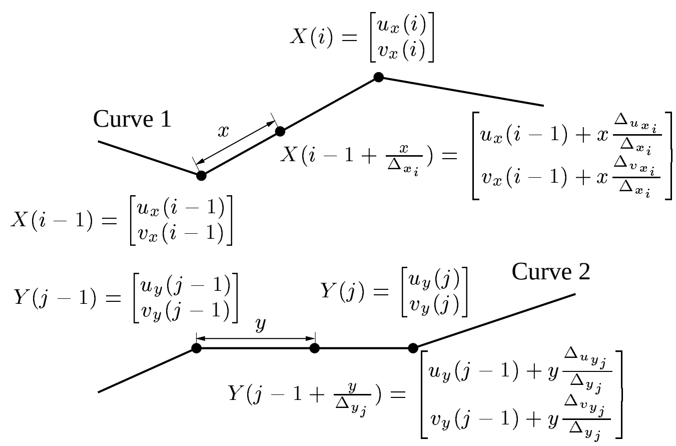


Figure 4.3: Curve parameterization used in CDPM.

4.2.1 Analysis of a single step of the CDPM recursion

Due to the recursive nature of the dynamic programming method, a single step of the algorithm is the basic building block to be studied. This single step involves the matching between two segments, one from each curve, and it is governed by the local continuity constraints imposed onto the minimization of equation 3.6. As mentioned before, the samples on each of the curves impose a grid onto the warping plane as shown in figure 4.2(c)-(d). This single step corresponds to a segment of the warping path joining two sides of one of the squares of the grid. Figure 4.4(a) shows the local constraints used for DPM (see figure 3.8 and section 3.2 for a detailed explanation) and figure 4.4(b) displays the corresponding generalization of the local continuity constraints for CDPM. We identify each square of the grid with the coordinates of its upper-right corner, e.g., the squares shown in figure 4.4 (a)-(b) are identified with node (i,j).

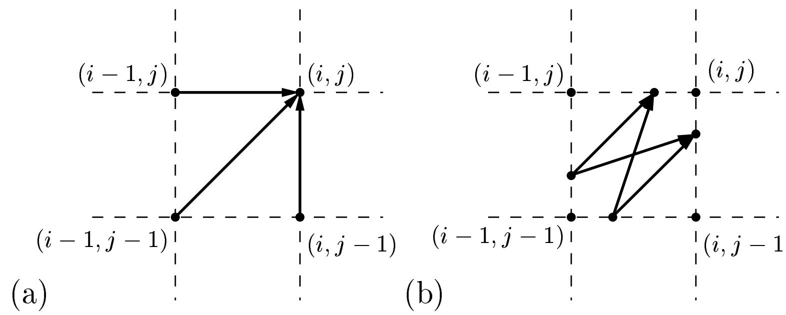


Figure 4.4: (a) Local continuity constraints imposed onto DPM. (b) Generalization of the constraints for CDPM. Note that in (a) only samples are allowed to match while in (b) samples on one curve can be matched to inter-sample points on the other curve and vice-versa.

Consider the four possible matching cases of figure 4.4(b). Assume the sides of the square corresponding to points of coordinates $(i, j-1+\frac{y}{\Delta_{y_j}})$ and $(i-1+\frac{x}{\Delta_{x_i}}, j)$ to be the "output" sides of the square and the sides corresponding to points of coordinates $(i-1, j-1+\frac{y'}{\Delta_{y_j}})$ and $(i-1+\frac{x'}{\Delta_{x_i}}, j-1)$ to be the "input" sides of the square (this assumption makes sense since we explore the warping from left to right and from bottom to top) as shown in figure 4.5. We use a quote $'$ on the curvilinear variables x, y to differentiate between "input" and "output" sides of the square.

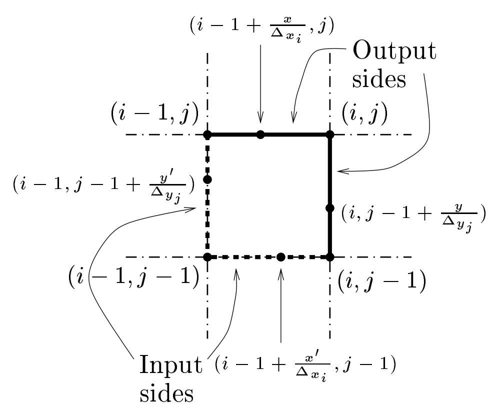


Figure 4.5: Notation used to identify points on one of the squares of the grid of the warping plane.

In section 3.2.2 we defined the elementary distance $d((t_{k-1}, s_{k-1}), (t_k, s_k))$ of having

$\mathbf{X}(t_k)$ in correspondence with $\mathbf{Y}(s_k)$ and $\mathbf{X}(t_{k-1})$ in correspondence with $\mathbf{Y}(s_{k-1})$ as

$$d((t_{k-1}, s_{k-1}), (t_k, s_k)) = \|\overrightarrow{\mathbf{X}(t_k)\mathbf{Y}(s_k)} - \overrightarrow{\mathbf{X}(t_{k-1})\mathbf{Y}(s_{k-1})}\|^2$$

where $\|.\|^2$ is the Euclidean norm. This elementary distance can be extended in a straightforward way to the case of inter-sampling matching once the correspondence between pairs of points has been established.

The four possible correspondence cases are depicted on figure 4.6. The first column of the figure exhibits portions of the matching path in the warping plane. The second column of the figure displays the correspondence between sample and inter-sample points. The calculation of the elementary distance $d((t_{k-1}, s_{k-1}), (t_k, s_k))$ is demonstrated on the third column of the figure, where $X(t_k)$ and $Y(s_k)$ are superimposed in order to provide the geometrical interpretation of the distance, that is represented by the dashed line.

The distance $d((t_{k-1}, s_{k-1}), (t_k, s_k))$ is calculated using the cosine law as follows:

$$\begin{array}{ll}
\text{Case 1:} & d((i-1+\frac{x'}{\Delta_{x_i}}, j-1), (i, j-1+\frac{y}{\Delta_{y_j}})) = y^2 + (\Delta_{x_i} - x')^2 - 2y(\Delta_{x_i} - x')\cos\theta_{ij} \\
\text{Case 2:} & d((i-1, j-1+\frac{y'}{\Delta_{y_j}}), (i-1+\frac{x}{\Delta_{x_i}}, j)) = x^2 + (\Delta_{y_j} - y')^2 - 2x(\Delta_{y_j} - y')\cos\theta_{ij} \\
\text{Case 3:} & d((i-1+\frac{x'}{\Delta_{x_i}}, j-1), (i-1+\frac{x}{\Delta_{x_i}}, j)) = \Delta_{y_j}^2 + (x - x')^2 - 2\Delta_{y_j}(x - x')\cos\theta_{ij} \\
\text{Case 4:} & d((i-1, j-1+\frac{y'}{\Delta_{y_j}}), (i, j-1+\frac{y}{\Delta_{y_j}})) = \Delta_{x_i}^2 + (y - y')^2 - 2\Delta_{x_i}(y - y')\cos\theta_{ij}
\end{array} \tag{4.1}$$

where $\cos\theta_{ij}$ is the cosine of the angle defined by the vectors $\overrightarrow{X(i-1)X(i)}$ and $\overrightarrow{Y(j-1)Y(j)}$ as shown in figure 4.6. We see that case 2 is the dual of case 1, and case 3 is the dual of case 4, so all the properties presented in the following section are going to be demonstrated only for cases 1 and 3. The use of a linear interpolation in the curves gives rise to elementary distances that are quadratic in the variables of interest x and y.

In section 3.2, we presented the recursion equation 3.6 that is used in order to

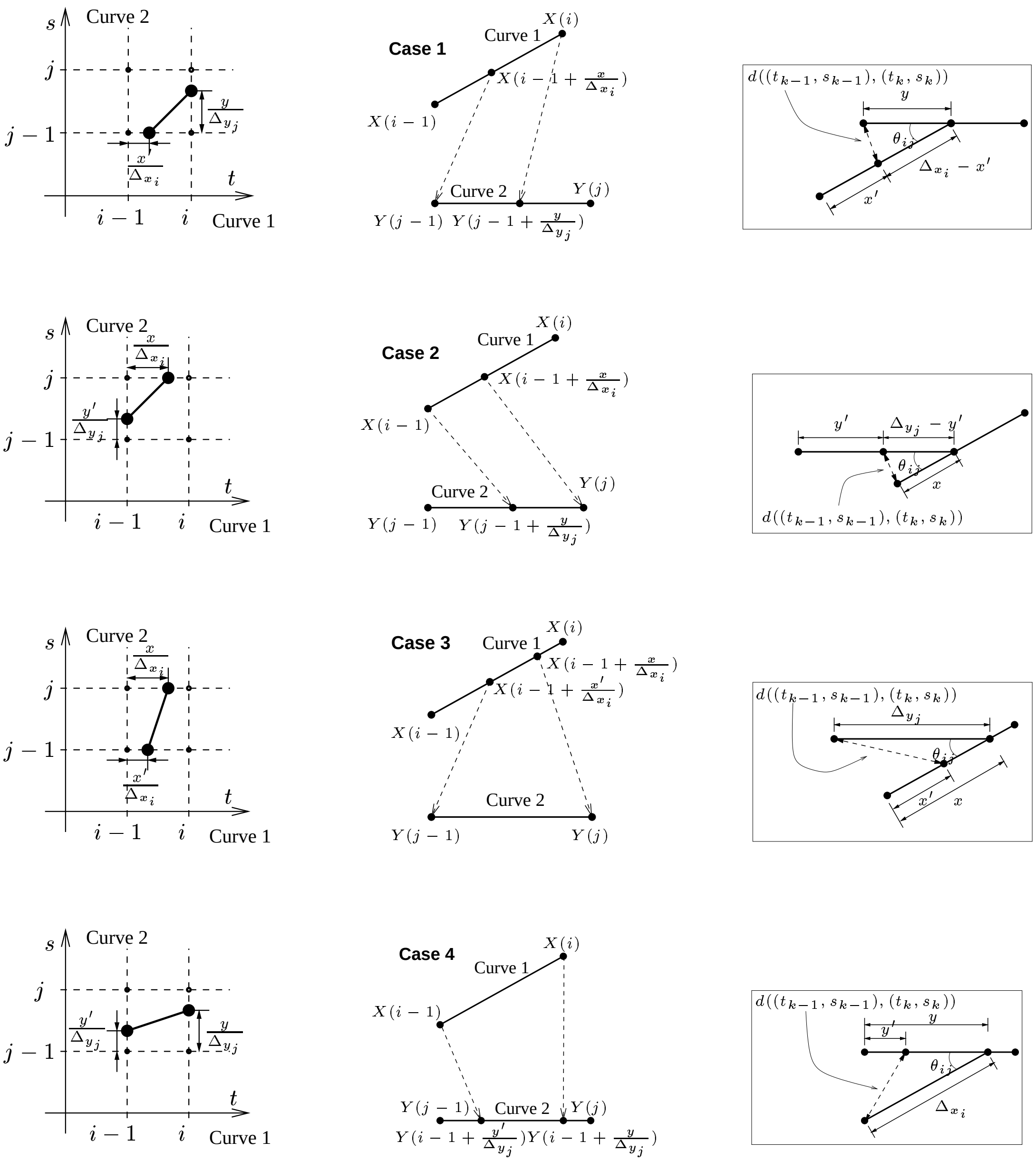


Figure 4.6: Different matching cases that are possible at each step of the algorithm. The first column shows the segments on the warping plane. The second column displays the point correspondence. The third column demonstrates the calculation of the elementary distance $d((t_{k-1}, s_{k-1}), (t_k, s_k))$ using the cosine law.

find the optimal matching with DPM

$$\mathcal{D}(n_x,n_y)=\min_{(n'_x,n'_y)}\left\{\mathcal{D}(n'_x,n'_y)+\xi((n'_x,n'_y),(n_x,n_y))\right\}$$

where (n_x,n_y) and (n'_x,n'_y) were nodes of the grid on the warping plane and $\xi((n'_x,n'_y),(n_x,n_y))$ was defined by the local continuity constraints. This equation is still valid as the recursion for CDPM by an adequate choice of (n_x,n_y), (n'_x,n'_y), and $\xi((n'_x,n'_y),(n_x,n_y))$ and a proper arrangement of the minimization. (n_x,n_y) and (n'_x,n'_y) respectively correspond to coordinates of points on the input and output sides of the square of the grid under consideration. Given the local continuity constraints for CDPM that are shown in figure 4.4(b), $\xi((n'_x,n'_y),(n_x,n_y))$ consists of only one segment that joins two sides of the square of the grid on the warping plane and the value of ξ is given by equation 4.1. For example, for case 1, $\xi((n'_x,n'_y),(n_x,n_y))=\xi((i-1+\frac{x'}{\Delta_{x_i}},j-1),(i,j-1+\frac{y}{\Delta_{y_j}}))=d((i-1+\frac{x'}{\Delta_{x_i}},j-1),(i,j-1+\frac{y}{\Delta_{y_j}}))$ and the recursion equation takes the following form (see the first row of figure 4.6):

$$\mathcal{D}_{(i,j)}(y)=\min_{x'}\left\{\mathcal{D}_{(i-1,j-1)}(x')+d((i-1+\frac{x'}{\Delta_{x_i}},j-1),(i,j-1+\frac{y}{\Delta_{y_j}}))\right\}\quad(4.2)$$

where the $\mathcal{D}_{(i,j)}(y)$ is now a function of a continuous variable y and the minimization is performed with respect to another continuous variable x'. Depending on the correspondence case under analysis, the cumulated distance is noted as $\mathcal{D}_{(i,j)}(y)$ or $\mathcal{D}_{(i,j)}(x)$, where the curvilinear coordinates x,y point out the output side of the square of the grid that $\mathcal{D}_{(i,j)}(\cdot)$ refers to. The index (i,j) indicates the points in the two curves up to which the cumulated distance is stored in $\mathcal{D}_{(i,j)}(\cdot)$. We see that, at each step in the algorithm, the cumulated distance function $\mathcal{D}_{(\cdot,\cdot)}(\cdot)$ is a function of a continuous variable instead of a single number, as we had in DPM. This continuous function will propagate from one step to the following via the recursion equation. Given the simple expression for the elementary distances shown in equation 4.1, it is easy to show using mathematical induction that the cumulated distance function $\mathcal{D}_{(\cdot,\cdot)}(\cdot)$ is a quadratic function of the continuous curvilinear coordinates x, y.

Property 4.2.1 *The cumulated distance $\mathcal{D}_{(\cdot,\cdot)}(\cdot)$ is a quadratic function of the curvilinear coordinate of interest at every step of the algorithm.*

Proof: We develop the proof for cases 1 and 3 separately. We use induction.
Case 1: The first step of the induction corresponds to the matching between the first segments of each curve. Looking at the first row of figure 4.6 and considering $i = j = 2$, the recursion equation 4.2 takes the following form:

$$\begin{aligned} \mathcal{D}_{(2,2)}(y) = & \min_{x'} \left\{ d((1+\frac{x'}{\Delta_{x_2}}, 1), (2, 1+\frac{y}{\Delta_{y_2}})) \right\} \\ = & \min_{x'} \left\{ y^2 + (\Delta_{x_2} - x')^2 - 2y(\Delta_{x_2} - x') \cos\theta_{22} \right\} \end{aligned} \tag{4.3}$$

Taking the first derivative of $d((1+\frac{x'}{\Delta_{x_2}}, 1), (2, 1+\frac{y}{\Delta_{y_2}}))$ with respect to x' and equating it to zero, the solution of the minimization is the following:

$$\begin{aligned} x' &= \Delta_{x_2} - y\cos\theta_{22} \\ \mathcal{D}_{(2,2)}(y) &= y^2(1-\cos^2\theta_{22}) \end{aligned} \tag{4.4}$$

In fact, the optimal matching value x' for a given value of y is just defined by the projection of $Y(1)$ onto the segment $\overline{X(1)X(2)}$ once $X(2)$ and $Y(s_1 = 1 + \frac{y}{\Delta_{y_2}})$ are superimposed. The cumulated distance $\mathcal{D}_{(2,2)}(y)$ is the distance from $Y(1)$ to the segment $\overline{X(1)X(2)}$ and it is a quadratic function of the variable y, as shown in figure 4.7.

Assuming that the cumulated distance for $i \geq 1, j-1 \geq 1$ is a quadratic function of x', say $\mathcal{D}_{(i,j-1)}(x') = A_{x'}x'^2 + 2B_{x'}x' + C_{x'}$, then the cumulated distance for (i, j) is computed as follows:

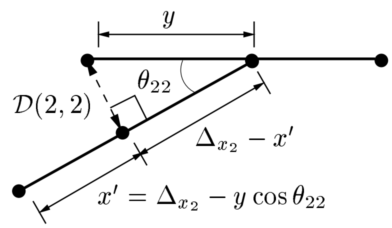


Figure 4.7: Optimal correspondence for the first match with case 1.

$$\begin{aligned}
\mathcal{D}_{(i,j)}(y) &= \min_{x'} \left\{ (A_{x'}{x'}^2 + 2B_{x'}x' + C_{x'}) + y^2 + (\Delta_{x_i} - x')^2 - 2y(\Delta_{x_i} - x')\cos\theta_{ij} \right\} \\
&= \min_{x'} \left\{ (A_{x'}+1){x'}^2 + 2(B_{x'} - \Delta_{x_i} + y\cos\theta_{ij})x' + (C_{x'} + \Delta_{x_i}^2 - 2\Delta_{x_i} y\cos\theta_{ij} + y^2) \right\} \\
&= \tfrac{A_{x'}+1-\cos^2\theta_{ij}}{A_{x'}+1} y^2 - \tfrac{(A_{x'}\Delta_{x_i}+B_{x'})\cos\theta_{ij}}{A_{x'}+1} y + (C_{x'} + \Delta_{x_i}^2 - \tfrac{(B_{x'}-\Delta_{x_i})^2}{A_{x'}+1}) \\
&= A_y y^2 + 2B_y y + C_y
\end{aligned}$$

$$\begin{aligned}
A_y &= \frac{A_{x'} + 1 - \cos^2\theta_{ij}}{A_{x'}+1} & B_y &= -\frac{(A_{x'}\Delta_{x_i} + B_{x'})\cos\theta_{ij}}{A_{x'}+1} \\
C_y &= C_{x'} + \Delta_{x_i}^2 - \frac{(B_{x'} - \Delta_{x_i})^2}{A_{x'}+1} & x' &= \frac{-y\cos\theta_{ij} + (\Delta_{x_i} - B_{x'})}{A_{x'}+1}
\end{aligned} \tag{4.5}$$

Having a sequence of elementary matches that consist of only correspondences with case 3 (or 4), the cumulated distance is a constant as will be shown in equation 4.9. Assuming that the cumulated distance for $i \geq 1, j-1 \geq 1$ is a constant function of x', say $\mathcal{D}_{(i,j-1)}(x') = C_{x'}$, then the cumulated distance for (i,j) is computed as follows:

$$\begin{aligned}
\mathcal{D}_{(i,j)}(y) &= \min_{x'} \left\{ C_{x'} + y^2 + (\Delta_{x_i} - x')^2 - 2y(\Delta_{x_i} - x')\cos\theta_{ij} \right\} \\
&= \min_{x'} \left\{ x'^2 + 2(y\cos\theta_{ij} - \Delta_{x_i})x' + (C_{x'} + \Delta_{x_i}^2 - 2\Delta_{x_i} y\cos\theta_{ij} + y^2) \right\} \\
&= (1 - \cos^2\theta_{ij})y^2 + C_{x'} \\
&= A_y y^2 + 2B_y y + C_y
\end{aligned}$$

$$\begin{array}{ll}
A_y = 1 - \cos^2\theta_{ij} & B_y = 0 \\
C_y = C_{x'} & x' = -y\cos\theta_{ij} + \Delta_{x_i}
\end{array} \tag{4.6}$$

Therefore, the cumulated distance $\mathcal{D}_{(i,j)}(y)$ is a quadratic function of the variable y.

Case 3: The first step of the induction corresponds to the matching between the first segments of each curve. Looking at the third row of figure 4.6, the recursion equation 3.6 takes the following form:

$$\begin{aligned}
\mathcal{D}_{(2,2)}(x) &= \min_{x'} \left\{ d((1 + \frac{x'}{\Delta_{x_2}}, 1), (1 + \frac{x}{\Delta_{x_2}}, 2)) \right\} \\
&= \min_{x'} \left\{ \Delta_{y_2}^2 + (x - x')^2 - 2\Delta_{y_2}(x - x')\cos\theta_{22} \right\}
\end{aligned} \tag{4.7}$$

Taking the first derivative of $d((1+\frac{x'}{\Delta_{x_2}}, 1), (1+\frac{x}{\Delta_{x_2}}, 2))$ with respect to x' and equating it to zero, the solution of the minimization is the following:

$$\begin{aligned}
x' &= x - \Delta_{y_2}\cos\theta_{22} \\
\mathcal{D}_{(2,2)}(x) &= \Delta_{y_2}^2(1 - \cos^2\theta_{22})
\end{aligned} \tag{4.8}$$

In fact, the optimal matching value x' for a given value of x is just defined by the projection of $Y(1)$ onto the segment $\overline{X(1)X(2)}$ once $Y(2)$ and $X(t_1 = 1 + \frac{x}{\Delta_{x_2}})$ are superimposed. The cumulated distance $\mathcal{D}_{(2,2)}(x)$ is the distance from $Y(1)$ to the

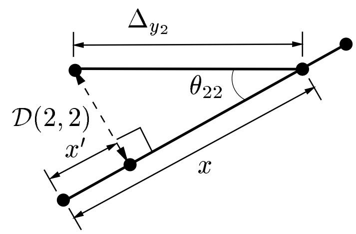


Figure 4.8: Optimal correspondence for the first match with case 3.

segment $\overline{X(1)X(2)}$, as shown in figure 4.8, and it is a constant function of the variable x; in fact, it is a degenerated quadratic function of the variable x.

Assuming that the cumulated distance for $i \geq 1, j-1 \geq 1$ is a constant function of x', say $\mathcal{D}_{(i,j-1)}(x') = C_{x'}$, then the cumulated distance at step $k+1$ is computed as follows:

$$\begin{aligned}
\mathcal{D}_{(i,j)}(x) &= \min_{x'} \left\{ C_{x'} + \Delta_{y_j}^2 + (x-x')^2 - 2\Delta_{y_j}(x-x')\cos\theta_{ij} \right\} \\
&= \min_{x'} \left\{ x'^2 + 2(\Delta_{y_j}\cos\theta_{ij} - x)x' + (C_{x'} + \Delta_{y_j}^2 - 2\Delta_{y_j}x'\cos\theta_{ij} + x'^2) \right\} \\
&= C_{x'} + \Delta_{y_j}^2(1-\cos^2\theta)
\end{aligned}$$

$$\begin{aligned}
&C_x = C_{x'} + \Delta_{y_j}^2(1-\cos^2\theta) \\
&x' = x - \Delta_{y_j}\cos\theta_{ij}
\end{aligned} \tag{4.9}$$

Given that the cumulated distance for $i \geq 1, j-1 \geq 1$ is a constant, then the cumulated distance for (i,j) is also a constant. This situation arises by a sequence of elementary matches consisting of only correspondences with case 3 (or 4). We have shown in equations 4.5 and 4.6 that the cumulated distance after a correspondence with case 1 (or 2) is a quadratic function of the curvilinear variable x (or y). Therefore, in the general setting we have to assume that the cumulated distance for $i \geq 1, j-1 \geq 1$ is a quadratic function of x', say $\mathcal{D}_{(i,j-1)}(x') = A_{x'}x'^2 + 2B_{x'}x' + C_{x'}$, and compute the cumulated distance for (i,j)

$$\begin{aligned}
\mathcal{D}_{(i,j)}(x) &= \min_{x'} \left\{ (A_{x'}x'^2 + 2B_{x'}x' + C_{x'}) + \Delta_{y_j}^2 + (x-x')^2 - 2\Delta_{y_j}(x-x')\cos\theta_{ij} \right\} \\
&= \min_{x'} \left\{ (A_{x'}+1)x'^2 + 2(B_{x'} + \Delta_2\cos\theta_{ij} - x)x' + (C_{x'} + \Delta_{y_j}^2 - 2\Delta_2 x\cos\theta_{ij} + x^2) \right\} \\
&= \tfrac{A_{x'}}{A_{x'}+1}x^2 + \tfrac{B_{x'} - A_{x'}\Delta_{y_j}\cos\theta_{ij}}{A_{x'}+1}x + (C_{x'} + \Delta_{y_j}^2 - \tfrac{(B_{x'}+\Delta_{y_j}\cos\theta_{ij})^2}{A_{x'}+1}) \\
&= A_x x^2 + 2B_x x + C_x
\end{aligned}$$

$$\begin{array}{ll}
A_x = \dfrac{A_{x'}}{A_{x'}+1} & B_x = \dfrac{B_{x'} - A_{x'}\Delta_{y_j}\cos\theta_{ij}}{A_{x'}+1} \\
C_x = C_{x'} + \Delta_{y_j}^2 - \dfrac{(B_{x'} + \Delta_{y_j}\cos\theta_{ij})^2}{A_{x'}+1} & x' = \dfrac{x - \Delta_{y_j}\cos\theta_{ij} - B_{x'}}{A_{x'}+1}
\end{array} \tag{4.10}$$

Therefore, the cumulated distance $\mathcal{D}_{(i,j)}(x)$ is a quadratic function of the variable x. The induction is complete. ∎

4.2.2 Analysis of the CDPM algorithm

The above equations show how the cumulated distance functions propagate through the warping plane. Given the four possible correspondence cases at each step of the recursion, it is easy to see that there are a number of cumulated distance functions that are propagating through the warping plane. Each of these functions corresponds to one possible series of elementary matchings. These cumulated distance functions compete in order to provide the minimum distance between the curves. This competition is materialized in the final step of the recursion where the minimum value of all these functions is obtained, providing the minimum distance between the curves. The corresponding matching function is given by the back-propagation of the value of the curvilinear coordinate that provides the minimum. This back-propagation through the warping plane is very simple since at each step of the algorithm, there is a linear relationship between the corresponding curvilinear coordinates x, y and x', y'. We showed that the cumulated distance is either a constant or a quadratic function of the curvilinear coordinate x, y. The situation in which the distance is a constant occurs only for a very particular warping function that consists only of elementary matches with cases 3 or 4. Given that the distance is a constant at one particular iteration, we showed in equations 4.9 and 4.10 that the distance would continue being a constant or it would become a full quadratic function. Also, given that the distance is a quadratic function, we showed in equations 4.5 and 4.6 that the distance would continue being a quadratic function.

The CDPM algorithm performs single step matching at each node of the grid on the warping plane, proceeding from left to right and from bottom to top. After reaching the last node of the grid, the minimum distance is computed. There is also a final backtracking step in order to find the point correspondence that provides the minimum distance. As presented in section 3.2, only part of the warping plane is explored in order to find the correspondence function that minimizes the distance between curves. The allowed region on the warping plane is defined by global constraints. These constraints limit the number of nodes that are evaluated during the recursion, decreasing the computational complexity of the algorithm from $o(N_x N_y)$ to $o(\#\text{of nodes inside the allowed region})$.

In order to summarize the CDPM algorithm, we need a few auxiliary variables in order to recall the information required for the backtracking step. At each node (i, j) of the grid, we have a number of parabolas $\mathcal{D}_{(i,j)}(y)$ and $\mathcal{D}_{(i,j)}(x)$, where the curvilinear variables x and y specify the output side of the square of the grid that the parabola corresponds to. We need to recall the parent parabola for each parabola $\mathcal{D}_{(i,j)}(y)$ and $\mathcal{D}_{(i,j)}(x)$ and also the side of the square to which the parent belongs. Let us store the parent index of parabola k at square (i, j) in $\zeta^{x,y}_{(i,j)}(k)$ and the parent's side on $\psi^{x,y}_{(i,j)}(k)$, where the super-index x or y indicates the correspondence of auxiliary variables with $\mathcal{D}_{(i,j)}(\cdot)$. The CDPM algorithm is as follows:

1 Initialization:

(a) Compute the two parabolas $\mathcal{D}_{(2,2)}(y)$ corresponding to the first iteration of cases 1 and 4. $\zeta^{y}_{(2,2)}(1) = \zeta^{y}_{(2,2)}(2) = 1$, $\psi^{y}_{(2,2)}(1) = x$, and $\psi^{y}_{(2,2)}(2) = y$.

(b) Compute the two parabolas $\mathcal{D}_{(2,2)}(x)$ corresponding to the first iteration of cases 2 and 3. $\zeta^x_{(2,2)}(1) = \zeta^x_{(2,2)}(2) = 1$, $\psi^x_{(2,2)}(1) = y$, and $\psi^x_{(2,2)}(2) = x$.

(c) For $i = 2$ and $2 < j \leq N_y$, such that i and j stay within the allowed grid,

i. Compute $\mathcal{D}^1_{(2,j)}(y)$ using the equations for the first iteration of case 4. $\zeta^y_{(2,j)}(1) = 1$ and $\psi^y_{(2,j)}(1) = y$.

ii. Compute $\mathcal{D}^1_{(2,j)}(x)$ using the equations for the first iteration of case 2. $\zeta^x_{(2,j)}(1) = 1$ and $\psi^x_{(2,j)}(1) = y$.

iii. Propagate all parabolas $\mathcal{D}^k_{(2,j-1)}(x),\ k = 1, 2\cdots$ to $\mathcal{D}^{k+1}_{(2,j)}(y)$ using the equations for case 1. $\zeta^y_{(2,j)}(k+1) = k$ and $\psi^y_{(2,j)}(k+1) = x$.

iv. Propagate all parabolas $\mathcal{D}^k_{(2,j-1)}(x),\ k = 1, 2\cdots$ to $\mathcal{D}^{k+1}_{(2,j)}(x)$ using the equations for case 3. $\zeta^x_{(2,j)}(k+1) = k$ and $\psi^x_{(2,j)}(k+1) = x$.

(d) For $2 < i \leq N_x$ and $j = 2$, such that i and j stay within the allowed grid,

i. Compute $\mathcal{D}^1_{(i,2)}(y)$ using the equations for the first iteration of case 1. $\zeta^y_{(i,2)}(1) = 1$ and $\psi^y_{(i,2)}(1) = x$.

ii. Compute $\mathcal{D}^1_{(i,2)}(x)$ using the equations for the first iteration of case 3. $\zeta^x_{(i,2)}(1) = 1$ and $\psi^x_{(i,2)}(1) = x$.

iii. Propagate all parabolas $\mathcal{D}^k_{(i-1,2)}(y),\ k = 1, 2\cdots$ to $\mathcal{D}^{k+1}_{(i,2)}(y)$ using the equations for case 4. $\zeta^y_{(i,2)}(k+1) = k$ and $\psi^y_{(i,2)}(k+1) = y$.

iv. Propagate all parabolas $\mathcal{D}^k_{(i-1,2)}(x),\ k = 1, 2\cdots$ to $\mathcal{D}^{k+1}_{(i,2)}(x)$ using the equations for case 2. $\zeta^x_{(i,2)}(k+1) = k$ and $\psi^x_{(i,2)}(k+1) = y$.

2 Recursion: for $3 \leq i \leq N_x$, $3 \leq j \leq N_y$, such that i and j stay within the allowed grid,

(a) Propagate all parabolas $\mathcal{D}^k_{(i,j-1)}(x),\ k = 1, 2\cdots p_x$ to $\mathcal{D}^k_{(i,j)}(y)$ using the equations for case 1. $\zeta^y_{(i,j)}(k) = k$ and $\psi^y_{(i,j)}(k) = x$.

(b) Propagate all parabolas $\mathcal{D}^k_{(i,j-1)}(x),\ k = 1, 2\cdots p_x$ to $\mathcal{D}^k_{(i,j)}(x)$ using the equations for case 3. $\zeta^x_{(i,j)}(k) = k$ and $\psi^x_{(i,j)}(k) = x$.

(c) Propagate all parabolas $\mathcal{D}^{k}_{(i-1,j)}(y)$, $k = 1, 2 \cdots p_y$ to $\mathcal{D}^{k+p_x}_{(i,j)}(y)$ using the equations for case 4. $\zeta^{y}_{(i,j)}(k + p_x) = k$ and $\psi^{y}_{(i,j)}(k + p_x) = y$.

(d) Propagate all parabolas $\mathcal{D}^{k}_{(i-1,j)}(x)$, $k = 1, 2 \cdots p_y$ to $\mathcal{D}^{k+p_x}_{(i,j)}(x)$ using the equations for case 2. $\zeta^{x}_{(i,j)}(k + p_x) = k$ and $\psi^{x}_{(i,j)}(k + p_x) = y$.

3 Termination:

Find the minimum of all parabolas $\mathcal{D}_{(N_x,N_y)}(y)$ and $\mathcal{D}_{(N_x,N_y)}(x)$. The minimum value is the remaining distance after matching. The corresponding curvilinear coordinate x or y that provides the minimum distance needs to be propagated back through the warping plane to obtain the correspondence function.

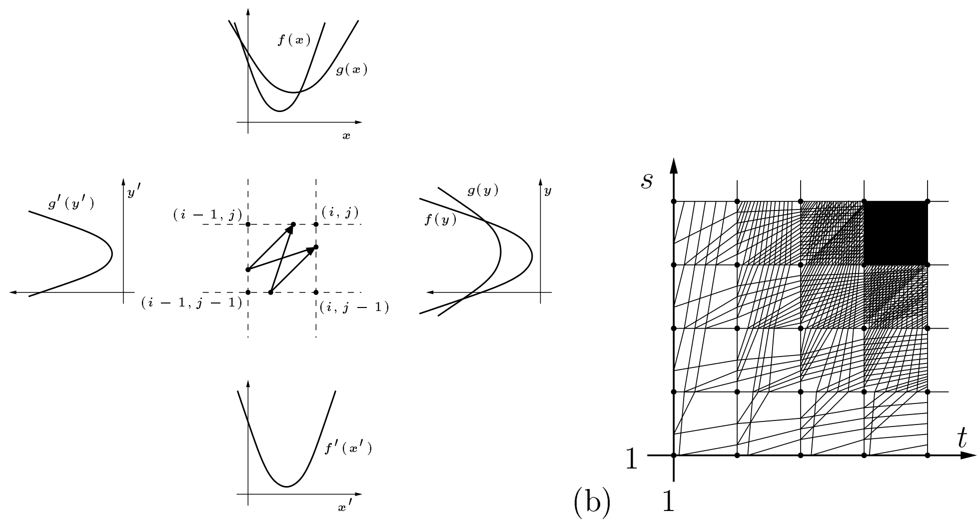
4 Path Backtracking:

Propagate the curvilinear coordinate x or y that provides the minimum distance using the equations for a single step shown in the previous section and the parent information stored in the variables $\zeta^{x,y}_{(i,j)}(k)$ and $\psi^{x,y}_{(i,j)}(k)$.

4.2.3 Computational complexity of the CDPM algorithm

The computational complexity of the CDPM algorithm is composed of two parts, one is the temporal complexity, i.e., the number of steps needed to compute in order to achieve the final result, and the spatial complexity, i.e., the number of variables needed to store in order to calculate this result. The temporal complexity of the above algorithm is given by the number of nodes of the grid within the allowed region of the warping plane. The maximum temporal complexity is $(N_x N_y)$ which occurs when all the nodes of the warping plane are explored during the computation. Therefore, the temporal complexity is $o(N_x N_y)$, polynomial in the number of samples on each curve.

The spatial complexity of the algorithm is more complex than the temporal one. Figure 4.9(a) shows the quadratic cumulated distances $f'(x')$ and $g'(y')$ corresponding to each of the input sides of the square at node (i, j) of the grid. Since there are four possible correspondence cases, two for each output side of the square, each quadratic function on the input sides gets propagated into two different quadratic functions at the output sides. In other words, for each output side we have two cumulated distance functions corresponding to the propagation of the quadratic functions at each of the input sides. Then, the number of cumulated distance functions *doubles* at each step of the recursion, resulting in a combinatorial explosion in the number of cumulated distance functions to be stored at each step of the algorithm and, therefore, in a spatial complexity that grows combinatorially. Figure 4.9(b) depicts the combinatorial explosion in the number of cumulated distance functions.



(a) (b)

Figure 4.9: (a) $f'(x')$ and $g'(x')$ are two quadratic functions corresponding to each of the input sides of the square of the grid at node (i, j). After the propagation of these functions with the four possible correspondence cases, we have four different quadratic functions, two for each output side of the square. We observe that the number of cumulate distance functions doubles at each step of the recursion. (b) Combinatorial explosion in the number of cumulated distance functions needed to be stored at each step of the algorithm. Each segment that joins two different sides of a square corresponds to the propagation of a cumulate distance function.

In fact, if no global constraint is imposed on the minimization of equation 3.6, the following property holds:

Property 4.2.2 *Consider the square of the grid on the warping plane corresponding to node (i, j), then the number of cumulated distance functions for each output side is* $\frac{(i-1+j-1)!}{(i-1)!(j-1)!}$.

Proof: We use induction. Let us call $N(i,j)$ the number of cumulated distance functions for each output side of the square corresponding to node (i,j) of the grid.

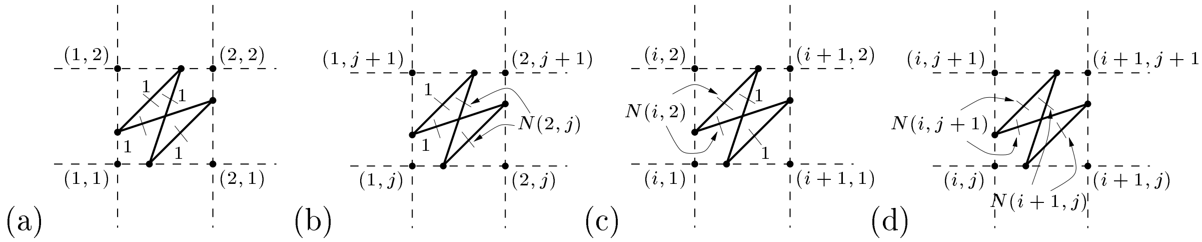


Figure 4.10: Number of cumulated distance functions.

From figure 4.10(a), we see that $N(2,2) = 2 = \frac{(2-1+2-1)!}{(2-1)!(2-1)!}$. Assume that $N(2,j) = \frac{(2-1+j-1)!}{(2-1)!(j-1)!} = j$, from figure 4.10(b), we see that $N(2,j+1) = j+1 = \frac{(2-1+j+1-1)!}{(2-1)!(j+1-1)!}$. Assume that $N(i,2) = \frac{(i-1+2-1)!}{(i-1)!(2-1)!} = i$, from figure 4.10(c), we see that $N(i+1,2) = i+1 = \frac{(i+1-1+2-1)!}{(i+1-1)!(2-1)!}$. Let us assume that the property holds for $i \geq 1$ and $j \geq 1$, then we have the following (see figure 4.10(d)):

$$
\begin{aligned}
N(i,j+1) = N(i,j) + N(i-1,j+1) &= \binom{i+j-2}{j-1} + \binom{i+j-2}{j} = \binom{i+j-1}{j} \\
N(i+1,j) = N(i,j) + N(i+1,j-1) &= \binom{i+j-2}{i-1} + \binom{i+j-2}{i} = \binom{i+j-1}{i} \\
N(i+1,j+1) = N(i,j+1) + N(i+1,j) &= \binom{i+j-1}{j} + \binom{i+j-1}{i} \\
&= \binom{i+j-1}{j} + \binom{i+j-1}{j-1} = \binom{i+j}{j} = \frac{(i+j)!}{i!j!}
\end{aligned}
$$

The induction is complete. ∎

Enforcing global constraints on the minimization would reduce the spatial complexity of the algorithm, although this spatial complexity would remain exponential in the number of iterations. In order to make the algorithm computationally efficient, we need to find conditions for bringing this spatial complexity down to a reasonable value (e.g., polynomial growth in the number of iterations). As mentioned before, we have a number of parabolas that are competing for providing the minimum matching distance. Each parabola corresponds to a possible series of elementary matchings, i.e., it is associated with a particular warping path. Since we are only interested in the matching function that provides the minimum distance between the curves, it should be possible to keep at each iteration only those parabolas that have a chance to give this minimum distance. In other words, we need to find properties that allow us to discard a whole set of parabolas at each iteration, while making sure that the parabola that provides the minimum distance is among the retained set.

Serra and Berthod [67, 68] proposed the use of heuristic constraints in order to limit the spatial complexity of the algorithm. They divided the range of excursion of x, y into a set of intervals, each of them corresponding to a particular distance function that is the minimum of all distance functions for this interval. At each iteration, they only kept the distance functions that belonged to the minimum envelope of all parabolas.

They assumed in their algorithm that, at all iterations, the parabola that provides the minimum matching distance had to belong to the minimum envelope of the set of parabolas in the interval $[0, \Delta]$. Figure 4.11 shows a counter-example to this assumption. We show the propagation of three different parabolas through four iterations of the algorithm. We see that the parabola plotted with solid line, that does not belong to the minimum envelope in the first plot, is the one that provides the minimum cost a few iterations later. We also see that after three iterations, the three parabolas are very similar, only displaced vertically, as if the coefficients A and B were the same and the difference between the C coefficients would be constant. We will describe this convergence of the coefficients of the parabolas in the following sections.

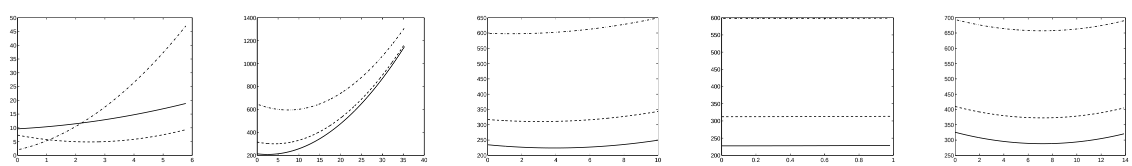


Figure 4.11: Propagation of three quadratic functions through four iterations of the algorithm. We observe that the parabola plotted with a solid line that does not belong to the minimum envelope in the first plot is later on the parabola that provides the minimum cost.

Serra and Berthod further assumed that the intersection points of different parabolas would be in correspondence as the algorithm proceeds. In other words, given two parabolas at one iteration and given the corresponding parabolas after propagation, then the intersection points of the latter pair of parabolas corresponds to the propagation of the intersection points of the first pair of parabolas. In figure 4.12 we show a counter-example to this assumption. What goes wrong is that each distance function has different propagation equations as shown in equations 4.5, 4.6, 4.9, 4.10 and, therefore, the intersection point of the two parabolas at one iteration does not correspond to the intersection point of the two parabolas after propagation.

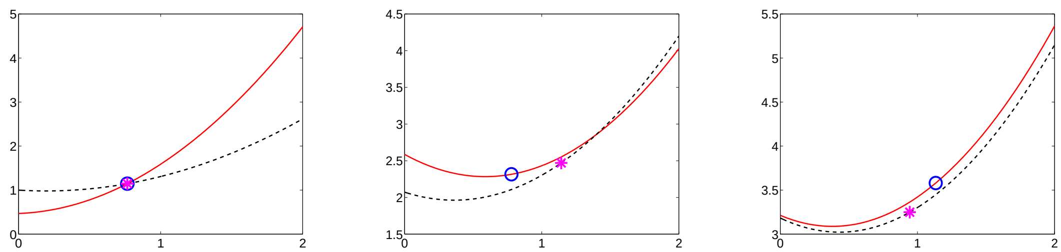


Figure 4.12: Propagation of two quadratic distance functions through two iterations of the algorithm. We observe that the position of the intersection of the parabolas at the initial condition corresponds to two different points after propagating the parabolas.

From figures 4.12 and 4.11, we observe that the relative position and the shape of the parabolas change from one iteration to the next. Therefore, we need to find properties that describe an order relationship between parabolas that is preserved through the iterations. In other words, we need properties that assure that a particular parabola is below some other parabola(s) on the interval $[0, \Delta]$ for all iterations. These properties allow us to perform comparisons between the parabolas in order to keep only the parabola that may provide the minimum matching distance. Considering a pair of parabolas, there are three different cases of intersection between them; they could either have no intersection or have one or two intersection points. Figure 4.13 shows these three possible intersection cases. The case in which the two parabolas intersect in only one point happens when the parabolas either are tangent to each other or have the same quadratic coefficient.

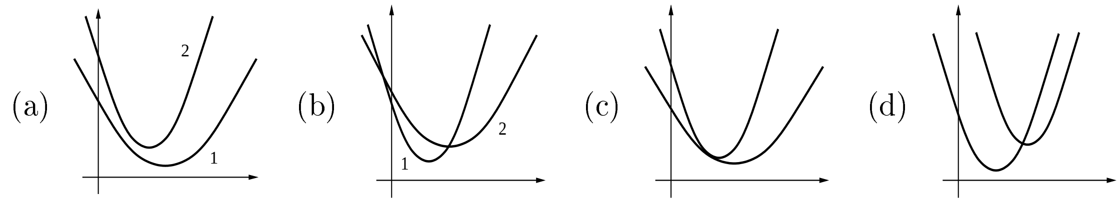


Figure 4.13: Cases of intersection between two parabolas: (a) no points of intersection, (b) two points of intersection and (c)-(d) one point of intersection ((c) corresponds to tangent parabolas and (d) corresponds to parabolas with the same quadratic coefficient).

In the following sections we describe two sets of properties that are used to decrease the spatial complexity of the algorithm. These properties focus on pairwise relationship between parabolas, in particular, on properties corresponding to cases (a) and (b) of figure 4.13. A pair of parabolas would compete and therefore would be compared whenever the associated warping paths coincide on the same square of the grid on the warping plane. The first time in which this comparison is performed would be considered as the initial condition. The first set of properties refers to the case of a pair of parabolas that do not intersect at the initial condition. We show that these two parabolas do not intersect at any further iteration, allowing one to discard the parabola that is biggest at all points (e.g., parabola 1 of figure 4.13(a)). The second set of properties refers to a pair of parabolas that intersect at the initial condition. We show that these two parabolas intersect at all further iterations. We demonstrate that the quadratic and linear coefficients of these two parabolas converge to the same value and that the zeroth order coefficient converges to a constant difference after a few iterations. Finally, we prove that the intersection points between the parabolas move towards infinity, allowing us to keep only the parabola that is smallest for all points on the interval of interest (e.g., parabola 1 of figure 4.13(b)).

4.2.4 Pairwise comparison of cumulate distance functions

The relationship between two cumulated distance functions and the changes in this relationship generated by the propagation of these distances through the warping plane are studied in this section.

The following property describes the behavior of the quadratic coefficient of the cumulated distance function through the iterations.

Property 4.2.3 *At every step of the algorithm in which the cumulated distance is a quadratic function of x, y, the quadratic coefficient of the parabola A is $0 < A \leq 1$.*

Proof: We use induction. Let us call A_k the value of A after k iterations of the algorithm.

For the first step of case 1, we showed in equation 4.4 that $A_1 = 1 - \cos^2\theta$. Since $0 \leq |\cos^2\theta| \leq 1$, then $0 \leq A_1 \leq 1$. $A = 0$ happens for $\theta = 0$ or $\theta = \pi$, a borderline situation between cases 1 and 3, in which the resultant cumulate distance is 0 and that can be analyzed in the same fashion that we analyze the first step of case 3. Thus, for $\theta \neq 0, \pi$, we have $0 < A_1 \leq 1$.

For the first step of case 3, we showed in equation 4.8 that the cumulated distance is a constant, and we showed in equation 4.9 that the cumulated distance remains a constant if the elementary matching occurs with case 3 (or case 4). However, equation 4.6 shows that $A = 1 - \cos^2\theta$ if the matching happens with case 1 (or case 2). As before, $\theta = 0$ or $\theta = \pi$ is a borderline situation between cases 1 and 3 that we regard as being case 3 since the cumulated distance continues being a constant. We consider the first matching in which the cumulated distance is a quadratic function, as the first step of the induction. Then, for $\theta \neq 0, \pi$, we have $0 < A_1 \leq 1$. The result holds for $n = 1$.

Assuming that $0 < A_k \leq 1$ for $k \geq 1$. For case 1, we have from equation 4.5 that $A_{k+1} = \frac{A_k + (1-\cos^2\theta)}{1+A_k}$, then

$0 < A_k \leq 1 \Rightarrow 1 < 1 + A_k \leq 2 \Rightarrow \frac{1}{2} \leq \frac{1}{1+A_k} < 1.$
$0 \leq \cos^2\theta \leq 1 \Rightarrow 0 \leq (1-\cos^2\theta) \leq 1 \Rightarrow A_k \leq A_k + (1-\cos^2\theta) \leq 1 + A_k \Rightarrow$
$\Rightarrow \frac{A_k}{1+A_k} \leq \frac{A_k+(1-\cos^2\theta)}{1+A_k} \leq 1 \Rightarrow \frac{A_k}{1+A_k} \leq A_{k+1} \leq 1.$
$\frac{1}{2} \leq \frac{1}{1+A_k} \Rightarrow \frac{A_k}{2} \leq \frac{A_k}{1+A_k} \leq A_{k+1} \Rightarrow 0 < \frac{A_k}{2} \leq \frac{A_k}{1+A_k} \leq A_{k+1} \leq 1$
$\Rightarrow 0 < A_{k+1} \leq 1$

For case 3, we have from equation 4.9 that $A_{k+1} = \frac{A_k}{1+A_k}$, then

$0 < A_k \leq 1 \Rightarrow 1 < 1 + A_k \leq 2 \Rightarrow \frac{1}{2} \leq \frac{1}{1+A_k} < 1 \Rightarrow \frac{A_k}{2} \leq \frac{A_k}{1+A_k} = A_{k+1} < A_k$
$\Rightarrow 0 < \frac{A_k}{2} \leq A_{k+1} < A_k \leq 1 \Rightarrow 0 < A_{k+1} \leq 1$

The induction is complete. ∎

Given two parabolas, we study their relationship by looking at their difference function and the propagation of this function through the computation. The following properties are stated considering case 1 and 3 since the other two cases are the dual ones. We abuse the notation by writing $\mathcal{D}(x)$ instead of $\mathcal{D}_{(i,j)}(x)$ and we use a quote $'$ to denote distance functions before propagating them using the formulas presented in section 4.2.1. We suppress the subscripts on Δ_x and $\cos\theta$ since the properties are demonstrated for a generic iteration of the algorithm.

Property 4.2.4 *Let $\mathcal{D}'^{(1)}(x') = A'_1x'^2 + 2B'_1x' + C'_1$ and $\mathcal{D}'^{(2)}(x') = A'_2x'^2 + 2B'_2x' + C'_2$ be two cumulate distance functions at a certain iteration of the algorithm. Let's call $\mathcal{D}^{(1)}(y) = A_1y^2 + 2B_1y + C_1$ and $\mathcal{D}^{(2)}(y) = A_2y^2 + 2B_2y + C_2$ the corresponding distance functions after propagating with case 1. Let us call $f'(x') = \mathcal{D}'^{(1)}(x') - \mathcal{D}'^{(2)}(x') = \Delta A'x'^2 + 2\Delta B'x' + \Delta C'$ and $f(y) = \mathcal{D}^{(1)}(y) - \mathcal{D}^{(2)}(y) = \Delta Ay^2 + 2\Delta By + \Delta C$ the corresponding difference functions. Then, having $A'_1 = \Delta A' + A'_2$, $B'_1 = \Delta B' + B'_2$,*

$C_1' = \Delta C' + C_2',$

$$\begin{aligned}
\Delta A &= \frac{\Delta A' \cos^2\theta}{(1 + A_2' + \Delta A')(1 + A_2')} \\
\Delta B &= -\frac{(\Delta_x - B_2')\Delta A' \cos\theta}{(1 + A_2' + \Delta A')(1 + A_2')} - \frac{\Delta B' \cos\theta}{(1 + A_2' + \Delta A')} \\
\Delta C &= \Delta C' + \frac{(B_2' - \Delta_x)^2 \Delta A'}{(1 + A_2' + \Delta A')(1 + A_2')} - \frac{2(B_2' - \Delta_x)\Delta B' + \Delta B'^2}{(1 + A_2' + \Delta A')} \\
(\Delta B^2 - \Delta A \Delta C) &= \frac{(\Delta B'^2 - \Delta A' \Delta C') \cos^2\theta}{(1 + A_2' + \Delta A')(1 + A_2')}
\end{aligned}$$

Proof:

$$\Delta A = A_1 - A_2 = \frac{A_2' + \Delta A' + 1 - \cos^2\theta}{1 + A_2' + \Delta A'} - \frac{A_2' + 1 - \cos^2\theta}{A_2' + 1} = \frac{\Delta A' \cos^2\theta}{(1 + A_2' + \Delta A')(1 + A_2')}$$

$$\begin{aligned}
\Delta B = B_1 - B_2 &= -\frac{((A_2' + \Delta A')\Delta_x + B_2' + \Delta B')\cos\theta}{1 + A_2' + \Delta A'} + \frac{(A_2'\Delta_x + B_2')\cos\theta}{A_2' + 1} \\
&= -\frac{(\Delta_x \Delta A' + \Delta B' + A_2' \Delta B' - B_2' \Delta A')\cos\theta}{(1 + A_2' + \Delta A')(1 + A_2')} = -\frac{(\Delta_x - B_2')\Delta A' \cos\theta}{(1 + A_2' + \Delta A')(1 + A_2')} - \frac{\Delta B' \cos\theta}{(1 + A_2' + \Delta A')}
\end{aligned}$$

$$\begin{aligned}
\Delta C = C_1 - C_2 &= C_1' - \frac{(\Delta_x - B_2' - \Delta B')^2}{1 + A_2' + \Delta A'} - C_2' + \frac{(\Delta_x - B_2')^2}{1 + A_2'} \\
&= \Delta C' + \frac{(B_2' - \Delta_x)^2 \Delta A'}{(1 + A_2' + \Delta A')(1 + A_2')} - \frac{2(B_2' - \Delta_x)\Delta B' + \Delta B'^2}{(1 + A_2' + \Delta A')}
\end{aligned}$$

Going through the algebra,

$$(\Delta B^2 - \Delta A \Delta C) = \frac{(\Delta B'^2 - \Delta A' \Delta C')\cos^2\theta}{(1 + A_2' + \Delta A')(1 + A_2')} \quad \blacksquare$$

Corollary 4.2.1 *If* $\cos\theta \neq 0$ *then*

1 If two parabolas intersect in two points at a certain iteration in the algorithm, *then they will intersect in two points after propagating with case 1.*

2 If two parabolas do not intersect at a certain iteration in the algorithm, then they will not intersect after propagating with case 1.

Proof: These two corollaries follow directly from properties 4.2.1, 4.2.4, given that $0 < \cos^2\theta \leq 1$, and $0 < A \leq 1$ as shown in property 4.2.3. ∎

Corollary 4.2.2 *If* $\mathcal{D}'^{(1)}(x') > \mathcal{D}'^{(2)}(x')$, $\forall x'$, *then* $\mathcal{D}^{(1)}(y) > \mathcal{D}^{(2)}(y)$, $\forall y$, *after propagating with case 1.*

Proof: Given that $\mathcal{D}'^{(1)}(x') > \mathcal{D}'^{(2)}(x')$, $\forall x'$, the function $f'(x') = \mathcal{D}'^{(1)}(x') - \mathcal{D}'^{(2)}(x')$ is strictly positive and is either a parabola or a constant. In the case of $f'(x')$ being a parabola, if $\cos^2\theta \neq 0$, then from corollary 4.2.1(2) and from the fact that ΔA keeps its sign after propagation as seen from property 4.2.4, the parabola $f(y) = \mathcal{D}^{(1)}(y) - \mathcal{D}^{(2)}(y)$ has its opening pointing upwards and does not intersect the horizontal axis after the propagation, then $\mathcal{D}^{(1)}(y) > \mathcal{D}^{(2)}(y)$, $\forall y$. In particular, $f(0) = \Delta C > 0$.

If $\cos^2\theta = 0$, then $A_1 = A_2 = 1$, $B_1 = B_2 = 0$ and $\Delta C = \Delta C' + \frac{(B'_2-\Delta_x)^2\Delta A'}{(1+A'_2+\Delta A')(1+A'_2)} - \frac{2(B'_2-\Delta_x)\Delta B'+\Delta B'^2}{(1+A'_2+\Delta A')}$; thus, $f(y) = \Delta C$ is a constant. The value of ΔC is independent of the value of $\cos^2\theta$. We proved in the previous paragraph that $\Delta C > 0$; therefore, $f(y) > 0$.

In the case in which $f'(x')$ is a constant, we have $A'_1 = A'_2$, $B'_1 = B'_2$, and $\Delta C' = f'(x') > 0$. Using the propagation equations of property 4.2.1, we have $A_1 = A_2$, $B_1 = B_2$, and $\Delta C = \Delta C' > 0$; therefore, $f(y) = \Delta C > 0$. ∎

Property 4.2.5 *Let* $\mathcal{D}'^{(1)}(x') = A'_1x'^2 + 2B'_1x' + C'_1$ *and* $\mathcal{D}'^{(2)}(x') = A'_2x'^2 + 2B'_2x' + C'_2$ *be two cumulate distance functions at a certain iteration of the algorithm. Let's call* $\mathcal{D}^{(1)}(x) = A_1x^2 + 2B_1x + C_1$ *and* $\mathcal{D}^{(2)}(x) = A_2x^2 + 2B_2x + C_2$ *the corresponding distance* *functions after propagation with case 3. Let us call* $f'(x') = \mathcal{D}'^{(1)}(x') - \mathcal{D}'^{(2)}(x') = \Delta A' x'^2 + 2\Delta B' x' + \Delta C'$ *and* $f(x) = \mathcal{D}^{(1)}(x) - \mathcal{D}^{(2)}(x) = \Delta A x^2 + 2\Delta B x + \Delta C$ *the corresponding difference functions. Then, having* $A'_1 = \Delta A' + A'_2$*,* $B'_1 = \Delta B' + B'_2$*,* $C'_1 = \Delta C' + C'_2$*,*

$$\begin{aligned}
\Delta A &= \frac{\Delta A'}{(1 + A'_2 + \Delta A')(1 + A'_2)} \\
\Delta B &= \frac{(B'_2 - \Delta_y \cos\theta)\Delta A'}{(1 + A'_2 + \Delta A')(1 + A'_2)} + \frac{\Delta B'}{(1 + A'_2 + \Delta A')} \\
\Delta C &= \Delta C' + \frac{(B'_2 + \Delta_y \cos\theta)^2 \Delta A'}{(1 + A'_2 + \Delta A')(1 + A'_2)} - \frac{2(B'_2 + \Delta_y \cos\theta)\Delta B' + \Delta B'^2}{(1 + A'_2 + \Delta A')} \\
(\Delta B^2 - \Delta A \Delta C) &= \frac{(\Delta B'^2 - \Delta A' \Delta C')}{(1 + A'_2 + \Delta A')(1 + A'_2)}
\end{aligned}$$

Proof:

$$\Delta A = A_1 - A_2 = \frac{A'_2 + \Delta A'}{1 + A'_2 + \Delta A'} - \frac{A'_2}{1 + A'_2} = \frac{\Delta A'}{(1 + A_{b_1})(1 + A_{b_2})}$$

$$\begin{aligned}
\Delta B = B_1 - B_2 &= \frac{B'_2 + \Delta B' - (A'_2 + \Delta A')\Delta_y \cos\theta}{1 + A'_2 + \Delta A'} - \frac{B'_2 - A'_2 \Delta_y \cos\theta}{1 + A'_2} = \frac{\Delta B' - \Delta_y \Delta A' \cos\theta + \Delta B' A'_2 - B'_2 \Delta A'}{(1 + A'_2 + \Delta A')(1 + A'_2)} \\
&= \frac{(B'_2 - \Delta_y \cos\theta)\Delta A'}{(1 + A'_2 + \Delta A')(1 + A'_2)} + \frac{\Delta B'}{(1 + A'_2 + \Delta A')}
\end{aligned}$$

$$\begin{aligned}
\Delta C = C_1 - C_2 &= C'_1 - \frac{(B'_2 + \Delta B' + \Delta_y \cos\theta)^2}{1 + A'_2 + \Delta A'} - C'_2 + \frac{(B'_2 + \Delta_y \cos\theta)^2}{1 + A'_2} \\
&= \Delta C' + \frac{(B'_2 + \Delta_y \cos\theta)^2 \Delta A'}{(1 + A'_2 + \Delta A')(1 + A'_2)} - \frac{2(B'_2 + \Delta_y \cos\theta)\Delta B' + \Delta B'^2}{(1 + A'_2 + \Delta A')}
\end{aligned}$$

Going through the algebra,

$$(\Delta B^2 - \Delta A \Delta C) = \frac{(\Delta B'^2 - \Delta A' \Delta C')}{(1 + A'_2 + \Delta A')(1 + A'_2)} \quad \blacksquare$$

Corollary 4.2.3

1 If two parabolas intersect in two points at a certain iteration in the algorithm, then they will intersect in two points after propagating with case 3.

2 If two parabolas do not intersect at a certain iteration in the algorithm, then they will not intersect after propagating with case 3.

Proof: These two corollaries follow directly from properties 4.2.1, 4.2.5, given that $0 \leq \cos^2\theta \leq 1$, and $0 < A \leq 1$ as shown in property 4.2.3. ∎

Corollary 4.2.4 *If $\mathcal{D}'^{(1)}(x') > \mathcal{D}'^{(2)}(x')$, $\forall x'$, then $\mathcal{D}^{(1)}(x) > \mathcal{D}^{(2)}(x)$, $\forall x$, after propagating with case 3.*

Proof: Given that $\mathcal{D}'^{(1)}(x') > \mathcal{D}'^{(2)}(x')$, $\forall x'$, the function $f'(x') = \mathcal{D}'^{(1)}(x') - \mathcal{D}'^{(2)}(x')$ is strictly positive and is either a parabola or a constant. In the case of $f'(x')$ being a parabola, then from corollary 4.2.3(2) and from the fact that ΔA keeps its sign after propagation as seen from property 4.2.5, the parabola $f(x) = \mathcal{D}^{(1)}(x) - \mathcal{D}^{(2)}(x)$ has its opening pointing upwards and does not intersect the horizontal axis after the propagation.

In the case in which $f'(x')$ is a constant, we have $A'_1 = A'_2$, $B'_1 = B'_2$, and $\Delta C' = f'(x') > 0$. Using the propagation equations of property 4.2.1, we have $A_1 = A_2$, $B_1 = B_2$, and $\Delta C = \Delta C' > 0$; therefore, $f(x) = \Delta C > 0$. ∎

The above properties allow us to compare all the quadratic distance functions pairwise and discard the ones that are "minimized" by another parabola for all the real line. These properties provide a way to break down the combinatorial explosion of the spatial complexity of the algorithm, as we will show in the experiments.

4.2.5 Asymptotic behavior of pairs of cumulated distance functions

The previous section described properties referring to pairs of non-intersecting cumulated distance functions. These properties allowed us to discard one of the cumulated distances of the pair. The remaining cumulated distances correspond to quadratic functions that intersect at all iterations. The coefficients of these quadratic functions have to be propagated throughout all iterations and determine the remaining spatial complexity of the algorithm. The study of the propagation properties of these coefficients provides grounds to further reduce the spatial complexity of the algorithm. Let us focus on the propagation of two of these quadratic functions throughout the warping plane. Let $\mathcal{D}^{(1)}(r) = A_1 r^2 + 2B_1 r + C_1$ and $\mathcal{D}^{(2)}(r) = A_2 r^2 + 2B_2 r + C_2$ be two quadratic functions at a particular iteration in the algorithm, where r generically represents either x, y. Let's call $f(r) = \mathcal{D}^{(1)}(r) - \mathcal{D}^{(2)}(r) = \Delta A r^2 + 2\Delta B r + \Delta C$. This section describes the behavior of the coefficients of $f(r)$ as the algorithm progresses through the warping plane. Let us call iteration 0 when the two functions are compared for the first time (i.e., when the associated warping paths coincide at the same square of the warping plane grid for the first time). Let us call ΔA_n, ΔB_n and ΔC_n the coefficients of $f_n(r)$, the difference function after n iterations of the algorithm. We also call $\cos\theta_n$ and Δ_n accordingly.

A generic path through the warping plane consists of transitions of type 1, 2 and type 3, 4; therefore, the two extreme cases of allowed paths are the ones that consist only of transitions of type 1 (or 2) and only of transitions of type 3 (or 4), as shown in figure 4.14. The asymptotic behavior of any path falls between the behavior of these two extreme cases and we study them separately.

The equations for a path that consists only of transitions of type 3,4 are simpler, so we present them first.

Property 4.2.6 $A_n = \dfrac{A_0}{1 + nA_0}$

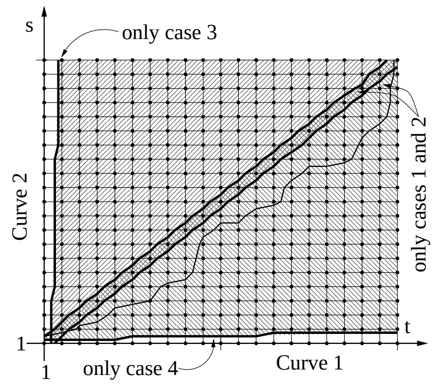


Figure 4.14: A generic path through the warping plane consists of transitions of type 1, 2 and type 3, 4; therefore, the two extreme cases of allowed paths are the ones that consist only of transitions of type 1 (or 2) and only of transitions of type 3 (or 4).

Proof: We use induction. From equation 4.10, we have that $A_n = \frac{A_{n-1}}{1+A_{n-1}}$, then $A_1 = \frac{A_0}{1+A_0}$. The result holds for $n = 1$. Assume that it holds for $k \geq 1$. Therefore, $A_{k+1} = \frac{A_k}{1+A_k} = \frac{A_0}{1+kA_0}\frac{1+kA_0}{1+kA_0+A_0} = \frac{A_0}{1+(k+1)A_0}$. The induction is complete. ∎

Property 4.2.7 $\Delta A_n = \dfrac{\Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)}$

Proof: We use induction. From property 4.2.5, we have that $\Delta A_n = \frac{\Delta A_{n-1}}{(1+A_{n-1}+\Delta A_{n-1})(1+A_{n-1})}$, then $\Delta A_1 = \frac{\Delta A_0}{(1+A_0+\Delta A_0)(1+A_0)}$. The result holds for $n = 1$. Assume that it holds for $k \geq 1$. Therefore,

$$\begin{aligned}\Delta A_{k+1} &= \frac{\Delta A_k}{(1+A_k+\Delta A_k)(1+A_k)} = \frac{\Delta A_0}{(1+kA_0)(1+kA_0+k\Delta A_0)}\frac{(1+kA_0)}{(1+kA_0+A_0)}\frac{(1+kA_0+k\Delta A_0)}{(1+kA_0+k\Delta A_0+A_0+\Delta A_0)} \\ &= \frac{\Delta A_0}{(1+(k+1)A_0)(1+(k+1)A_0+(k+1)\Delta A_0)}\end{aligned}$$

The induction is complete. ∎

Property 4.2.8 $\Delta B_n^2 - \Delta A_n \Delta C_n = \dfrac{\Delta B_0^2 - \Delta A_0 \Delta C_0}{(1+nA_0)(1+nA_0+n\Delta A_0)}$

Proof: We use induction. From property 4.2.5, we have that $\Delta B_n^2 - \Delta A_n \Delta C_n = \frac{\Delta B_{n-1}^2 - \Delta A_{n-1}\Delta C_{n-1}}{(1+A_{n-1}+\Delta A_{n-1})(1+A_{n-1})}$, then $\Delta B_1^2 - \Delta A_1 \Delta C_1 = \frac{\Delta B_0^2 - \Delta A_0 \Delta C_0}{(1+A_0+\Delta A_0)(1+A_0)}$. The result holds for $n = 1$. Assume that it holds for $k \geq 1$. Therefore,

$$\begin{aligned}
\Delta B_{k+1}^2 - \Delta A_{k+1}\Delta C_{k+1} &= \frac{\Delta B_k^2 - \Delta A_k \Delta C_k}{(1+A_k+\Delta A_k)(1+A_k)} \\
&= \frac{\Delta B_0^2 - \Delta A_0 \Delta C_0}{(1+kA_0)(1+kA_0+k\Delta A_0)} \frac{(1+kA_0)}{(1+(k+1)A_0)} \frac{(1+kA_0+k\Delta A_0)}{(1+(k+1)A_0+(k+1)\Delta A_0)} \\
&= \frac{\Delta B_0^2 - \Delta A_0 \Delta C_0}{(1+(k+1)A_0)(1+(k+1)A_0+(k+1)\Delta A_0)}
\end{aligned}$$

The induction is complete. ∎

Property 4.2.9 $B_n = \dfrac{B_0}{1+nA_0} - \dfrac{A_0}{1+nA_0} \sum_{j=0}^{n-1} \Delta_j \cos\theta_j$

Proof: We use induction. From equation 4.10, we have that $B_n = \frac{B_{n-1}}{1+A_{n-1}} - \frac{A_{n-1}\Delta_{n-1}\cos\theta_{n-1}}{1+A_{n-1}}$, then $B_1 = \frac{B_0}{1+A_0} - \frac{A_0}{1+A_0}\Delta_0 \cos\theta_0$. The result holds for $n = 1$. Assume that it holds for $k \geq 1$. Therefore, $B_{k+1} = \frac{B_k - A_k \Delta_k \cos\theta_k}{1+A_k} = -\frac{A_k \Delta_k \cos\theta_k}{1+A_k} + \frac{1+kA_0}{1+(k+1)A_0}\left(\frac{B_0}{1+kA_0} - \frac{A_0}{1+kA_0}\sum_{j=0}^{k-1}\Delta_j \cos\theta_j\right) = \frac{B_0}{1+(k+1)A_0} - \frac{A_0}{1+(k+1)A_0}\sum_{j=0}^{k}\Delta_j\cos\theta_j$. The induction is complete. ∎

Property 4.2.10

$$\Delta B_n = \frac{\Delta B_0}{1+nA_0+n\Delta A_0} - \frac{nB_0\Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)} - \frac{\Delta A_0 \sum\limits_{j=0}^{n-1} \Delta_j \cos\theta_j}{(1+nA_0)(1+nA_0+n\Delta A_0)}$$

Proof: From property 4.2.9 we have that $B_n = \frac{B_0}{1+nA_0} - \frac{A_0}{1+nA_0}\sum_{j=0}^{n-1}\Delta_j\cos\theta_j$, then $B_n + \Delta B_n = \frac{B_0+\Delta B_0}{1+n(A_0+\Delta A_0)} - \frac{A_0+\Delta A_0}{1+n(A_0+\Delta A_0)}\sum_{j=0}^{n-1}\Delta_j\cos\theta_j$. Therefore, $\Delta B_n = \left(\frac{B_0+\Delta B_0}{1+n(A_0+\Delta A_0)} - \frac{B_0}{1+nA_0}\right) + \left(\frac{A_0}{1+nA_0} - \frac{A_0+\Delta A_0}{1+n(A_0+\Delta A_0)}\right)\sum_{j=0}^{n-1}\Delta_j\cos\theta_j = \frac{\Delta B_0 + n(A_0\Delta B_0 - B_0\Delta A_0)}{(1+nA_0)(1+nA_0+n\Delta A_0)} - \frac{\Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)}\sum_{j=0}^{n-1}\Delta_j\cos\theta_j$. ∎

Property 4.2.11

$$\Delta C_n = \Delta C_0 + \frac{(nB_0 + \sum_{i=0}^{n-1}\Delta_i \cos\theta_i)^2 \Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)} - \frac{n\Delta B_0^2 + 2nB_0\Delta B_0 + 2\Delta B_0 \sum_{i=0}^{n-1}\Delta_i\cos\theta_i}{(1+nA_0+n\Delta A_0)}$$

Proof: We use induction. From property 4.2.5, we have that $\Delta C_n = \Delta C_{n-1} + \frac{(B_{n-1}+\Delta_{n-1}\cos\theta_{n-1})^2\Delta A_{n-1}}{(1+A_{n-1}+\Delta A_{n-1})(1+A_{n-1})} - \frac{2(B_{n-1}+\Delta_{n-1}\cos\theta_{n-1})\Delta B_{n-1}+\Delta B_{n-1}^2}{(1+A_{n-1}+\Delta A_{n-1})}$, then $\Delta C_1 = \Delta C_0 + \frac{(B_0+\Delta_0\cos\theta_0)^2\Delta A_0}{(1+A_0+\Delta A_0)(1+A_0)} - \frac{2(B_0+\Delta_0\cos\theta_0)\Delta B_0+\Delta B_0^2}{(1+A_0+\Delta A_0)}$. The result holds for $n=1$. Assume that it holds for $k \geq 1$. Therefore,

$$\begin{aligned}
\Delta C_{k+1} &= \Delta C_k + \frac{(B_k+\Delta_k\cos\theta_k)^2\Delta A_k}{(1+A_k+\Delta A_k)(1+A_k)} - \frac{2(B_k+\Delta_k\cos\theta_k)\Delta B_k+\Delta B_k^2}{(1+A_k+\Delta A_k)} \\
&= \Delta C_0 + \frac{(kB_0+\sum_{i=0}^{k-1}\Delta_i\cos\theta_i)^2\Delta A_0}{(1+kA_0)(1+kA_0+k\Delta A_0)} - \frac{k\Delta B_0^2+2kB_0\Delta B_0+2\Delta B_0\sum_{i=0}^{k-1}\Delta_i\cos\theta_i}{(1+kA_0+k\Delta A_0)} + \\
&\quad \frac{(B_k+\Delta_k\cos\theta_k)^2\Delta A_k}{(1+A_k+\Delta A_k)(1+A_k)} - \frac{2(B_k+\Delta_k\cos\theta_k)\Delta B_k+\Delta B_k^2}{(1+A_k+\Delta A_k)} \\
&= \Delta C_0 + \frac{(kB_0+\sum_{i=0}^{k-1}\Delta_i\cos\theta_i)^2\Delta A_0}{(1+kA_0)(1+kA_0+k\Delta A_0)} - \frac{k\Delta B_0^2+2kB_0\Delta B_0+2\Delta B_0\sum_{i=0}^{k-1}\Delta_i\cos\theta_i}{(1+kA_0+k\Delta A_0)} + \\
&\quad \frac{(\frac{B_0}{1+kA_0} - \frac{A_0}{1+kA_0}\sum_{j=0}^{k-1}\Delta_j\cos\theta_j + \Delta_k\cos\theta_k)^2\Delta A_0}{(1+(k+1)A_0+(k+1)\Delta A_0)(1+(k+1)A_0)} - \frac{(\frac{\Delta B_0}{(1+kA_0+k\Delta A_0)} - \frac{\Delta A_0(kB_0+\sum_{j=0}^{k-1}\Delta_j\cos\theta_j)}{(1+kA_0)(1+kA_0+k\Delta A_0)})^2}{\frac{(1+(k+1)A_0+(k+1)\Delta A_0)}{(1+kA_0+k\Delta A_0)}} - \\
&\quad \frac{2(\frac{B_0}{1+kA_0} - \frac{A_0}{1+kA_0}\sum_{j=0}^{k-1}\Delta_j\cos\theta_j + \Delta_k\cos\theta_k)(\frac{\Delta B_0}{(1+kA_0+k\Delta A_0)} - \frac{\Delta A_0(kB_0+\sum_{j=0}^{k-1}\Delta_j\cos\theta_j)}{(1+kA_0)(1+kA_0+k\Delta A_0)})}{\frac{(1+(k+1)A_0+(k+1)\Delta A_0)}{(1+kA_0+k\Delta A_0)}} = \cdots \\
&= \Delta C_0 + \frac{((k+1)B_0+\sum_{i=0}^{k}\Delta_i\cos\theta_i)^2\Delta A_0}{(1+(k+1)A_0)(1+(k+1)A_0+(k+1)\Delta A_0)} - \frac{(k+1)\Delta B_0^2+2(k+1)B_0\Delta B_0+2\Delta B_0\sum_{i=0}^{k}\Delta_i\cos\theta_i}{(1+(k+1)A_0+(k+1)\Delta A_0)}
\end{aligned}$$

The induction is complete. ∎

From the expressions for ΔA_n and ΔB_n, it is clear that $\Delta A_n \underset{n\to\infty}{\longrightarrow} 0$ quadratically and $\Delta B_n \underset{n\to\infty}{\longrightarrow} 0$ linearly. From the expression for ΔC_n, we see that $|\Delta C_n|$ is bounded. In fact, $|\Delta C_n| \leq |\Delta C_0| + \frac{(n|B_0|+n\Delta_{max})^2|\Delta A_0|}{(1+nA_0)(1+nA_0+n\Delta A_0)} + \frac{n\Delta B_0^2+2n|B_0||\Delta B_0|+2n|\Delta B_0|\Delta_{max}}{(1+nA_0+n\Delta A_0)} \underset{n\to\infty}{\longrightarrow} |\Delta C_0| + \frac{(|B_0|+\Delta_{max})^2|\Delta A_0|}{A_0(A_0+\Delta A_0)} + \frac{\Delta B_0^2+2|B_0||\Delta B_0|+2|\Delta B_0|\Delta_{max}}{(A_0+\Delta A_0)}$.

Let us call $x_{1,2}$ the zeros of $f(r) = \mathcal{D}^{(1)}(r) - \mathcal{D}^{(2)}(r)$. The corresponding equations are $x_1 = -\,\mathrm{sign}(\Delta B)(\frac{|\Delta B|}{\Delta A} + \frac{\sqrt{\Delta B^2 - \Delta A\Delta C}}{\Delta A})$ and $x_2 = -\,\mathrm{sign}(\Delta B)\frac{\Delta C}{|\Delta B|+\sqrt{\Delta B^2-\Delta A\Delta C}}$.

The zeros of $f(r)$ define the interval in which $\mathcal{D}^{(1)}(r) \leq \mathcal{D}^{(2)}(r)$, if $\Delta A > 0$ (or the interval in which $\mathcal{D}^{(1)}(r) \geq \mathcal{D}^{(2)}(r)$, if $\Delta A < 0$), so if the position of $x_{1,2}$ diverges with n, it is possible to find out which parabola provides the minimum cost.

Property 4.2.12 $|x_{1_n}| \underset{n\to\infty}{\longrightarrow} \infty$, $|x_{2_n}| \underset{n\to\infty}{\longrightarrow} \infty$ *if* $|\Delta C_n| \underset{n\to\infty}{\not\rightarrow} 0$

Proof: For $|x_{1_n}| = \frac{|\Delta B|}{|\Delta A|} + \frac{\sqrt{\Delta B^2 - \Delta A \Delta C}}{|\Delta A|}$ we have

$$|x_{1_n}| = \frac{|\Delta B_n|}{|\Delta A_n|} + \frac{\sqrt{\Delta {B_n}^2 - \Delta A_n \Delta C_n}}{|\Delta A_n|} \geq \frac{\sqrt{\Delta {B_n}^2 - \Delta A_n \Delta C_n}}{|\Delta A_n|}$$
$$= \sqrt{\frac{\Delta {B_0}^2 - \Delta A_0 \Delta C_0}{\Delta A_0^2}} \sqrt{(1 + nA_0)(1 + nA_0 + n\Delta A_0)} \underset{n\to\infty}{\longrightarrow} \infty$$

For $|x_{2_n}| = \frac{|\Delta C_n|}{|\Delta B_n| + \sqrt{\Delta {B_n}^2 - \Delta A_n \Delta C_n}}$, we study the numerator and the denominator separately. The denominator of $|x_{2_n}|$ is as follows:

$$|\Delta B_n| + \sqrt{\Delta {B_n}^2 - \Delta A_n \Delta C_n} = \left| \frac{\Delta B_0}{(1+nA_0+n\Delta A_0)} - \frac{nB_0\Delta A_0 - \Delta A_0 \sum_{j=0}^{n-1} \Delta_j \cos\theta_j}{(1+nA_0)(1+nA_0+n\Delta A_0)} \right|$$
$$+ \sqrt{\frac{\Delta {B_0}^2 - \Delta A_0 \Delta C_0}{(1+nA_0)(1+nA_0+n\Delta A_0)}} = \frac{1}{n}\left[\left| \frac{\Delta B_0}{\frac{1}{n} + A_0 + \Delta A_0} - \frac{B_0 \Delta A_0 - \Delta A_0 (\frac{1}{n} \sum_{j=0}^{n-1} \Delta_j \cos\theta_j)}{(\frac{1}{n} + A_0)(\frac{1}{n} + A_0 + \Delta A_0)} \right| \right.$$
$$\left. + \sqrt{\frac{\Delta {B_0}^2 - \Delta A_0 \Delta C_0}{(\frac{1}{n} + A_0)(\frac{1}{n} + A_0 + \Delta A_0)}} \right]$$

The denominator of $|x_{2_n}|$ is composed of two factors; one of them is $\frac{1}{n}$, that tends to zero as $n \to \infty$, and the other is a quantity that tends to a constant as $n \to \infty$. This second factor tends to a constant since $A_0 > 0$ and $A_0 + \Delta A_0 > 0$ as shown in property 4.2.3 and the term $\frac{1}{n} \sum_{j=0}^{n-1} \Delta_j \cos\theta_j$ is bounded between 0 and Δ_{max} due to the fact that $0 \leq |\cos\theta_j| \leq 1, \forall j$ and $0 \leq \Delta_j \leq \Delta_{max}, \forall j$. Moving the n in the factor $\frac{1}{n}$ from the denominator to the numerator, we have a resulting numerator of the form $n|\Delta C_n|$, leaving the denominator with an expression that tends to a constant as $n \to \infty$. Since $|\Delta C_n|$ is bounded for all n, the resulting numerator $n|\Delta C_n| \underset{n\to\infty}{\longrightarrow} \infty$ unless $|\Delta C_n| \underset{n\to\infty}{\longrightarrow} 0$. Therefore, $|x_{2_n}| \underset{n\to\infty}{\longrightarrow} \infty$ if $|\Delta C_n| \underset{n\to\infty}{\not\rightarrow} 0$. ∎

Let us explore the conditions for $|\Delta C_n| \underset{n\to\infty}{\longrightarrow} 0$. Let us call $\xi_n = \sum_{i=0}^{n-1} \Delta_i \cos\theta_i$, then the expression for ΔC_n is the following:

$$\Delta C_n = \left[\Delta C_0 + \frac{(nB_0+\xi_n)^2\Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)} - \frac{n\Delta B_0^2+2nB_0\Delta B_0+2\Delta B_0\xi_n}{(1+nA_0+n\Delta A_0)}\right]$$
$$= \frac{\left[(1+nA_0)((1+nA_0+n\Delta A_0)\Delta C_0-n\Delta B_0^2-2nB_0\Delta B_0)+n^2B_0^2\Delta A_0\right]+2\xi_n(nB_0\Delta A_0-(1+nA_0)\Delta B_0)+\Delta A_0\xi_n^2}{(1+nA_0)(1+nA_0+n\Delta A_0)}$$

The equation for ΔC_n has a numerator that is a quadratic equation on ξ_n, so let us find the values of ξ_n that makes the numerator of ΔC_n equal to zero. The solution is the following

$$\xi_{n_{1,2}} = n\left[-B_0 + (A_0+\frac{1}{n})\frac{\Delta B_0}{\Delta A_0} \pm \sqrt{(A_0+\frac{1}{n})(A_0+\Delta A_0+\frac{1}{n})\frac{(\Delta B_0^2-\Delta A_0\Delta C_0)}{\Delta A_0^2}}\right] \tag{4.11}$$

For this particular value of ξ_n, the denominator of $|x_{2_n}|$ takes the following value:

$$|\Delta B_n| + \sqrt{{\Delta B_n}^2 - \Delta A_n\Delta C_n} = \frac{1}{n}\left[\sqrt{\frac{{\Delta B_0}^2-\Delta A_0\Delta C_0}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}} + \left|\frac{\Delta B_0(\frac{1}{n}+A_0)-B_0\Delta A_0}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)} - \right.\right.$$
$$\left.\left.\frac{\Delta A_0\left(-B_0+(A_0+\frac{1}{n})\frac{\Delta B_0}{\Delta A_0}\pm\sqrt{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)\frac{(\Delta B_0^2-\Delta A_0\Delta C_0)}{\Delta A_0^2}}\right)}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}\right|\right] = \frac{2}{n}\sqrt{\frac{{\Delta B_0}^2-\Delta A_0\Delta C_0}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}}$$

The factor ${\Delta B_0}^2-\Delta A_0\Delta C_0 \neq 0$ since these properties correspond to a pair of parabolas that intersect. The denominator of $|x_{2_n}|$ is different from zero and the numerator of $|x_{2_n}|$ is equal to $\Delta C_n = 0$ for this particular value of ξ_n; therefore, $|x_{2_n}| = 0$, $\forall n$.

The particular value of ξ_n given by equation 4.11 is defined by having $\Delta C_n = 0$.

Let us find out whether there is a different condition on ξ_n such that $|\Delta C_n| \underset{n\to\infty}{\longrightarrow} 0$. Let us write the equation for ΔC_n having several terms re-arranged

$$\Delta C_n = \left[\Delta C_0 + \frac{(nB_0+\xi_n)^2\Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)} - \frac{n\Delta B_0^2+2nB_0\Delta B_0+2\Delta B_0\xi_n}{(1+nA_0+n\Delta A_0)}\right]$$
$$= \frac{(A_0(A_0+\Delta A_0)\Delta C_0+B_0^2\Delta A_0-A_0\Delta B_0^2-2A_0B_0\Delta B_0)+2\frac{\xi_n}{n}(B_0\Delta A_0-A_0\Delta B_0)+\Delta A_0\frac{\xi_n^2}{n^2}}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}$$
$$+ \frac{\frac{(\Delta C_0(2A_0+\Delta A_0)-\Delta B_0^2-2B_0\Delta B_0)}{n}-2\Delta B_0\frac{\xi_n}{n^2}+\frac{\Delta C_0}{n^2}}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}$$

Let us call $\xi = \lim_{n\to\infty}\frac{1}{n}\xi_n$. Since $0 \le |\cos\theta_j| \le 1$, $\forall j$ and $0 \le \Delta_j \le \Delta_{max}$, $\forall j$, the value of $|\xi|$ is bounded $0 \le |\xi| \le \Delta_{max}$. Therefore, the three first terms of the above expression converge to a constant value as $n \to \infty$ while the last three terms converge to zero as $n \to \infty$. The questions is whether the first three terms cancel out making $|\Delta C_n| \underset{n\to\infty}{\longrightarrow} 0$. Let us find out the condition on ξ_n that make the sum of these terms equal to zero for all n. We need to solve the following quadratic equation on ξ_n.

$$\Delta A_0\xi_n^2 + 2n\xi_n(B_0\Delta A_0 - A_0\Delta B_0)$$
$$+ n^2(A_0(A_0+\Delta A_0)\Delta C_0 + B_0^2\Delta A_0 - A_0\Delta B_0^2 - 2A_0B_0\Delta B_0) = 0$$

The solution for ξ_n is the following:

$$\xi^*_{n_{1,2}} = n\left[-B_0 + A_0\frac{\Delta B_0}{\Delta A_0} \pm \sqrt{\frac{A_0(A_0+\Delta A_0)(\Delta B_0^2 - \Delta A_0\Delta C_0)}{\Delta A_0^2}}\right] \tag{4.12}$$

For this particular value of ξ_n, the denominator of $|x_{2_n}|$ takes the following value:

$$|\Delta B_n| + \sqrt{{\Delta B_n}^2 - \Delta A_n \Delta C_n} = \frac{1}{n}\left[\left|\frac{\Delta B_0(\frac{1}{n}+A_0) - B_0 \Delta A_0}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}\right.\right.$$

$$\left.\left. - \frac{\Delta A_0\left(-B_0 + A_0\frac{\Delta B_0}{\Delta A_0} \pm \sqrt{\frac{A_0(A_0+\Delta A_0)(\Delta B_0^2 - \Delta A_0 \Delta C_0)}{\Delta A_0^2}}\right)}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}\right| + \sqrt{\frac{{\Delta B_0}^2 - \Delta A_0 \Delta C_0}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}}\right]$$

$$= \frac{1}{n}\left[\left|\frac{\frac{\Delta B_0}{n} \mp \sqrt{A_0(A_0+\Delta A_0)(\Delta B_0^2 - \Delta A_0 \Delta C_0)}}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}\right| + \sqrt{\frac{{\Delta B_0}^2 - \Delta A_0 \Delta C_0}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}}\right]$$

For this particular value of ξ_n, the numerator of $|x_{2_n}|$ takes the following value:

$$\Delta C_n = \frac{1}{n}\left[\frac{\frac{(2A_0+\Delta A_0)(\Delta A_0 \Delta C_0 - {\Delta B_0}^2) \mp \sqrt{A_0(A_0+\Delta A_0)(\Delta B_0^2 - \Delta A_0 \Delta C_0)}}{\Delta A_0} + \frac{\Delta C_0}{n}}{(\frac{1}{n}+A_0)(\frac{1}{n}+A_0+\Delta A_0)}\right]$$

The multiplicative factors $\frac{1}{n}$ in the numerator and the denominator of $|x_{2_n}|$ cancel out, leaving an expression for $|x_{2_n}|$ that tends to a constant as $n \to \infty$. The value of this constant is defined by the initial conditions A_0, B_0, C_0, ΔA_0, ΔB_0, and ΔC_0. Therefore, depending on the value of ξ_n, the zero $|x_{2_n}|$ tends to infinity, to a constant or is equal to zero.

The conditions imposed on ξ_n by equations 4.11 and 4.11 are quite restrictive since there is no a priori information on the values of A_0, B_0, C_0, ΔA_0, ΔB_0, ΔC_0, $\cos\theta_i$ and Δ_i other than $\cos\theta_i \approx 1$ for similar curves, $0 < A_0 \leq 1$, and ${\Delta B_0}^2 - \Delta A_0 \Delta C_0 \neq 0$. Therefore, it is reasonable to expect that in general $\xi_n \neq \xi_{n_{1,2}}$, and the value of $n|\Delta C_n|$ diverges as $n \to \infty$.

We showed that $|x_{1_n}|$ and $|x_{2_n}|$ diverge as $n \to \infty$, but we do not know whether they had the same sign or not. The first plot of figure 4.15 shows the case of the two zeros having different sign and the last two plots of the figure show the case of the two zeros having the same sign. In the first case, the parabola that provides the minimum cumulated distance is $\mathcal{D}^{(2)}(r)$ while in the last two cases the minimum is achieved by $\mathcal{D}^{(1)}(r)$.

The product of the two zeros is $x_{1_n} x_{2_n} = \frac{\Delta C_n}{\Delta A_n}$, so the sign of $\frac{\Delta C_n}{\Delta A_n}$ tells us whether

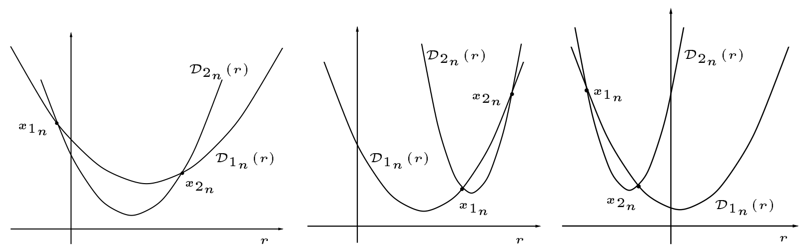


Figure 4.15: Different possible locations of x_{1_n} and x_{2_n}.

the two zeros have the same sign or not. From property 4.2.7 we see that the sign of ΔA_n is the same for all iterations, thus the sign of ΔC_n carries the information about the sign of the zeros.

We observe that depending on the value of ξ_n, ΔC_n could be either positive or negative and could switch signs from iteration to iteration. Looking at figure 4.15, it would be possible that in one iteration the parabola that provides the minimum would be $\mathcal{D}^{(1)}(r)$ while in the following one it would be $\mathcal{D}^{(2)}(r)$; therefore, it would not be possible to make an optimal decision.

The value of $\frac{\xi_n}{n} = \frac{1}{n}\sum_{j=0}^{n-1}\Delta_j \cos\theta_j$ is bounded $0 \leq |\frac{\xi_n}{n}| \leq \Delta_{max}$ since $0 \leq |\cos\theta_j| \leq 1$, $\forall j$ and $0 \leq \Delta_j \leq \Delta_{max}$, $\forall j$. If $[-\Delta_{max}, \Delta_{max}] \subset [\frac{\xi_{n_1}}{n}, \frac{\xi_{n_2}}{n}]$, $\forall n$, then $\xi_n \in [\xi_{n_1}, \xi_{n_2}]$, $\forall n$, and the sign of ΔC_n would be opposite to the sign of ΔA_n for all the iterations; therefore, the signs of $x_{1,2_n}$ would always be different and the situation would correspond to the first plot of figure 4.15 and we could make an optimal decision. Since the values of $\xi_{n_{1,2}}$ depend only on the initial conditions of the two parabolas, the check for the above condition could be done at the very first iteration and then make the decision on which parabola to keep later, whenever the coefficients of the two parabolas have converged.

For the extreme case of a path consisting only of transitions of type 1,2, the equations are a bit more complicated. There is a particular case if $\cos^2\theta_n = 1\ \forall n$, in which the equations simplify and resemble the equations for the extreme case of a path consisting only of transition of type 3,4. It can be shown that the following properties hold:

$$
\begin{aligned}
&A_n = \tfrac{A_0}{1+nA_0} \\
&\Delta A_n = \tfrac{\Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)} \\
&\Delta B_n^2 - \Delta A_n \Delta C_n = \tfrac{\Delta B_0^2 - \Delta A_0 \Delta C_0}{(1+nA_0)(1+nA_0+n\Delta A_0)} \\
&B_n = -\tfrac{A_0 \sum_{j=0}^{n-1} (-1)^{n-j} \Delta_j}{1+nA_0} + \tfrac{(-1)^n B_0}{1+nA_0} \\
&\Delta B_n = \tfrac{(-1)^n \Delta B_0}{1+nA_0+n\Delta A_0} + \tfrac{(-1)^{n+1} n B_0 \Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)} - \tfrac{\Delta A_0 \sum_{j=0}^{n-1} (-1)^{n-j} \Delta_j}{(1+nA_0)(1+nA_0+n\Delta A_0)} \\
&\Delta C_n = \Delta C_0 + \tfrac{(nB_0 - \sum_{i=0}^{n-1} (-1)^i \Delta_i)^2 \Delta A_0}{(1+nA_0)(1+nA_0+n\Delta A_0)} - \tfrac{n\Delta B_0^2 + 2nB_0 \Delta B_0 + 2\Delta B_0 \sum_{i=0}^{n-1} (-1)^i \Delta_i}{(1+nA_0+n\Delta A_0} \\
&|x_{1_n}| \underset{n\to\infty}{\longrightarrow} \infty \\
&|x_{2_n}| \underset{n\to\infty}{\longrightarrow} \infty \quad \text{if} \quad \Delta C_n| \underset{n\to\infty}{\nrightarrow} 0
\end{aligned}
\tag{4.13}
$$

In the general case, $\cos^2\theta_n \neq 1$ for most n since it would be very unlikely to be matching exactly the same curve; in fact, even if the two curves are the same, the noise in the acquisition provides a random variability on the position of the samples that make $\cos^2\theta_n \neq 1$ with high probability, although $\cos^2\theta_n$ would be very close to 1 in this case. Then, the following properties hold:

Property 4.2.13 $\Delta A_n = \Delta A_0 \prod_{i=0}^{n-1} \dfrac{\cos^2\theta_i}{(1+A_i)(1+A_i+\Delta A_i)}$

Proof: We use induction. From property 4.2.4, we have that $\Delta A_n = \frac{\Delta A_{n-1}\cos^2\theta_{n-1}}{(1+A_{n-1}+\Delta A_{n-1})(1+A_{n-1})}$, then $\Delta A_1 = \frac{\Delta A_0 \cos^2\theta_0}{(1+A_0+\Delta A_0)(1+A_0)}$. The result holds for $n = 1$. Assume that it holds for $k \geq 1$. Therefore, $\Delta A_{k+1} = \frac{\Delta A_k \cos^2\theta_k}{(1+A_k+\Delta A_k)(1+A_k)} = \Delta A_0 \prod_{i=0}^{k} \frac{\cos^2\theta_i}{(1+A_i)(1+A_i+\Delta A_i)}$. The induction is complete. ∎

Property 4.2.14 $B_n = \sum_{j=0}^{n-1}(-1)^{n-j}A_j\Delta_j\left(\prod_{i=j}^{n-1}\dfrac{\cos\theta_i}{(1+A_i)}\right)+(-1)^nB_0\left(\prod_{i=0}^{n-1}\dfrac{\cos\theta_i}{(1+A_i)}\right)$

Proof: We use induction. From equation 4.5, we have that $B_n = -\frac{(A_{n-1}\Delta_{n-1}+B_{n-1})\cos\theta_{n-1}}{1+A_{n-1}}$, then $B_1 = -\frac{A_1\Delta_1\cos\theta_0}{1+A_0} - \frac{B_0\cos\theta_0}{1+A_0}$. The result holds for $n=1$. Assume that it holds for $k \geq 1$. Therefore, $B_{k+1} = -\frac{A_k\Delta_k\cos\theta_k}{1+A_k} - \frac{B_k\cos\theta_k}{1+A_k} = -\frac{A_k\Delta_k\cos\theta_k}{1+A_k} - \frac{\cos\theta_k}{1+A_k}\left(\sum_{j=0}^{k-1}(-1)^{k-j}A_j\Delta_j\prod_{i=j}^{k-1}\frac{\cos\theta_i}{(1+A_i)} + (-1)^kB_0\prod_{i=j}^{k-1}\frac{\cos\theta_i}{(1+A_i)}\right) = \sum_{j=0}^{k}(-1)^{k+1-j}A_j\Delta_j\left(\prod_{i=j}^{k}\frac{\cos\theta_i}{(1+A_i)}\right) + (-1)^{k+1}B_0\left(\prod_{i=0}^{k}\frac{\cos\theta_i}{(1+A_i)}\right)$. The induction is complete. ∎

Property 4.2.15

$$\Delta B_n = \Delta A_0\left(\prod_{j=0}^{n-1}\frac{\cos\theta_j}{(1+A_j+\Delta A_j)}\right)\Bigg[(-1)^n\frac{\Delta B_0}{\Delta A_0} + (-1)^{n-1}B_0\sum_{i=0}^{n-1}\frac{1}{1+A_i}\left(\prod_{j=0}^{i-1}\frac{\cos^2\theta_j}{(1+A_j)^2}\right) + \sum_{i=0}^{n-2}(-1)^{n-1-i}A_i\Delta_i\left(\prod_{m=0}^{i-1}\frac{\cos\theta_m}{(1+A_m)}\right)\sum_{j=i+1}^{n-1}\frac{1}{1+A_j}\left(\prod_{m=i}^{j-1}\frac{\cos^2\theta_m}{(1+A_m)^2}\right) + \sum_{i=0}^{n-1}(-1)^{n-i}\frac{\Delta_i}{1+A_i}\left(\prod_{j=0}^{i-1}\frac{\cos\theta_j}{(1+A_j)}\right)\Bigg]$$

Proof: We use induction. From property 4.2.4, we have that $\Delta B_n = -\frac{\Delta B_{n-1}\cos\theta_{n-1}}{(1+A_{n-1}+\Delta A_{n-1})} - \frac{(\Delta_{n-1}-B_{n-1})\Delta A_{n-1}\cos\theta_{n-1}}{(1+A_{n-1}+\Delta A_{n-1})(1+A_{n-1})}$, then $\Delta B_1 = \Delta A_0\frac{\cos\theta_0}{(1+A_0+\Delta A_0)}\left(-\frac{\Delta B_0}{\Delta A_0} + \frac{B_0}{(1+A_0)} - \frac{\Delta_0}{(1+A_0)}\right)$. The result holds for $n=1$. Assume that it holds for $k \geq 1$. Therefore,

$$\begin{aligned}
\Delta B_{k+1} =& -\frac{(\Delta_k - B_k)\Delta A_k \cos\theta_k}{(1+A_k+\Delta A_k)(1+A_k)} - \frac{\Delta B_k \cos\theta_k}{(1+A_k+\Delta A_k)} \\
=& \Delta A_0 \big(\textstyle\prod_{j=0}^{k} \frac{\cos\theta_j}{(1+A_j+\Delta A_j)}\big)\Big[(-1)\frac{\Delta_k}{1+A_k}\big(\prod_{j=0}^{k-1}\frac{\cos\theta_j}{(1+A_j)}\big) + \\
& \sum_{i=0}^{k-1}(-1)^{k-i}\tfrac{A_i\Delta_i}{1+A_k}\big(\textstyle\prod_{j=i}^{k-1}\frac{\cos^2\theta_j}{(1+A_j)^2}\big)\big(\prod_{j=0}^{i-1}\frac{\cos\theta_j}{(1+A_j)}\big) + (-1)^k\frac{B_0}{1+A_k}\big(\prod_{j=0}^{k-1}\frac{\cos^2\theta_j}{(1+A_j)^2}\big) + \\
& (-1)^k B_0 \sum_{i=0}^{k-1}\tfrac{1}{1+A_i}\big(\textstyle\prod_{j=0}^{k-1}\frac{\cos^2\theta_j}{(1+A_j)^2}\big) + (-1)^{k+1}\frac{\Delta B_0}{\Delta A_0} + \\
& \sum_{i=0}^{k-2}(-1)^{k-i}A_i\Delta_i\big(\textstyle\prod_{m=0}^{i-1}\frac{\cos\theta_m}{(1+A_m)}\big)\displaystyle\sum_{j=i+1}^{n-1}\tfrac{1}{1+A_j}\big(\textstyle\prod_{m=i}^{j-1}\frac{\cos^2\theta_m}{(1+A_m)^2}\big)\Big] \\
=& \Delta A_0\big(\textstyle\prod_{j=0}^{n-1}\frac{\cos\theta_j}{(1+A_j+\Delta A_j)}\big)\Big[(-1)^n\frac{\Delta B_0}{\Delta A_0} + (-1)^{n-1}B_0\displaystyle\sum_{i=0}^{n-1}\tfrac{1}{1+A_i}\big(\textstyle\prod_{j=0}^{i-1}\frac{\cos^2\theta_j}{(1+A_j)^2}\big) + \\
& \sum_{i=0}^{n-2}(-1)^{n-1-i}A_i\Delta_i\big(\textstyle\prod_{m=0}^{i-1}\frac{\cos\theta_m}{(1+A_m)}\big)\displaystyle\sum_{j=i+1}^{n-1}\tfrac{1}{1+A_j}\big(\textstyle\prod_{m=i}^{j-1}\frac{\cos^2\theta_m}{(1+A_m)^2}\big) + \\
& \sum_{i=0}^{n-1}(-1)^{n-i}\tfrac{\Delta_i}{1+A_i}\Big(\textstyle\prod_{j=0}^{i-1}\frac{\cos\theta_j}{(1+A_j)}\Big)\Big]
\end{aligned}$$

The induction is complete. ∎

Property 4.2.16 $\Delta C_n = \Delta C_0 + \sum_{i=0}^{n-1}\left(\frac{(B_i-\Delta_i)^2\Delta A_i}{(1+A_i)(1+A_i+\Delta A_i)} - \frac{2(B_i-\Delta_i)\Delta B_i + \Delta B_i^2}{(1+A_i+\Delta A_i)}\right)$

Proof: We use induction. From property 4.2.4, we have that $\Delta C_n = \Delta C_{n-1} + \frac{(B_{n-1}-\Delta_{n-1})^2\Delta A_{n-1}}{(1+A_{n-1}+\Delta A_{n-1})(1+A_{n-1})} - \frac{2(B_{n-1}-\Delta_{n-1})\Delta B_{n-1}+\Delta B_{n-1}{}^2}{(1+A_{n-1}+\Delta A_{n-1})}$, then $\Delta C_1 = \Delta C_0 + \frac{(B_0-\Delta_0)^2\Delta A_0}{(1+A_0+\Delta A_0)(1+A_0)} - \frac{2(B_0-\Delta_0)\Delta B_0+\Delta B_0{}^2}{(1+A_0+\Delta A_0)}$. The result holds for $n=1$. Assume that it holds for $k \geq 1$. Therefore, $\Delta C_{k+1} = \Delta C_k + \frac{(B_k-\Delta_k)^2\Delta A_k}{(1+A_k+\Delta A_k)(1+A_k)} - \frac{2(B_k-\Delta_k)\Delta B_k+\Delta B_k{}^2}{(1+A_k+\Delta A_k)} = \Delta C_0 + \sum_{i=0}^{k}\big(\frac{(B_i-\Delta_i)^2\Delta A_i}{(1+A_i)(1+A_i+\Delta A_i)} - \frac{2(B_i-\Delta_i)\Delta B_i+\Delta B_i^2}{(1+A_i+\Delta A_i)}\big)$. The induction is complete. ∎

As we observe from the previous properties, the asymptotic behavior of the coefficients of $f_n(r)$ depends upon the behavior of $\prod_{j=0}^{n}(1+A_j)$. Let's call $P_n = \prod_{j=0}^{n}(1+A_j)$, then $P_n > P_{n-1}$ since $P_n = (1+A_n)P_{n-1}$ and $0 < A \leq 1$. Thus, P_n is a monotonically increasing function of n.

Property 4.2.17 *P_n behaves according to the following difference equation* $P_n - 2P_{n-1} + (1-\sin^2\theta_{n-1})P_{n-2} = 0$.

Proof: From equation 4.2.4 we have that $A_n = \frac{A_{n-1}+\sin^2\theta_{n-1}}{1+A_{n-1}}$. It can be shown by induction that $A_n = \sum_{i=0}^{n-1}\frac{\sin^2\theta_i P_{i-1}}{P_{n-1}} + \frac{A_0}{P_{n-1}}$. In fact, $A_1 = \frac{\sin^2\theta_0}{1+A_0} + \frac{A_0}{1+A_0} = \frac{\sin^2\theta_0 P_{-1}}{P_0} + \frac{A_0}{P_0}$, with $P_{-1} = 1$. Assuming valid for $k \geq 1$, $A_{k+1} = \frac{A_k+\sin^2\theta_k}{1+A_k} = \sum_{i=0}^{k}\frac{\sin^2\theta_i P_{i-1}}{P_k} + \frac{A_0}{P_k}$.

Now, let us look at $P_n = \prod_{j=0}^{n}(1+A_j) = (1+A_n)P_{n-1}$, then replacing by the equation for A_n described in the previous paragraph, $P_n = P_{n-1} + A_0 + \sum_{i=0}^{n-1}\sin^2\theta_i P_{i-1}$, with the initial condition that $P_{-1} = 1$. Thus, $\Delta P_n = P_n - P_{n-1} = A_0 + \sum_{i=0}^{n-1}\sin^2\theta_i P_{i-1}$ and $\Delta\Delta P_n = (P_n - P_{n-1}) - (P_{n-1} - P_{n-2}) = \sum_{i=0}^{n-1}\sin^2\theta_i P_{i-1} - \sum_{i=0}^{n-2}\sin^2\theta_i P_{i-1} = \sin^2\theta_{n-1}P_{n-2}$. Finally, $P_n - 2P_{n-1} + (1-\sin^2\theta_{n-1})P_{n-2} = 0$. ∎

Let us analyze a few solutions of the differential equation for P_n. As we pointed out before, the case in which $\sin^2\theta_n = 0$, $\forall n$, i.e., $\cos^2\theta_n = 1$, $\forall n$, simplifies the problem as seen in equation 4.13. The difference equation for P_n has two equal roots and the solution is simply $P_n = 1 + (n+1)A_0$. Therefore, P_n diverges linearly in this case. Let us solve for P_n when $\sin^2\theta_n$ is a constant different from zero.

Property 4.2.18 *For* $\sin^2\theta_n = \gamma \neq 0$, $\forall n$ *is* $P_n = (\frac{1+A_0}{2} + \frac{\sqrt{\gamma}}{2}(1+\frac{A_0}{\gamma}))(1+\sqrt{\gamma})^n + (\frac{1+A_0}{2} - \frac{\sqrt{\gamma}}{2}(1+\frac{A_0}{\gamma}))(1-\sqrt{\gamma})^n$.

Proof: If $\sin^2\theta_n = \gamma \neq 0$, $\forall n$, then the difference equation for P_n is the following:

$$\begin{cases} P_n - 2P_{n-1} + (1-\gamma)P_{n-2} = 0 \\ P_{-1} = 1 \\ P_0 = 1 + A_0 \end{cases}$$

The solution of the above difference equation is of the type $P_n = \alpha^n$ that gives rise to the following quadratic equation on α, $\alpha^2 - 2\alpha + (1-\gamma) = 0$, whose solutions are $\alpha_{1,2} = 1 \pm \sqrt{\gamma} = 1 \pm |\sin\theta|$. The general form of the solution for P_n is $P_n = c_1\alpha_1 + c_2\alpha_2$ where the coefficients c_1 and c_2 are defined by the initial conditions: $P_{-1} = 1 = c_1\alpha_1^{-1} + c_2\alpha_2^{-1}$ and $P_0 = 1 + A_0 = c_1 + c_2$. The solution of this system of two equations on two unknowns is $c_1 = \frac{1+A_0}{2} + \frac{\sqrt{\gamma}}{2}(1+\frac{A_0}{\gamma})$ and $c_2 = \frac{1+A_0}{2} - \frac{\sqrt{\gamma}}{2}(1+\frac{A_0}{\gamma})$, that completes the proof. ∎

The behavior of P_n for large n in this case is defined by the evolution of the term $(1+\gamma)^n$ that diverges exponentially. The rate of divergence is directly associated with the value of γ. In practical terms, having $\sin^2\theta_n = \gamma \neq 0,\ \forall n$ is quite unlikely, unless we match one curve against a version of itself rotated by an angle equal to $\sin^{-1}(\sqrt{\gamma})$.

Property 4.2.19 *For* $\sin^2\theta_n = 1,\ \forall n$ *is* $P_n = 2^n$.

Proof: If $\sin^2\theta_n = 1,\ \forall n$, then $A_n = 1,\ \forall n$ and, therefore,
$P_n = \prod_{i=0}^{n}(1 + A_i) = \prod_{i=0}^{n}(1 + 1) = 2^n$. ∎

From the previous solutions for P_n, we infer that its general behavior is bounded by an increasing linear function and an exponential function of type 2^n. The case of the linear solution corresponds to having $\sin^2\theta_n = 0,\ \forall n$, but in practical situations it is quite difficult to get this case since the noise in the acquisition would make it quite unlikely since measuring the same coordinates for the same point has probability zero. Besides, it is also quite unlikely to have many segments for which $\sin^2\theta_n = 0$; therefore, it is reasonable to expect that $\sin^2\theta_n \neq 0$ for almost all n. In the case in which $\sin^2\theta_n = 0$ a few times, the following properties describe the asymptotic behavior of P_n after the last n for which $\sin^2\theta_n = 0$.

Let us analyze the case in which $\sin^2\theta_n > 0,\ \forall n$. The two cases analyzed in properties 4.2.18, 4.2.19 are the extreme possible behaviors of P_n. In general, P_n should behave in between these two extreme cases, still diverging exponentially but at an unknown rate.

Property 4.2.20 *Assume that* $\sin^2\theta_n > 0,\ \forall n$, *say* $\sin^2\theta_n \geq \epsilon,\ \forall n$, *let us call* P_n^ϵ

the solution for P_n *when* $\sin^2\theta_n = \epsilon,\ \forall n,$ *then* $P_n \geq P_n^\epsilon,\ \forall n.$

Proof: We use induction. The initial conditions for P_n and P_n^ϵ are $P_{-1} = P_{-1}^\epsilon = 1$ and $P_0 = P_0^\epsilon = 1 + A_0$. Take $\sin^2\theta_0 \geq \epsilon$, then $P_1 = 2P_0 - (1-\sin^2\theta_0)P_{-1} \geq 2P_0^\epsilon - (1-\epsilon)P_{-1}^\epsilon = P_1^\epsilon$. The result holds for $n = 1$. Assume that it holds for $k \geq 1$. Thus, $P_k \geq P_k^\epsilon$ and $P_{k-1} \geq P_{k-1}^\epsilon$. Take $\sin^2\theta_k \geq \epsilon$, then $-(1-\sin^2\theta_k)P_{k-1} \geq -(1-\epsilon)P_{k-1}^\epsilon$ and $P_{k+1} = 2P_k - (1-\sin^2\theta_k)P_{k-1} \geq 2P_k^\epsilon - (1-\epsilon)P_{k-1}^\epsilon = P_{k+1}^\epsilon$. The induction is complete. ∎

Let us study the asymptotic behavior of ΔA, ΔB, and ΔC. Property 4.2.13 shows that $\Delta A_n = \Delta A_0 \prod_{i=0}^{n-1} \frac{\cos^2\theta_i}{(1+A_i)(1+A_i+\Delta A_i)}$ and, therefore, $\Delta A_n \underset{n\to\infty}{\longrightarrow} 0$ exponentially. The following properties describe the behavior of ΔB_n and ΔC_n.

Property 4.2.21 $|\Delta B_n| \underset{n\to\infty}{\longrightarrow} 0$

Proof: Let us assume that $\Delta_n \leq \Delta_{max},\ \forall n,\ P_n = \prod_{i=0}^n (1+A_i) \geq K_0\varepsilon^n,\ \varepsilon > 1$ and $P'_n = \prod_{i=0}^n (1+A_i+\Delta A_i) \geq K'_0\varepsilon'^n,\ \varepsilon' > 1$. Let us recall that $0 \leq |\cos\theta_i| \leq 1,\ \forall i$ and $0 < A_i \leq 1,\ \forall i$. Then,

$$
\begin{aligned}
|\Delta B_n| \leq |\Delta A_0| \frac{\prod_{j=0}^{n-1}\cos\theta_j}{P'_n} \Bigg[& \frac{|\Delta B_0|}{|\Delta A_0|} + |B_0| \sum_{i=0}^{n-1} \frac{1}{1+A_i} \frac{\prod_{j=0}^{i-1}\cos^2\theta_j}{P_{i-1}^2} + \sum_{i=0}^{n-1} \frac{\Delta_i}{1+A_i} \frac{\prod_{j=0}^{i-1}|\cos\theta_j|}{P_{i-1}} + \\
& \sum_{i=0}^{n-2} (-1)^{n-1-i} A_i \Delta_i \frac{\prod_{m=0}^{i-1}|\cos\theta_m|}{P_{i-1}} \sum_{j=i+1}^{n-1} \frac{1}{1+A_j} \frac{P_{i-1}^2}{P_{j-1}^2} \left(\prod_{m=i}^{j-1}\cos^2\theta_m\right) \Bigg] \leq \\
& \frac{|\Delta A_0|}{K'_0} \varepsilon'^{-n+1} \left[\frac{|\Delta B_0|}{|\Delta A_0|} + \frac{|B_0|}{K_0^2} \sum_{i=0}^{n-1} \varepsilon^{-2i} + \frac{\Delta_{max}}{K_0} \sum_{i=0}^{n-1} \varepsilon^{-i} + \frac{\Delta_{max}}{K_0} \sum_{i=0}^{n-2} \varepsilon^{-(i-1)} \sum_{j=i+1}^{n-1} \varepsilon^{-2(j-i)} \right] \\
\leq & \frac{|\Delta A_0|}{K'_0} \varepsilon'^{-n+1} \Bigg[\frac{|\Delta B_0|}{|\Delta A_0|} + \frac{|B_0|\varepsilon}{K_0^2} \frac{1-\varepsilon^{-2n}}{1-\varepsilon^{-2}} + \frac{\Delta_{max}}{K_0} \frac{1-\varepsilon^{-n}}{1-\varepsilon^{-1}} + \\
& \frac{\Delta_{max}}{K_0} \frac{\varepsilon^{-2}-\varepsilon^{-4}-\varepsilon^{-8}+\varepsilon^{-9}-\varepsilon^{-n-1}+\varepsilon^{-n-3}+\varepsilon^{-2n-6}-\varepsilon^{-2n-7}}{(1-\varepsilon^{-1})(1-\varepsilon^{-2})^2} \Bigg] \underset{n\to\infty}{\longrightarrow} 0 \quad \blacksquare
\end{aligned}
$$

Property 4.2.22 $|\Delta C_n| \underset{n\to\infty}{\longrightarrow}$ *constant*

Proof: Let us assume that $\Delta_n \leq \Delta_{max}$, $\forall n$, $P_n = \prod_{i=0}^{n}(1+A_i) \geq K_0 \varepsilon^n$, $\varepsilon > 1$ and $P'_n = \prod_{i=0}^{n}(1+A_i+\Delta A_i) \geq K'_0 \varepsilon'^n$, $\varepsilon' > 1$. Let us recall that $0 \leq |\cos\theta_i| \leq 1$, $\forall i$ and $0 < A_i \leq 1$, $\forall i$. Then we need bounds for $|B_i|$ and $|\Delta B_i|$ since

$$|\Delta C_n| \leq |\Delta C_0| + \sum_{i=0}^{n-1}\left(\frac{(B_i+\Delta_1)^2|\Delta A_i|}{(1+A_i)(1+A_i+\Delta A_i)} + \frac{2(|B_i|+\Delta_i)|\Delta B_i|+\Delta B_i^2}{(1+A_i+\Delta A_i)}\right)$$

$$B_i = \sum_{j=0}^{i-1}(-1)^{i-j}A_j\Delta_j\left(\prod_{m=j}^{i-1}\cos\theta_m\right)\frac{P_{j-1}}{P_{i-1}} + (-1)^i B_0 \frac{\prod_{m=j}^{i-1}\cos\theta_m}{P_{i-1}}$$

$$\Rightarrow |B_i| \leq \frac{|B_0|}{P_{i-1}} + \frac{\Delta_{max}}{P_{i-1}}\sum_{j=0}^{i-1}\varepsilon^{j-1} = \frac{|B_0|}{K_0}\varepsilon^{-(i-1)} + \Delta_{max}\varepsilon^{-i}\frac{\varepsilon^i-1}{\varepsilon-1} =$$

$$\frac{|B_0|}{K_0}\varepsilon^{-(i-1)} + \Delta_{max}\frac{1-\varepsilon^{-i}}{\varepsilon-1} = \frac{\Delta_{max}}{\varepsilon-1} + \left(\frac{\varepsilon|B_0|}{K_0} - \frac{\Delta_{max}}{\varepsilon-1}\right)\varepsilon^{-i} = N_1 + N_2\varepsilon^{-i}$$

$$|\Delta B_i| \leq \frac{|\Delta A_0|}{K'_0}\,\varepsilon'^{-i+1}\left[\frac{|\Delta B_0|}{|\Delta A_0|} + \frac{|B_0|\varepsilon}{K_0^2}\frac{1-\varepsilon^{-2i}}{1-\varepsilon^{-2}} + \frac{\Delta_{max}}{K_0}\frac{1-\varepsilon^{-i}}{1-\varepsilon^{-1}} + \right.$$

$$\left.\frac{\Delta_{max}}{K_0}\frac{\varepsilon^{-2}-\varepsilon^{-4}-\varepsilon^{-8}+\varepsilon^{-9}-\varepsilon^{-i-1}+\varepsilon^{-i-3}+\varepsilon^{-2i-6}-\varepsilon^{-2i-7}}{(1-\varepsilon^{-1})(1-\varepsilon^{-2})^2}\right] =$$

$$M_1\varepsilon'^{-i} + M_2(\varepsilon'\varepsilon)^{-i} + M_3(\varepsilon'\varepsilon^2)^{-i}$$

$$|\Delta C_n| \leq |\Delta C_0| + \sum_{i=0}^{n-1}\Bigg(\tfrac{|\Delta A_0|}{K_0 K_0'}(N_1 + N_2\varepsilon^{-i} + \Delta_{max})^2(\varepsilon'\varepsilon)^{-i} +$$
$$2{\varepsilon'}^{-1}(N_1 + N_2\varepsilon^{-i} + \Delta_{max})(M_1{\varepsilon'}^{-i} + M_2(\varepsilon'\varepsilon)^{-i} + M_3(\varepsilon'\varepsilon^2)^{-i}) +$$
$${\varepsilon'}^{-1}(M_1{\varepsilon'}^{-i} + M_2(\varepsilon'\varepsilon)^{-i} + M_3(\varepsilon'\varepsilon^2)^{-i})^2\Bigg) =$$
$$|\Delta C_0| + \left(\tfrac{|\Delta A_0|}{K_0K_0'}(\tfrac{\Delta_{max}}{1-\varepsilon^{-1}})^2 + 2M_2\tfrac{{\varepsilon'}^{-1}\Delta_{max}}{1-\varepsilon^{-1}} + 2{\varepsilon'}^{-1}N_2M_1\right)\tfrac{1-(\varepsilon'\varepsilon)^{-n}}{1-(\varepsilon'\varepsilon)^{-1}} +$$
$$\left(2N_2\tfrac{|\Delta A_0|}{K_0K_0'}\tfrac{\Delta_{max}}{1-\varepsilon^{-1}} + 2M_3\tfrac{{\varepsilon'}^{-1}\Delta_{max}}{1-\varepsilon^{-1}} + 2{\varepsilon'}^{-1}N_2M_2\right)\tfrac{1-(\varepsilon'\varepsilon^2)^{-n}}{1-(\varepsilon'\varepsilon^2)^{-1}} +$$
$$\left(2N_2^2\tfrac{|\Delta A_0|}{K_0K_0'} + 2{\varepsilon'}^{-1}N_2M_3\right)\tfrac{1-(\varepsilon'\varepsilon^3)^{-n}}{1-(\varepsilon'\varepsilon^3)^{-1}} + 2M_1\tfrac{{\varepsilon'}^{-1}\Delta_{max}}{1-\varepsilon^{-1}}\tfrac{1-{\varepsilon'}^{-n}}{1-{\varepsilon'}^{-1}} + {\varepsilon'}^{-1}M_1^2\tfrac{1-({\varepsilon'}^2)^{-n}}{1-({\varepsilon'}^2)^{-1}}$$
$$(M_2^2 + 2M_1M_3)\tfrac{1-({\varepsilon'}^2\varepsilon^2)^{-n}}{1-({\varepsilon'}^2\varepsilon^2)^{-1}} + {\varepsilon'}^{-1}M_3^2\tfrac{1-({\varepsilon'}^2\varepsilon^4)^{-n}}{1-({\varepsilon'}^2\varepsilon^4)^{-1}} + 2{\varepsilon'}^{-1}M_1M_2\tfrac{1-({\varepsilon'}^2\varepsilon)^{-n}}{1-({\varepsilon'}^2\varepsilon)^{-1}} +$$
$$2{\varepsilon'}^{-1}M_2M_3\tfrac{1-({\varepsilon'}^2\varepsilon^3)^{-n}}{1-({\varepsilon'}^2\varepsilon^3)^{-1}} \underset{n\to\infty}{\longrightarrow} \text{constant} \quad \blacksquare$$

Let us call $x_{1,2}$ the zeros of $f(r) = \mathcal{D}^{(1)}(r) - \mathcal{D}^{(2)}(r)$. The corresponding equations are $x_1 = -\ \mathrm{sign}(\Delta B)(\frac{|\Delta B|}{\Delta A} + \frac{\sqrt{\Delta B^2 - \Delta A\Delta C}}{\Delta A})$ and $x_2 = -\ \mathrm{sign}(\Delta B)\frac{\Delta C}{|\Delta B| + \sqrt{\Delta B^2 - \Delta A \Delta C}}$. The zeros of $f(r)$ define the interval in which $\mathcal{D}^{(1)}(r) \leq \mathcal{D}^{(2)}(r)$, if $\Delta A > 0$ (or the interval in which $\mathcal{D}^{(1)}(r) \geq \mathcal{D}^{(2)}(r)$, if $\Delta A < 0$), so if the position of $x_{1,2}$ diverge as the algorithm goes through the iterations, it would be possible to find out which parabola provides the minimum cost.

Property 4.2.23 $|x_{1_n}| \underset{n\to\infty}{\longrightarrow} \infty$ *and* $|x_{2_n}| \underset{n\to\infty}{\longrightarrow} \infty$ *if* $|\Delta C_n| \underset{n\to\infty}{\not\rightarrow} 0$.

Proof: Let us assume that $P_n = \prod_{i=0}^n(1 + A_i) \geq K_0\varepsilon^n$, $\varepsilon > 1$ and $P_n' = \prod_{i=0}^n(1 + A_i + \Delta A_i) \geq K_0'{\varepsilon'}^n$, $\varepsilon' > 1$. Let us recall that $0 \leq |\cos\theta_i| \leq 1$, $\forall i$. For $|x_{1_n}| = \frac{|\Delta B|}{|\Delta A|} + \frac{\sqrt{\Delta B^2 - \Delta A\Delta C}}{|\Delta A|}$ we have

$$|x_{1_n}| = \frac{|\Delta B_n|}{|\Delta A_n|} + \frac{\sqrt{\Delta B_n{}^2 - \Delta A_n \Delta C_n}}{|\Delta A_n|} \geq \frac{\sqrt{\Delta B_n{}^2 - \Delta A_n \Delta C_n}}{|\Delta A_n|} \geq \sqrt{\frac{\Delta B_0{}^2 - \Delta A_0 \Delta C_0}{\Delta A_0^2}}\sqrt{P_{n-1}P'_{n-1}}$$
$$\geq \sqrt{\frac{\Delta B_0{}^2 - \Delta A_0 \Delta C_0}{\Delta A_0^2}}\sqrt{K_0 \varepsilon^{n-1} K'_0 \varepsilon'^{n-1}} \underset{n\to\infty}{\longrightarrow} \infty.$$

For $|x_{2_n}| = \frac{|\Delta C_n|}{|\Delta B_n| + \sqrt{\Delta B_n{}^2 - \Delta A_n \Delta C_n}}$, we study the numerator and the denominator separately. From property 4.2.22 we have that $|\Delta C_n| \underset{n\to\infty}{\longrightarrow}$ constant. For the denominator, we have

$$|\Delta B_n| + \sqrt{\Delta B_n{}^2 - \Delta A_n \Delta C_n} \leq M_1 \varepsilon'^{-n} + M_2(\varepsilon'\varepsilon)^{-n} + M_3(\varepsilon'\varepsilon^2)^{-n} +$$
$$\sqrt{\frac{\Delta B_0{}^2 - \Delta A_0 \Delta C_0}{\Delta A_0^2 K_0 K'_0}}\left(\sqrt{\varepsilon'\varepsilon}\right)^{-n+1} \underset{n\to\infty}{\longrightarrow} 0.$$

Therefore, $|x_{2_n}| \underset{n\to\infty}{\longrightarrow} \infty$ if $|\Delta C_n| \underset{n\to\infty}{\not\rightarrow} 0$. ∎

The equations corresponding to a warping path consisting only of transitions of type 1,2 are much more complicated than the equations corresponding to a warping path consisting only of transitions of type 3,4. In the latter case, we have closed form solution for all the expressions and we found a condition relating the initial conditions of the parabolas and Δ_{max} that makes sure that x_{1_n} and x_{2_n} have different sign throughout the iterations. In the present case, it is much more difficult to derive such a condition. However, we proved that $|\Delta C|$ tends to a constant exponentially fast, so we can check at each step in the algorithm whether ΔC is reaching a limit, oscillating or tending to zero and then proceed to make the corresponding decision.

Whether there exists a theoretical bound on the average of the maximum number of parabolic functions that need to be stored for a given set of curves is still an open research area. However, we found experimentally that this bound exists and it is a function of the number of samples in each of the curves and the constraints imposed on the warping plane, as we will shown in the experiments.

4.2.6 Interval range propagation

The equations derived in section 4.2.1 describe how to propagate different cumulated distance functions throughout the warping plane. These equations are derived considering that the warping function is defined for all real numbers. However, the range of the curvilinear variables x, y is restricted to $[0, \Delta_{x,y}]$ and we need to derive the propagation equations under these restrictions. We describe the equations for cases 1 and 3.

- First step, case 1: we have shown in equation 4.4 that $\mathcal{D}_{(2,2)}(y) = y^2(1-\cos^2\theta_{22})$ and $x' = \Delta_{x_2} - y\cos\theta_{22}$, but considering the above restrictions, the equations are the following:

  1. $\cos\theta_{22} \leq 0: \quad \mathcal{D}_{(2,2)}(y) = y^2,\ y \in [0, \Delta_y],\ x' = \Delta_{x_2}$
  2. $\cos\theta_{22} > 0:$
     (a) $\Delta_{x_2} - \Delta_{y_2}\cos\theta_{22} \geq 0: \quad \mathcal{D}_{(2,2)}(y) = (1-\cos^2\theta_{22})y^2,$
     $y \in [0, \Delta_y],\ x' = \Delta_{x_2} - y\cos\theta_{22}$
     (b) $\Delta_{x_2} - \Delta_{y_2}\cos\theta_{22} < 0: \quad \mathcal{D}_{(2,2)}(y) = (1-\cos^2\theta_{22})y^2,$
     $y \in [0, \frac{\Delta_{x_2}}{\cos\theta_{22}}],\ x' = \Delta_{x_2} - y\cos\theta_{22}$

- First step, case 3: we have shown in equation 4.8 that $\mathcal{D}_{(2,2)}(x) = \Delta_y^2(1-\cos^2\theta_{22})$ and $x' = x - \Delta_{y_2}\cos\theta_{22}$, but considering the restrictions, the equations are the following:

  1. $\cos\theta_{22} = 0: \quad \mathcal{D}_{(2,2)}(x) = \Delta_{y_2}^2,\ x \in [0, \Delta_{x_2}],\ x' = x$
  2. $\cos\theta_{22} < 0:$
     (a) $\Delta_{x_2} + \Delta_{y_2}\cos\theta_{22} \leq 0:$
     $\mathcal{D}_{(2,2)}(x) = x^2 - 2(\Delta_{x_2} + \Delta_{y_2}\cos\theta_{22})x + (\Delta_{x_2}^2 + \Delta_y^2 + 2\cos\theta_{22}\Delta_{x_2}\Delta_y),$
     $x \in [0, \Delta_{x_2}],\ x' = \Delta_{x_2}$
     (b) $\Delta_{x_2} + \Delta_{y_2}\cos\theta_{22} > 0:$
     $\mathcal{D}_{(2,2)}(x) = (1-\cos^2\theta_{22})\Delta_{y_2}^2,\ x \in [0, \Delta_{x_2} + \Delta_{y_2}\cos\theta_{22}],$
     $x' = x - \Delta_{y_2}\cos\theta_{22}$

3. $\cos\theta_{22} > 0$:

   (a) $\Delta_{x_2} - \Delta_{y_2}\cos\theta_{22} \leq 0$:

   $\mathcal{D}_{(2,2)}(x) = x^2 - 2\Delta_{y_2} x\cos\theta_{22} + \Delta_{y_2}^2,\ x \in [0, \Delta_{x_2}],\ x' = 0$

   (b) $\Delta_{x_2} - \Delta_{y_2}\cos\theta_{22} > 0$:

   $\mathcal{D}_{(2,2)}(x) = (1-\cos^2\theta_{22})\Delta_{y_2}^2,\ x \in [\Delta_y\cos\theta_{22}, \Delta_{x_2}],\ x' = x - \Delta_{y_2}\cos\theta_{22}$

- General step, case 1: we have shown in equation 4.5 that $\mathcal{D}_{(i,j)}(y) = A_y y^2 + 2B_y y + C_y$, with $A_y = \frac{A_{x'}+1-\cos^2\theta_{ij}}{A_{x'}+1}$, $B_y = -\frac{\cos\theta_{ij}(A_{x'}\Delta_{x_i}+B_{x'})}{A_{x'}+1}$, $C_y = C_{x'} + \Delta_{x_i}^2 - \frac{(B_{x'}-\Delta_{x_i})^2}{A_{x'}+1}$, and $x' = \frac{-y\cos\theta_{ij}+(\Delta_{x_i}-B_{x'})}{A_{x'}+1}$. Given that $y \in [0, \Delta_{y_j}]$, then

$$x' \in [x_1', x_2'], \text{with} \begin{cases} x_1' = \frac{(\Delta_{x_i}-B_{x'})}{A_{x'}+1} & x_2' = \frac{\Delta_{x_i}-B_{x'}-\Delta_{y_j}\cos\theta_{ij}}{A_{b_1}+1} \quad \text{if } \cos\theta_{ij} < 0 \\ x_1' = \frac{\Delta_{x_i}-B_{x'}-\Delta_{y_j}\cos\theta_{ij}}{A_{x'}+1} & x_2' = \frac{(\Delta_{x_i}-B_{x'})}{A_{x'}+1} \quad \text{if } \cos\theta_{ij} > 0 \end{cases}$$

The restriction of the excursion of the curvilinear variables from previous iterations is such that $x' \in [I_1, I_2]$. Then the corresponding values of y are $y_1 = \frac{(\Delta_{x_i}-B_{x'})-(A_{x'}+1)x_1'}{\cos\theta_{ij}}$, $y_2 = \frac{(\Delta_{x_i}-B_{x'})-(A_{x'}+1)x_2'}{\cos\theta_{ij}}$, if $\cos\theta_{ij} \neq 0$. The propagation equations taking in consideration the restrictions on x' and y are the following:

1. $x_1' \geq I_2$:

   $\mathcal{D}_{(i,j)}(y) = y^2 - 2y(\Delta_{x_i} - I_2)\cos\theta_{ij} + (\Delta_{x_i}^2 - I_2)^2 + A_{x'}I_2^2 + 2B_{x'}R_2 + C_{x'}$,

   $y \in [0, \Delta_{y_j}],\ x' = I_2$

2. $I_1 \leq x_1' < I_2 \ \wedge\ x_2' > I_2$:

$$\mathcal{D}_{(i,j)}(y) = A_y y^2 + 2B_y y + C_y,\ x' = \frac{-y\cos\theta_{ij}+(\Delta_{x_i}-B_{x'})}{A_{x'}+1} \begin{cases} y \in [0, y_2], & \text{if } \cos\theta_{ij} < 0 \\ y \in [y_2, \Delta_{y_j}], & \text{if } \cos\theta_{ij} > 0 \\ y \in [0, \Delta_{y_j}], & \text{if } \cos\theta_{ij} = 0 \end{cases}$$

3. $x'_1 < I_1 \ \wedge \ x'_2 > I_2$:

$$\mathcal{D}_{(i,j)}(y) = A_y y^2 + 2B_y y + C_y, \ x' = \frac{-y\cos\theta_{ij} + (\Delta_1 - B_{x'})}{A_{x'}+1} \begin{cases} y \in [y_1, y_2], & \text{if } \cos\theta_{ij} < 0 \\ y \in [y_2, y_1], & \text{if } \cos\theta_{ij} > 0 \\ y \in [0, \Delta_{y_j}], & \text{if } \cos\theta_{ij} = 0 \end{cases}$$

4. $x'_1 < I_1 \ \wedge \ I_1 < x'_2 \leq I_2$:

$$\mathcal{D}_{(i,j)}(y) = A_y y^2 + 2B_y y + C_y, \ x' = \frac{-y\cos\theta_{ij} + (\Delta_1 - B_{x'})}{A_{x'}+1} \begin{cases} y \in [y_1, \Delta_{y_j}], & \text{if } \cos\theta_{ij} < 0 \\ y \in [0, y_1], & \text{if } \cos\theta_{ij} > 0 \\ y \in [0, \Delta_{y_j}], & \text{if } \cos\theta_{ij} = 0 \end{cases}$$

5. $x'_1 \geq I_1 \ \wedge \ x'_2 \leq I_2$:
   $\mathcal{D}_{(i,j)}(y) = A_y y^2 + 2B_y y + C_y, \ x' = \frac{-y\cos\theta_{ij} + (\Delta_1 - B_{x'})}{A_{x'}+1}, \ y \in [0, \Delta_{y_j}]$

6. $x'_2 \leq I_1$:
   $\mathcal{D}_{(i,j)}(y) = y^2 - 2(\Delta_1 - I_1)y\cos\theta_{ij} + (\Delta_1^2 - I_1)^2 + A_{x'}I_1^2 + 2B_{x'}I_1 + C_{x'}$,
   $x' = I_1, \ y \in [0, \Delta_{y_j}]$

- General step, case 3: we have shown in equation 4.10 that $\mathcal{D}_{(i,j)}(x) = A_x x^2 + 2B_x x + C_x$, with $A_x = \frac{A_{x'}}{A_{x'}+1}$, $B_x = \frac{B_{x'} - A_{x'}\Delta_{y_j}\cos\theta_{ij}}{A_{x'}+1}$, $C_x = C_{x'} + \Delta_{y_j}^2 - \frac{(B_{x'} + \Delta_{y_j}\cos\theta_{ij})^2}{A_{x'}+1}$, and $x' = \frac{x - \Delta_{y_j}\cos\theta_{ij} - B_{x'}}{A_{x'}+1}$. Given that $x \in [0, \Delta_{x_i}]$, then $x' \in [x'_1, x'_2]$, with $x'_1 = \frac{-\Delta_{y_j}\cos\theta_{ij} - B_{x'}}{A_{x'}+1}$, $x'_2 = \frac{\Delta_{x_i} - \Delta_{y_j}\cos\theta_{ij} - B_{x'}}{A_{x'}+1}$. The restriction of the excursion of the curvilinear variables from previous iterations is such that $x' \in [I_1, I_2]$. Then the corresponding values of x are $x_1 = \Delta_{y_j}\cos\theta_{ij} + B_{x'} + (A_{x'}+1)I_1$, $x_2 = \Delta_{y_j}\cos\theta_{ij} + B_{x'} + (A_{x'}+1)I_2$. The propagation equations taking in consideration the restrictions on x' and x are the following:

  1. $x'_1 \geq I_2$:
     $\mathcal{D}_{(i,j)}(x) = x^2 - 2(I_2 + \Delta_{y_j}\cos\theta_{ij})x + (A_{x'}+1)I_2^2 + 2(B_{x'} + \Delta_{y_j}\cos\theta_{ij})I_2 +$

$C_{x'} + \Delta_{y_j}^2,$

$x \in [0, \Delta_{x_i}],\ x' = I_2$

2. $I_1 \leq x'_1 < I_2\ \wedge\ x'_2 > I_2:$

   $\mathcal{D}_{(i,j)} = A_x x^2 + 2B_x x + C_x,\ x \in [0, x_2],\ x' = \frac{x - \Delta_{y_j}\cos\theta_{ij} - B_{x'}}{A_{x'}+1}$

3. $x'_1 < I_1\ \wedge\ x'_2 > I_2:$

   $\mathcal{D}_{(i,j)}(x) = A_x x^2 + 2B_x x + C_x,\ x \in [x_1, x_2],\ x' = \frac{x - \Delta_{y_j}\cos\theta_{ij} - B_{x'}}{A_{x'}+1}$

4. $x'_1 < I_1\ \wedge\ I_1 < x'_2 \leq I_2:$

   $\mathcal{D}_{(i,j)}(x) = A_x x^2 + 2B_x x + C_x,\ x \in [x_1, \Delta_{x_i}],\ x' = \frac{x - \Delta_{y_j}\cos\theta_{ij} - B_{x'}}{A_{x'}+1}$

5. $x'_1 \geq I_1\ \wedge\ x'_2 \leq I_2:$

   $\mathcal{D}_{(i,j)}(x) = A_x x^2 + 2B_x x + C_x,\ x \in [0, \Delta_{x_i}],\ x' = \frac{x - \Delta_{y_j}\cos\theta_{ij} - B_{x'}}{A_{x'}+1}$

6. $x'_2 \leq I_1:$

   $\mathcal{D}_{(i,j)}(x) = x^2 - 2(I_1 + \Delta_{y_j}\cos\theta_{ij})x + (A_{x'}+1)I_1^2 + 2(B_{x'} + \Delta_{y_j}\cos\theta_{ij})I_1 +$

   $C_{x'} + \Delta_{y_j}^2,$

   $x \in [0, \Delta_{x_i}],\ x' = I_1$

Having derived the propagation equations that incorporate the constraints on the excursion of the curvilinear coordinates, the remaining question is whether corollaries 4.2.2 and 4.2.4 hold, and therefore, whether it is possible to discard cumulative distances in the case in which the constraints in the excursion of the curvilinear coordinates alters the normal propagation of the cumulative distance throughout the warping plane. Given a distance function $\mathcal{D}'(r')$ at certain step in the algorithm, where r' is corresponds to either x' or y', and given that the curvilinear coordinate r' is constrained to a certain interval $[I_1, I_2]$, the equations described in this section show that there are three possible cumulative distances after propagation; one corresponding to the normal propagation equations derived in section 4.2.1 and other two corresponding to having fixed $r' = I_1$ or $r' = I_2$, when r' "saturates" at the extrema of the interval. The following properties describe the relationship between these three distance functions.

Property 4.2.24 *If* $\mathcal{D}^{(1)}(r) > \mathcal{D}^{(2)}(r)\ \forall r$ *and* $\mathcal{D}^{(2)}(r) > \mathcal{D}^{(3)}(r)\ \forall r$ *then* $\mathcal{D}^{(1)}(r) >$

$\mathcal{D}^{(3)}(r)\ \forall r$

Proof: Assume $\exists\, I = [r_1, r_2]$ such that $\mathcal{D}^{(1)}(r) \leq \mathcal{D}^{(3)}(r)\ \forall r \in [r_1, r_2]$. Take $r^* \in I$, $\mathcal{D}^{(1)}(r^*) \leq \mathcal{D}^{(3)}(r^*)$ and $\mathcal{D}^{(1)}(r^*) > \mathcal{D}^{(2)}(r^*)$, then $\mathcal{D}^{(2)}(r^*) < \mathcal{D}^{(3)}(r^*)$; that is a contradiction. ∎

Property 4.2.25 *Given a cumulative distance $\mathcal{D}'(r')$ at certain step in the algorithm, the corresponding cumulative distance after propagation using the equations derived in section 4.2.1 is $\mathcal{D}(r)$. Assuming that the curvilinear coordinate r' is constrained to a certain interval $[I_1, I_2]$, then the corresponding cumulative distances when r' is fixed at the extrema of the interval are $\mathcal{D}^{(1)}(r)$ and $\mathcal{D}^{(2)}(r)$. Then $\mathcal{D}^{(1),(2)}(r) \geq \mathcal{D}(r)$, $\forall r$.*

Proof: Let's call $f_{1,2}(r) = \mathcal{D}(r) - \mathcal{D}^{(1),(2)}(r) = \Delta A_{1,2} r^2 + 2\Delta B_{1,2} r + \Delta C_{1,2}$. We need to show that $f_{1,2}(r) \leq 0\ \forall r$.

Considering case 1, we have the following:

$$\mathcal{D}(y) = \tfrac{A_{x'}+1-\cos^2\theta}{A_{x'}+1} y^2 + 2\tfrac{-(A_{x'}\Delta_x + B_{x'})\cos\theta}{A_{x'}+1} y + C_{x'} + \Delta_x^2 - \tfrac{(B_{x'}-\Delta_x)^2}{A_{x'}+1}$$
$$\mathcal{D}^{(1),(2)}(y) = y^2 - 2(\Delta_x - I_{1,2}) y \cos\theta + (\Delta_x^2 - I_{1,2})^2 + A_{x'} I_{1,2}^2 + 2B_{x'} I_{1,2} + C_{x'}$$

therefore,

$$\Delta A_{1,2} = \tfrac{A_{x'}+1-\cos^2\theta}{A_{x'}+1} - 1 = \tfrac{-\cos^2\theta}{A_{x'}+1}$$
$$\Delta B_{1,2} = \tfrac{-(A_{x'}\Delta_x + B_{x'})\cos\theta}{A_{x'}+1} + (\Delta_x - I_{1,2})\cos\theta = \tfrac{-(B_{x'} + (A_{x'}+1) I_{1,2} - \Delta_x)\cos\theta}{A_{x'}+1}$$
$$\Delta C_{1,2} = -\tfrac{(B_{x'} + (A_{x'}+1) I_{1,2} - \Delta_x)^2}{A_{x'}+1}$$
$$\Delta {B_{1,2}}^2 - \Delta A_{1,2} \Delta C_{1,2} = 0$$

Thus, $f_{1,2}(y)$ is a quadratic function that has its opening pointing downwards and its vertex on the horizontal axis. These two conditions imply that $f_{1,2}(y) \leq 0\ \forall y$, and, consequently, $\mathcal{D}^{(1),(2)}(y) \geq \mathcal{D}(y)$, $\forall y$.

Considering case 3, we have the following:

$$\mathcal{D}(x) = \frac{A_{x'}}{A_{x'}+1}x^2 + 2\frac{B_{x'}-A_{x'}\Delta_y\cos\theta}{A_{x'}+1}x + C_{x'} + \Delta_y^2 - \frac{(B_{x'}+\Delta_y\cos\theta)^2}{A_{x'}+1}$$
$$\mathcal{D}^{(1),(2)}(x) = x^2 - 2(I_{1,2} + \Delta_y\cos\theta)x + (A_{x'}+1)I_{1,2}^2 + 2(B_{x'}\Delta_y\cos\theta)I_{1,2} + C_{x'} + \Delta_y^2,$$

therefore,

$$\Delta A_{1,2} = \frac{A_{x'}}{A_{x'}+1} - 1 = \frac{-1}{A_{x'}+1}$$
$$\Delta B_{1,2} = \frac{B_{x'}-A_{x'}\Delta_y\cos\theta}{A_{x'}+1} + (I_{1,2} + \Delta_y\cos\theta) = \frac{B_{x'}+(A_{x'}+1)I_{1,2}+\Delta_y\cos\theta}{A_{x'}+1}$$
$$\Delta C_{1,2} = -\frac{(B_{x'}+(A_{x'}+1)I_{1,2}+\Delta_y\cos\theta)^2}{A_{x'}+1}$$
$${\Delta B_{1,2}}^2 - \Delta A_{1,2}\Delta C_{1,2} = 0$$

Thus, $f_{1,2}(x)$ is a quadratic function that has its opening pointing downwards and its vertex on the horizontal axis. These two conditions imply that $f_{1,2}(x) \leq 0\ \forall x$, and, consequently, $\mathcal{D}^{(1),(2)}(y) \geq \mathcal{D}(y),\ \forall y$. ∎

Corollary 4.2.5 *If $\mathcal{D}'^{(1)}(r') > \mathcal{D}'^{(2)}(r'),\ \forall r'$, then $\mathcal{D}^{(1)}(r) > \mathcal{D}^{(2)}(r),\ \forall r$, independently of any interval constraint in r' that alters the normal propagation of $\mathcal{D}'^{(1)}(r')$.*

Proof: The proof follows from the two previous properties and corollaries 4.2.2 and 4.2.4. ∎

4.2.7 Summary of the CDPM algorithm

We have described the CDPM algorithm in full detail in previous sections. Let us summarize the main characteristics of the algorithm.

The algorithm finds the matching between two curves at inter-sample resolution such that a distance between curves is minimized.

The distance between curves is translation-independent.

The curves are parameterized in arc-length, assuming a linear interpolation model between samples in order to calculate the inter-sample points. This linear model gives rise to a distance between curves that is quadratic in the curvilinear coordinate.

The algorithm can be expressed in a recursive fashion, using the Dynamic Programming paradigm.

The cumulate distance between curves at each iteration of the recursion is a quadratic function of the continuous curvilinear variable.

The number of cumulative distance functions needed to be stored grows exponentially with the number of iterations.

These cumulative distance functions compete in order provide the correspondence between curves that minimizes the total distance.

The number of cumulative distance functions can be reduced using two different sets of properties based on pairwise comparison between these functions:

If two cumulative functions do not intersect at a certain iteration of the algorithm, they will never intersect. The parabola that is below will be kept and the other will be discarded.

If two cumulative distance functions intersect at a certain iteration, they will always intersect. After a few iterations of the algorithm the two distance functions will converge to be almost the same parabola, only displaced vertically. Upon convergence, the parabola that is below will be kept and the other will be discarded.

Each cumulative distance function has a corresponding interval of validity that has to be propagated through the warping plane. This interval of validity makes sure that the back-propagation of the correspondence function gives rise to points located between samples.

Let us summarize the Continuous Dynamic Programming Matching algorithm using all the derived properties. We refer to parabolas generically, including also in this denomination the constant functions that arise in the first iterations. In order to backtrack the correspondence function after obtaining the minimum distance, the algorithm needs to recall the parent parabola for each parabola $\mathcal{D}_{(i,j)}(y)$ and $\mathcal{D}_{(i,j)}(x)$ and also the side of the square ($(i-1,j)$ or $(i,j-1)$) to which the parent belongs. The algorithm needs to store the linear function that maps the curvilinear coordinates before and after propagation of each parabola in each square of the grid of the warping plane. The notation $\mathcal{D}_{(i,j)}(\cdot)$ uses the curvilinear variables x and y to clarify the output side of the square of the grid to which the parabola belongs to. We use a super-index x or y to indicate the correspondence of all the following auxiliary variables with $\mathcal{D}_{(i,j)}(\cdot)$. Let us store the parent index of parabola k at square (i,j) in $\zeta^{x,y}_{(i,j)}(k)$ and the parent's side on $\psi^{x,y}_{(i,j)}(k)$. Let us store the coefficients of the linear function corresponding to parabola k at square (i,j) in $\mathcal{M}^{x,y}_{(i,j)}(k)$ and $\mathcal{H}^{x,y}_{(i,j)}(k)$. All these variables stores the information required to perform the backtracking of the optimal correspondence map after having found the minimum matching distance.

1 Initialization:

(a) Compute the two parabolas $\mathcal{D}_{(2,2)}(y)$ corresponding to the first iteration of cases 1 and 4. Calculate the corresponding values of $\mathcal{M}^{y}_{(2,2)}$ and $\mathcal{H}^{y}_{(2,2)}$, $\zeta^{y}_{(2,2)}(1) = \zeta^{y}_{(2,2)}(2) = 1$, $\psi^{y}_{(2,2)}(1) = x$, and $\psi^{y}_{(2,2)}(2) = y$.

(b) Compute the two parabolas $\mathcal{D}_{(2,2)}(x)$ corresponding to the first iteration of cases 2 and 3. Calculate the corresponding values of $\mathcal{M}^{x}_{(2,2)}$ and $\mathcal{H}^{x}_{(2,2)}$, $\zeta^{x}_{(2,2)}(1) = \zeta^{x}_{(2,2)}(2) = 1$, $\psi^{x}_{(2,2)}(1) = y$, and $\psi^{x}_{(2,2)}(2) = x$.

(c) For $i = 2$ and $2 < j \leq N_y$, such that i and j stay within the allowed grid,

i. Compute $\mathcal{D}^{1}_{(2,j)}(y)$, $\mathcal{M}^{y}_{(2,j)}(1)$, and $\mathcal{H}^{y}_{(2,j)}(1)$ using the equations for the first iteration of case 4. $\zeta^{y}_{(2,j)}(1) = 1$ and $\psi^{y}_{(2,j)}(1) = y$.

ii. Compute $\mathcal{D}^{1}_{(2,j)}(x)$, $\mathcal{M}^{x}_{(2,j)}(1)$, and $\mathcal{H}^{x}_{(2,j)}(1)$ using the equations for the first iteration of case 2. $\zeta^{x}_{(2,j)}(1) = 1$ and $\psi^{x}_{(2,j)}(1) = y$.

iii. Propagate all parabolas $\mathcal{D}^k_{(2,j-1)}(x)$, $k = 1, 2\cdots$ to $\mathcal{D}^{k+1}_{(2,j)}(y)$ using the equations for case 1. Compute $\mathcal{M}^y_{(2,j)}(k+1)$, and $\mathcal{H}^y_{(2,j)}(k+1)$ accordingly. $\zeta^y_{(2,j)}(k+1) = k$ and $\psi^y_{(2,j)}(k+1) = x$.

iv. Propagate all parabolas $\mathcal{D}^k_{(2,j-1)}(x)$, $k = 1, 2\cdots$ to $\mathcal{D}^{k+1}_{(2,j)}(x)$ using the equations for case 3. Compute $\mathcal{M}^x_{(2,j)}(k+1)$, and $\mathcal{H}^x_{(2,j)}(k+1)$ accordingly. $\zeta^x_{(2,j)}(k+1) = k$ and $\psi^x_{(2,j)}(k+1) = x$.

v. Perform pairwise comparison between all parabolas $\mathcal{D}^k_{(2,j)}(y)$ in order to keep only the ones that belong to the minimum envelope or intersect the parabolas in the minimum envelope. Check for pairwise convergence of the coefficients of the kept parabolas. Upon convergence of the coefficients, i.e., $|\Delta A| \leq \epsilon$, $|\Delta B| \leq \epsilon$ and $|\delta\Delta C| \leq \epsilon$, choose the parabola to discard by looking at $\operatorname{sign}(\frac{\Delta C}{\Delta A})$.

vi. Perform pairwise comparison between all parabolas $\mathcal{D}^k_{(2,j)}(x)$ in order to keep only the ones that belong to the minimum envelope or intersect the parabolas in the minimum envelope. Check for pairwise convergence of the coefficients of the kept parabolas. Upon convergence of the coefficients, i.e., $|\Delta A| \leq \epsilon$, $|\Delta B| \leq \epsilon$ and $|\delta\Delta C| \leq \epsilon$, choose the parabola to discard by looking at $\operatorname{sign}(\frac{\Delta C}{\Delta A})$.

(d) For $2 < i \leq N_x$ and $j = 2$, such that i and j stay within the allowed grid,

i. Compute $\mathcal{D}^1_{(i,2)}(y)$, $\mathcal{M}^y_{(i,2)}(1)$, and $\mathcal{H}^y_{(i,2)}(1)$ using the equations for the first iteration of case 1. $\zeta^y_{(i,2)}(1) = 1$ and $\psi^y_{(i,2)}(1) = x$.

ii. Compute $\mathcal{D}^1_{(i,2)}(x)$, $\mathcal{M}^x_{(i,2)}(1)$, and $\mathcal{H}^x_{(i,2)}(1)$ using the equations for the first iteration of case 3. $\zeta^x_{(i,2)}(1) = 1$ and $\psi^x_{(i,2)}(1) = x$.

iii. Propagate all parabolas $\mathcal{D}^k_{(i-1,2)}(y)$, $k = 1, 2\cdots$ to $\mathcal{D}^{k+1}_{(i,2)}(y)$ using the equations for case 4. Compute $\mathcal{M}^y_{(i,2)}(k+1)$, and $\mathcal{H}^y_{(i,2)}(k+1)$ accordingly. $\zeta^y_{(i,2)}(k+1) = k$ and $\psi^y_{(i,2)}(k+1) = y$.

iv. Propagate all parabolas $\mathcal{D}^k_{(i-1,2)}(x)$, $k = 1, 2\cdots$ to $\mathcal{D}^{k+1}_{(i,2)}(x)$ using the equations for case 2. Compute $\mathcal{M}^x_{(i,2)}(k+1)$, and $\mathcal{H}^x_{(i,2)}(k+1)$ accordingly. $\zeta^x_{(i,2)}(k+1) = k$ and $\psi^x_{(i,2)}(k+1) = y$.

v. Perform pairwise comparison between all parabolas $\mathcal{D}^k_{(i,2)}(y)$ in order to keep only the ones that belong to the minimum envelope or intersect the parabolas in the minimum envelope. Check for pairwise convergence of the coefficients of the kept parabolas. Upon convergence of the coefficients, i.e., $|\Delta A| \leq \epsilon$, $|\Delta B| \leq \epsilon$ and $|\delta \Delta C| \leq \epsilon$, choose the parabola to discard by looking at $\mathrm{sign}(\frac{\Delta C}{\Delta A})$.

vi. Perform pairwise comparison between all parabolas $\mathcal{D}^k_{(i,2)}(x)$ in order to keep only the ones that belong to the minimum envelope or intersect the parabolas in the minimum envelope. Check for pairwise convergence of the coefficients of the kept parabolas. Upon convergence of the coefficients, i.e., $|\Delta A| \leq \epsilon$, $|\Delta B| \leq \epsilon$ and $|\delta \Delta C| \leq \epsilon$, choose the parabola to discard by looking at $\mathrm{sign}(\frac{\Delta C}{\Delta A})$.

2 Recursion: for $3 \leq i \leq N_x$, $3 \leq j \leq N_y$, such that i and j stay within the allowed grid,

(a) Propagate all parabolas $\mathcal{D}^k_{(i,j-1)}(x)$, $k = 1, 2 \cdots p_x$ to $\mathcal{D}^k_{(i,j)}(y)$ using the equations for case 1. Compute $\mathcal{M}^y_{(i,j)}(k)$, and $\mathcal{H}^y_{(i,j)}(k)$ accordingly. $\zeta^y_{(i,j)}(k) = k$ and $\psi^y_{(i,j)}(k) = x$.

(b) Propagate all parabolas $\mathcal{D}^k_{(i,j-1)}(x)$, $k = 1, 2 \cdots p_x$ to $\mathcal{D}^k_{(i,j)}(x)$ using the equations for case 3. Compute $\mathcal{M}^x_{(i,j)}(k)$, and $\mathcal{H}^x_{(i,j)}(k)$ accordingly. $\zeta^x_{(i,j)}(k) = k$ and $\psi^x_{(i,j)}(k) = x$.

(c) Propagate all parabolas $\mathcal{D}^k_{(i-1,j)}(y)$, $k = 1, 2 \cdots p_y$ to $\mathcal{D}^{k+p_x}_{(i,j)}(y)$ using the equations for case 4. Compute $\mathcal{M}^y_{(i,j)}(k+p_x)$, and $\mathcal{H}^y_{(i,j)}(k+p_x)$ accordingly. $\zeta^y_{(i,j)}(k+p_x) = k$ and $\psi^y_{(i,j)}(k+p_x) = y$.

(d) Propagate all parabolas $\mathcal{D}^k_{(i-1,j)}(x)$, $k = 1, 2 \cdots p_y$ to $\mathcal{D}^{k+p_x}_{(i,j)}(x)$ using the equations for case 2. Compute $\mathcal{M}^x_{(i,j)}(k+p_x)$, and $\mathcal{H}^x_{(i,j)}(k+p_x)$ accordingly. $\zeta^x_{(i,j)}(k+p_x) = k$ and $\psi^x_{(i,j)}(k+p_x) = y$.

(e) Perform pairwise comparison between all parabolas $\mathcal{D}^k_{(i,j)}(y)$ in order to keep only the ones that belong to the minimum envelope or intersect the parabolas in the minimum envelope. Check for pairwise convergence of the coefficients of the kept parabolas. Upon convergence of the coefficients, i.e., $|\Delta A| \leq \epsilon$, $|\Delta B| \leq \epsilon$ and $|\delta \Delta C| \leq \epsilon$, choose the parabola to discard by looking at $\text{sign}(\frac{\Delta C}{\Delta A})$.

(f) Perform pairwise comparison between all parabolas $\mathcal{D}^k_{(i,j)}(x)$ in order to keep only the ones that belong to the minimum envelope or intersect the parabolas in the minimum envelope. Check for pairwise convergence of the coefficients of the kept parabolas. Upon convergence of the coefficients, i.e., $|\Delta A| \leq \epsilon$, $|\Delta B| \leq \epsilon$ and $|\delta \Delta C| \leq \epsilon$, choose the parabola to discard by looking at $\text{sign}(\frac{\Delta C}{\Delta A})$.

3 Termination:

Find the minimum of all parabolas $\mathcal{D}_{(N_x,N_y)}(y)$ and $\mathcal{D}_{(N_x,N_y)}(x)$. The minimum value is the remaining distance after matching. The corresponding curvilinear coordinate x or y that provides the minimum distance needs to be propagated back through the warping plane to obtain the correspondence function.

4 Path Backtracking:

If the minimum matching distance is provided by a parabola $\mathcal{D}_{(N_x,N_y)}(y)$, then $\sigma = y$; otherwise, $\sigma = x$. Let us call *iota* the index of the parabola that gives the minimum distance, ρ the value of x or y corresponding to the minimum distance, and $\mathcal{X}$, $\mathcal{Y}$ the corresponding matching points. The resulting correspondence function $\begin{bmatrix} t_k \\ s_k \end{bmatrix}$ and the matching points arrays $\mathcal{X}$, $\mathcal{Y}$ are ordered backwards, i.e., starting from the last matching points, so they should be reordered if desired.

$i = N_x, \quad j = N_y, \quad k = 1$

do

if $\sigma = x$,

$$\text{then} \quad \begin{bmatrix} t_k \\ s_k \end{bmatrix} = \begin{bmatrix} i - 1 + \frac{\rho}{\Delta_{x_i}} \\ j \end{bmatrix}, \ \mathcal{X}(k) = \begin{bmatrix} u_x(i-1) + \rho\frac{\Delta u_{x_i}}{\Delta_{x_i}} \\ v_x(i-1) + \rho\frac{\Delta v_{x_i}}{\Delta_{x_i}} \end{bmatrix}, \ \mathcal{Y}(k) = Y(j)$$

$$\text{else} \quad \begin{bmatrix} t_k \\ s_k \end{bmatrix} = \begin{bmatrix} i \\ j - 1 + \frac{\rho}{\Delta_{y_j}} \end{bmatrix} \ \mathcal{X}(k) = X(i), \ \mathcal{Y}(k) = \begin{bmatrix} u_y(j-1) + \rho\frac{\Delta u_{y_j}}{\Delta_{y_j}} \\ v_y(j-1) + \rho\frac{\Delta v_{y_j}}{\Delta_{y_j}} \end{bmatrix}$$

$\rho = \mathcal{M}^{\sigma}_{(i,j)}(\iota)\rho + \mathcal{H}^{\sigma}_{(i,j)}(\iota)$

$\iota = \zeta^{\sigma}_{(i,j)}(\iota)$

$\sigma = \psi^{\sigma}_{(i,j)}(\iota)$

if $\sigma = x$, then $j = j - 1$, else $i = i - 1$

until $i = 1$ or $j = 1$

4.3 Experiments

We evaluate the performance of the continuous dynamic programming matching algorithm (CDPM) in comparison with DPM with and without oversampling. We use synthetic data as well as real signatures to perform the comparison.

4.3.1 Experiment 1: Comparison of CDPM with DPM for synthetic data

Figure 4.16 shows the matching results for DPM, DPM with oversampling, and CDPM on synthetic data. The first plot corresponds to DPM without oversampling, the second plot corresponds to DPM with oversampling of the more coarsely sampled curve with a factor such that both curves have a similar number of samples, the third plot corresponds to CDPM, and the fourth plot shows the warping functions corresponding to each of the algorithms. We observe that the warping path corresponding to the continuous case is smoother than the warping path obtained with DPM. The case of DPM with oversampling provides a reasonable result in terms of matching, even though the correspondence map is still not invertible. This problem appears because the re-sampling of the curve is uniform and independent of the position of the samples of the other curve; therefore, many new samples may be allocated in a region in which the other curve has few samples. The continuous algorithm instead, by its very nature, adapts to the number of samples in each of the curves, providing in this way a better matching between the two curves. In figure 4.17, we zoom into two different portions of the matching between the curves in order to compare the three different correspondence methods as well as the three different warping paths.

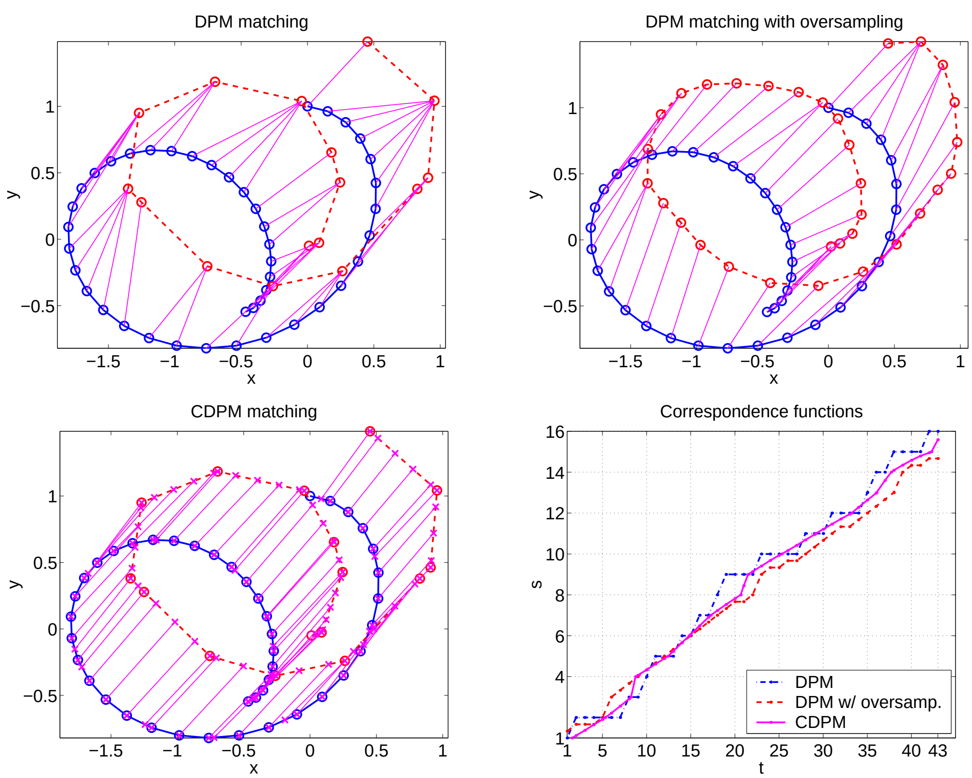


Figure 4.16: The first 3 plots shows the results of DPM, DPM with oversampling and CDPM applied to a synthetic pair of curves. The last plot displays the warping plane and the corresponding warping paths for each of the algorithms.

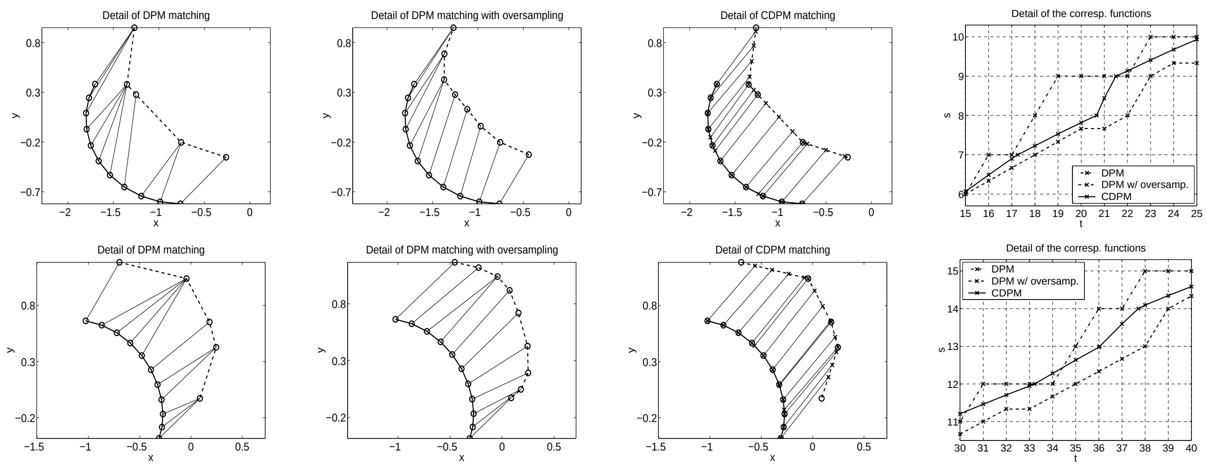


Figure 4.17: Detail of the matching between the curves of figure 4.16. In each row, we show two different portions of the curves and the corresponding matching obtained using the three methods under comparison. The last plot of each row displays the corresponding portion of the warping path.

4.3.2 Experiment 2: Comparison of CDPM with DPM for signatures

Figure 4.18 shows the correspondence between sample points obtained using DPM and CDPM for two signatures in the data set. The correspondence map provided by CDPM is more dense than the one obtained with DPM and is also invertible. The correspondence functions are displayed in the last plot of the figure, where we have also added the correspondence function obtained for DPM with the two signatures oversampled by a factor of 5. We observe that the three warping functions are almost indiscernible in this plot. Figure 4.19 shows in detail the matching of two portions of the signatures as well as the corresponding warping paths. We note that the warping paths corresponding to DPM with and without oversampling are almost the same. The warping path corresponding to CDPM is almost an interpolating function of the warping path for DPM.

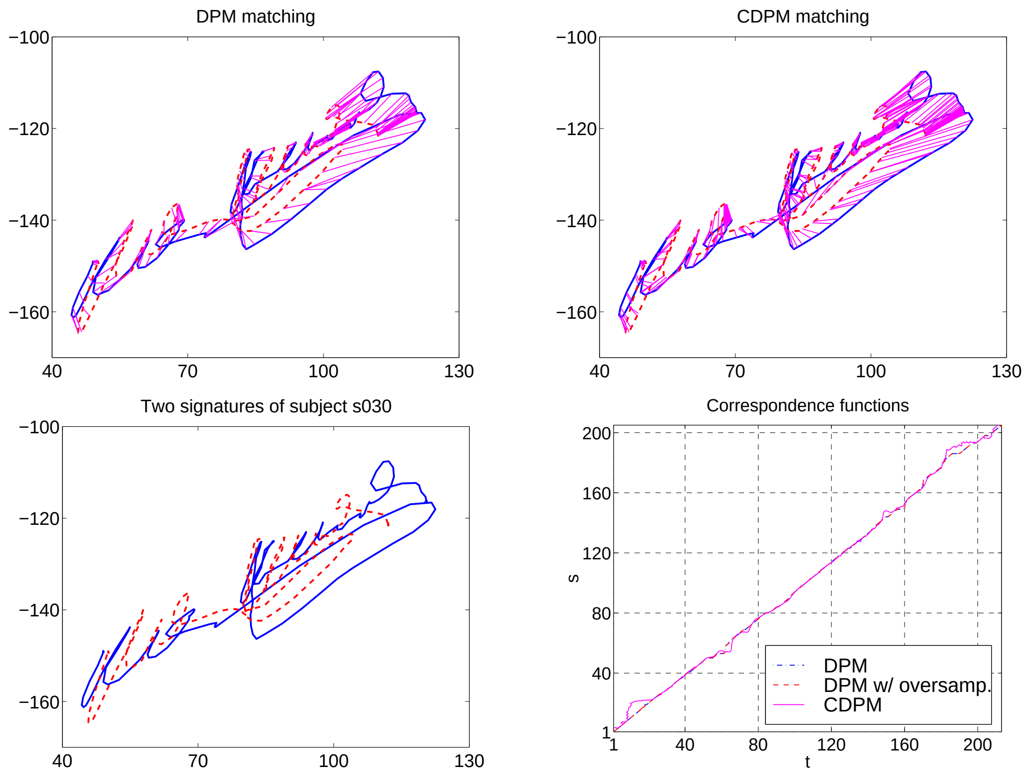


Figure 4.18: Signature matching using DPM and CDPM. The first two plots show the correspondence provided by the matching. The third plot shows the signatures and the fourth plot shows the corresponding warping paths and the warping path for DPM applied on an oversampled version of the signatures.

4.3.3 Experiment 3: Experimental evaluation of the computational cost of CDPM and comparison of the computational time of CDPM and DPM

In this experiment we evaluated the computational performance of CDPM in comparison with DPM with and without oversampling. We took two example signatures from each of the subjects in the databases and we matched them using each of the three methods. The matching is computed for four different values of maximum deviation from linear warping: 10, 20, 30, and 40 samples. For the oversampling case, we re-sampled the signatures by a factor of five using linear interpolation. We measured the time required for each algorithm to run as well as the maximum storage required for CDPM.

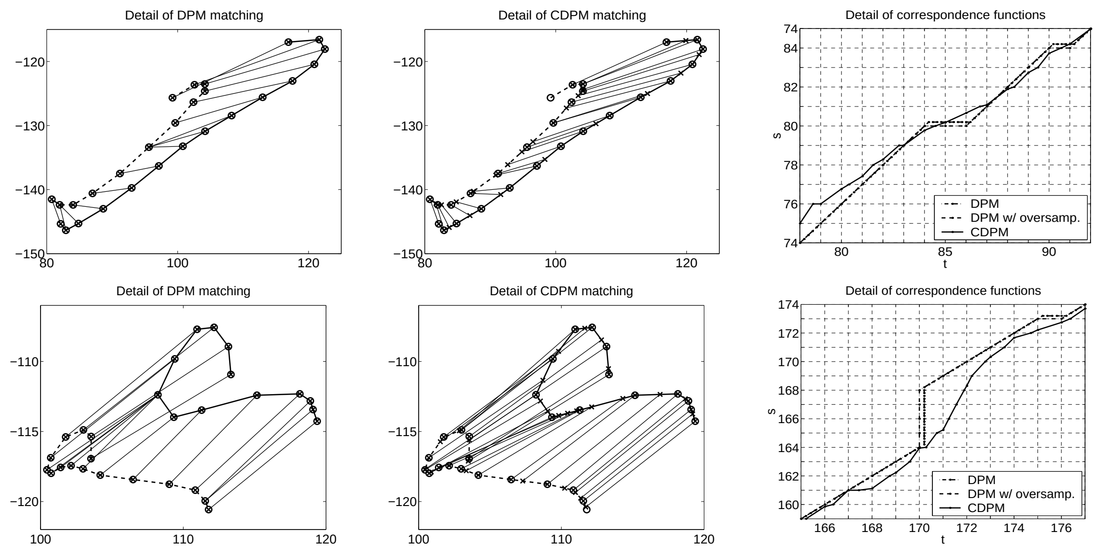


Figure 4.19: Detail of the matching between the two signatures shown in figure 4.18. In each row, we show two different portions of the curves and the corresponding matching obtained using DPM and CDPM. The last plot of each row displays the corresponding portion of the warping path.

Figure 4.20 shows the computation time required for each algorithm as a function of the length of one of the sequences under comparison. Each plot corresponds to a different warping plane constraint. We see three different sets of points in each figure. The lower set corresponds to DPM, the middle one corresponds to DPM with oversampling and the upper one corresponds to CDPM. We note that CDPM is three orders of magnitude slower than DPM. However, if the signatures are oversampled, the computation time for DPM increases. The middle set of points in each plot correspond to DPM after oversampling the signatures by five. We found that the computation times of CDPM and DPM with oversampling are similar if the signatures are oversampled by a factor of 25 approximately. We observe that the computation time of the three algorithms follows a roughly linear relationship with the length of the sequences. The lines in each of the plots show a linear fitting of the data in semilogarithmic space and the corresponding functional dependence is presented in the legend. All three lines have a very similar slope, for each of the warping plane constraints. Therefore, all three algorithms have a similar growth of the computation time with the length of the sequence, showing in this form that the properties derived for CDPM effectively limit the combinatorial complexity of this algorithm (otherwise, the computation time would have grown with slope much bigger than the slope of DPM). The difference between the computation times of DPM and CDPM is shown in the expression for the fitting functions by the multiplicative factor. The big factor for CDPM represents the overhead in computations required to carry around, propagate and prune a whole set of quadratic functions, that is not present in DPM. The fitting functions are such that they can be approximated by a low order polynomial (linear or quadratic would be enough). In fact, the computation time varies linearly with the signature length for the case of unconstrained DPM, so it is reasonable to get a fitting function that can be approximated by a low order polynomial.

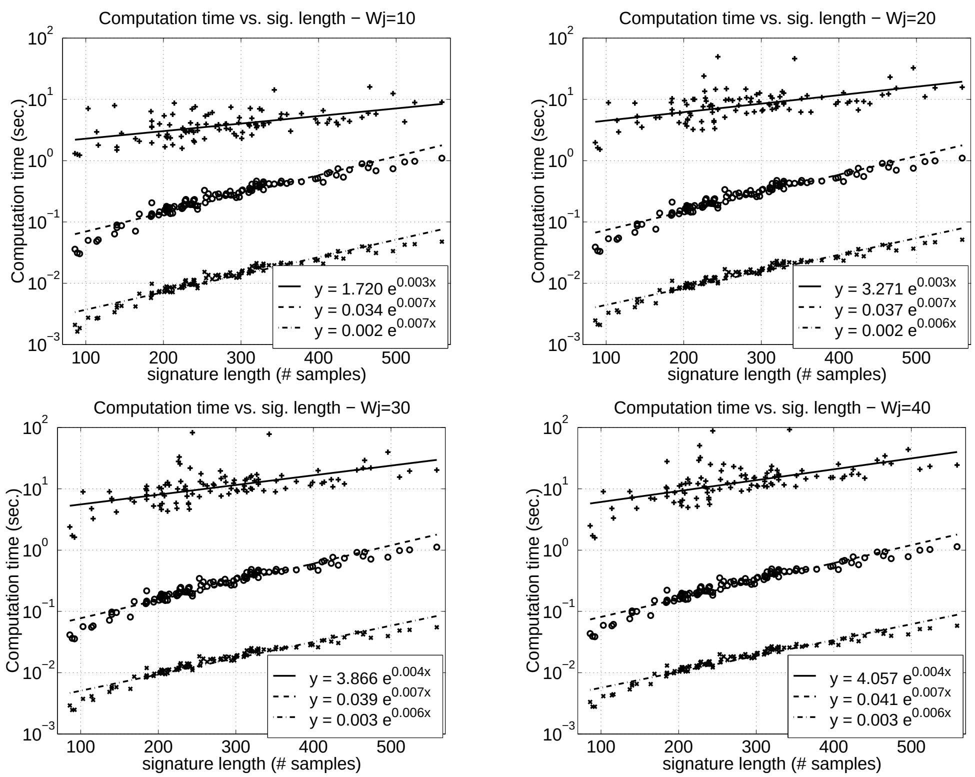


Figure 4.20: Plots of the computational time required for each algorithm as a function of the length of one of the signatures under comparison, for different warping plane constraints. The computation time for DPM is displayed with 'x' (lower curve), the time for DPM with oversampling is plotted with 'o' (middle curve) and the time for CDPM is shown with '+' (upper curve). The lines represent a linear fit in semilogarithmic space of the data.

Figure 4.21 shows the relationship between the maximum storage required for CDPM and the computation time required by the algorithm for the four different warping plane constraints. The solid line represents a linear fit of the data in logarithmic space. We note that the corresponding fitting functions are power laws with an exponent that is very close to one, i.e., the functions are almost linear functions.

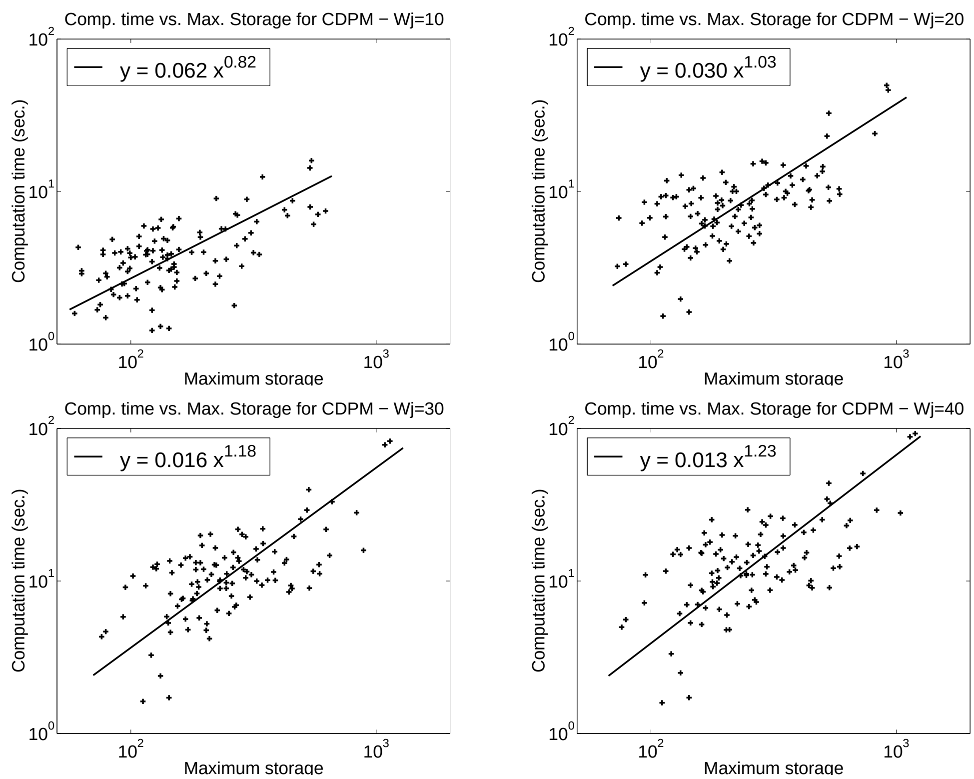


Figure 4.21: Plots of the maximum storage required for CDPM as a function of the computation time used by the algorithm, for the four different warping plane constraints. The solid line represents a linear fitting of the data in logarithmic space. We note that the corresponding fitting functions are power laws with an exponent that is very close to one, i.e., the functions are almost linear functions.

4.3.4 Experiment 4: Application to signature verification

This experiment was performed on the two databases of signatures described in chapter 3. In figure 4.22 we show several examples of signatures collected for our database, their corresponding training reference obtained with DPM and CDPM and one of the forgeries in the database. We observe that the prototype obtained with DPM is much noisier than the one obtained with CDPM, due to the fact that DPM computes the prototype only with the given discrete samples while CDPM calculates the reference signature with inter-sample points.

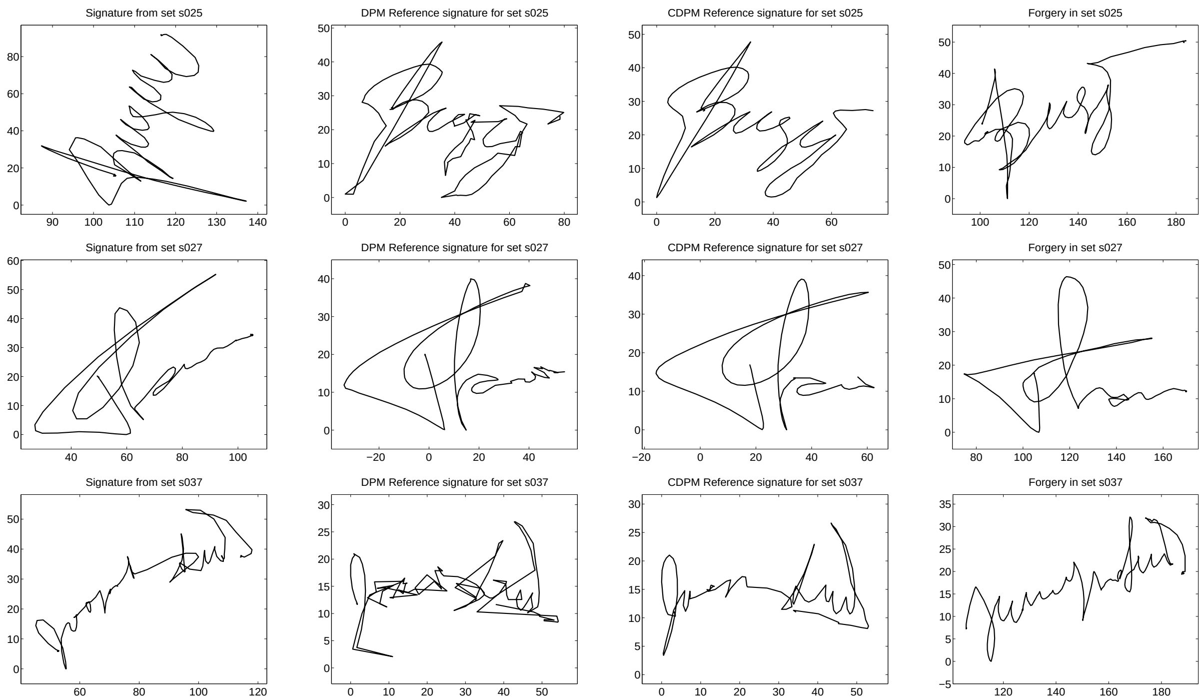


Figure 4.22: Several examples of signatures in our database and corresponding reference functions obtained with DPM and CDPM. In the first column we display signatures captured with the visual tracker, in the second and third columns, we show the corresponding reference signatures of the training set obtained with DPM and CDPM, and in the fourth column we display a forgery provided by the subjects.

Figure 4.23 shows the effect of sub-sampling some signatures in our database. The first column displays the original signatures, the second and third column presents the signatures sub-sampled by factors of 2 and 4 respectively. Given that the sampling rate is 60 Hz, the plots of the second and third column of the figure correspond to sampling rates of 30 Hz and 15 Hz respectively. We note that the differences between the signatures sampled at 60 Hz and 30 Hz are barely noticeable. However, there is a very noticeable distortion introduced by sampling at 15 Hz since the sampling rate is already below the Nyquist frequency for handwriting mentioned in chapter 2.

Figure 4.24 shows the effect of adding noise to each sample point of some signatures

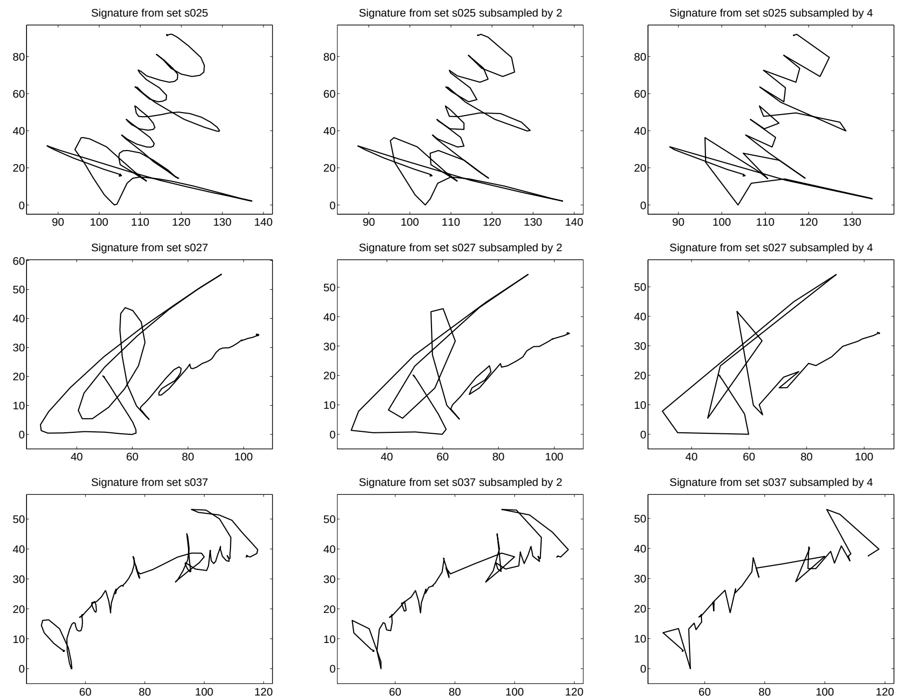


Figure 4.23: Effect of sub-sampling the signatures. The first column shows several examples of signatures in our database. The second and third column display the corresponding signatures sub-sampled by a factor of 2 and 4.

from our database. The noise is zero mean Gaussian with a standard deviation equal to five times the spatial resolution of the visual tracker calculated in chapter 2, i.e., with a standard deviation roughly equal to a quarter of a pixel. The first column of the figure displays the original signature and the second column shows the signatures after adding noise. Given the scale of the plot and the standard deviation of the noise, we barely notice any change in the plot of the signature. A blow up of a portion of the signatures with and without noise is presented in the third and fourth row of the figure, showing that there is a visible distortion introduced by the addition of noise.

The following figures show the error trade-off curves for CDPM and DPM, calculated using our databases of signatures. The performance of the DPM and CDPM algorithms are compared under different conditions of sampling rate and noise level. We only plot a section of the curve that is most informative.

Figure 4.25 presents the performance of DPM and CDPM using the distance after matching as the classification parameter. Figure 4.26 shows the performance using the

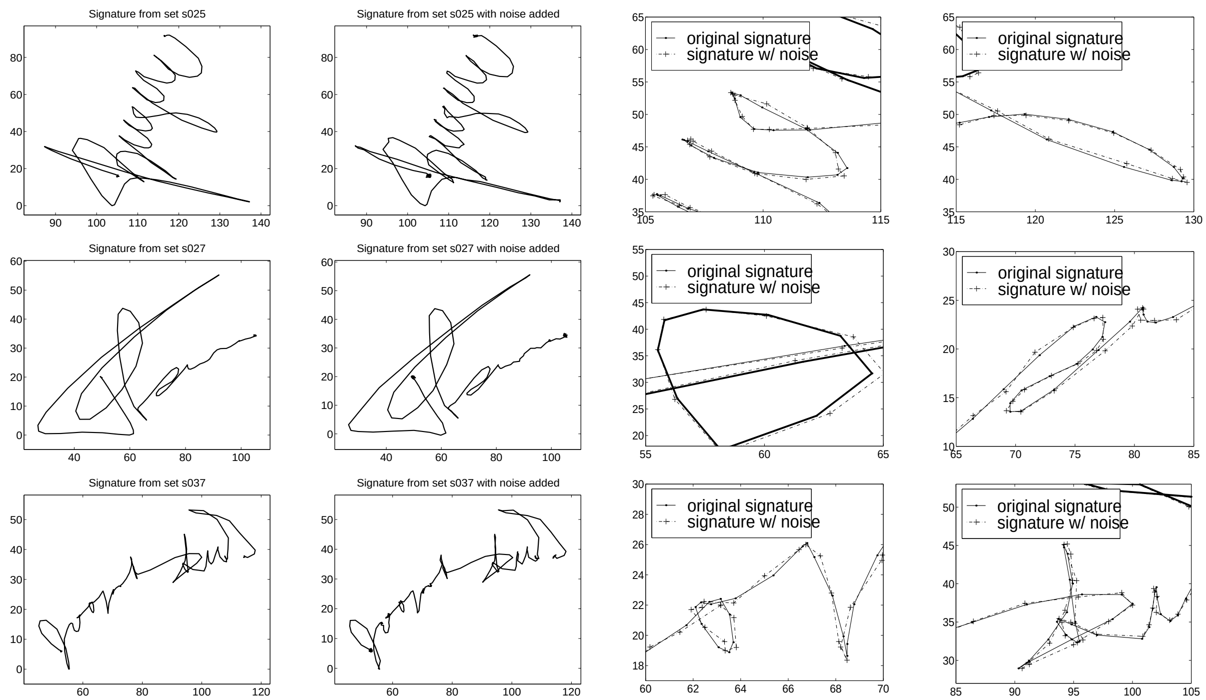


Figure 4.24: Effect of adding noise to the signatures. The first column shows several examples of signatures in our database. The second column displays the corresponding signatures after having added a zero mean Gaussian noise with a standard deviation equal to five times the spatial resolution of the visual tracker presented in chapter 2. The third and fourth columns show a detail of portions of the signatures with and without noise.

harmonic mean of all the distance measures described in chapter 3 for classification.

Figures 4.27 and 4.28 present the performance of DPM and CDPM after subsampling the signatures by a factor of two, i.e., the resulting sampling rate is 30 Hz.

Figures 4.29 and 4.30 present the performance of DPM and CDPM after subsampling the signatures by a factor of four, i.e., the resulting sampling rate is 15 Hz. This sampling rate is already below the Nyquist frequency for handwriting mentioned in chapter 2, so the performance decreases by a big factor since the signatures are highly distorted.

From figures 4.25, 4.27, and 4.29, we observe that CDPM outperforms DPM using the distance after alignment as the classification parameter. However, by adding the distances proposed in chapter 3 in order to improve the performance, figures 4.26, 4.28,

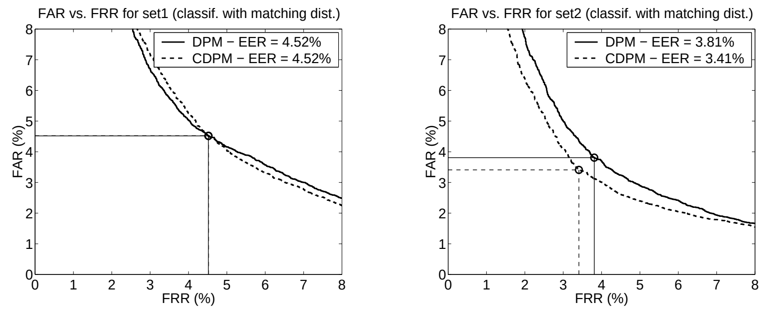


Figure 4.25: Error trade-off curves for CDPM and DPM evaluated on our databases of signatures. The distance after matching is used for classification.

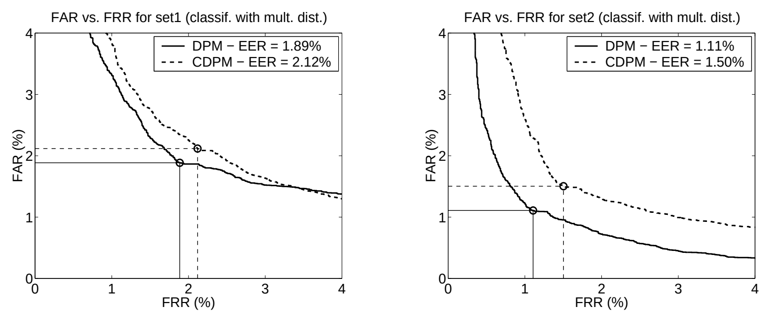


Figure 4.26: Error trade-off curves for CDPM and DPM evaluated on our databases of signatures. The harmonic mean of all the distances proposed in chapter 3 are used for classification.

and 4.30 show that DPM has a better performance than CDPM.

Figures 4.31 and 4.32 present the performance of DPM and CDPM after adding noise to the signatures. The noise is zero mean Gaussian with a standard deviation equal to five times the spatial resolution of the visual tracker presented in chapter 2. We observe that the performance of both algorithms decreases and that CDPM no longer performs better than DPM when using the distance after alignment as the classification parameter.

Figure 4.33 presents the comparison of the equal error rates for DPM and CDPM for different sub-sampling factors. We note that CDPM performs better than DPM when using the residual distance between the curves after matching as the classification parameter. However, DPM outperforms CDPM when the harmonic mean of the similarity measures defined on chapter 3 is used as the classification parameter.

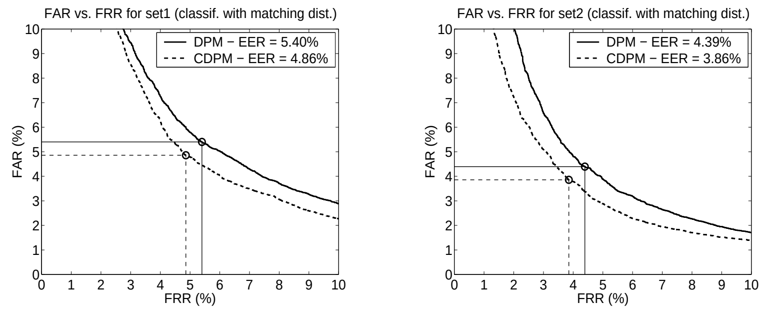


Figure 4.27: Error trade-off curves for CDPM and DPM evaluated on our databases of signatures after sub-sampling the signatures by a factor of two. The distance after matching is used for classification.

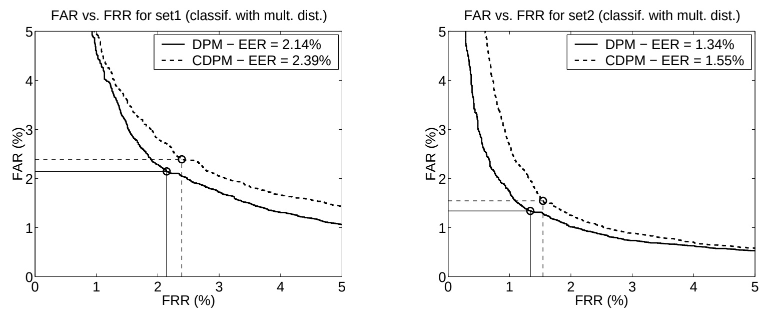


Figure 4.28: Error trade-off curves for CDPM and DPM evaluated on our databases of signatures after sub-sampling the signatures by a factor of two. The harmonic mean of all the distances proposed in chapter 3 are used for classification.

Figure 4.34 shows the comparison of the equal error rates for DPM and CDPM for different noise levels. We note that the effect of the addition of noise on the performance of DPM and CDPM is unclear since the values of the equal error rates increase, decrease or are the same, depending on the database used for testing and depending on the similarity measures used for classification.

4.3.5 Discussion

The previous section presented the experimental evaluation of the performance of the CDPM algorithm. We have shown that the algorithm is computationally tractable when applying the different properties that we have derived previously. The computational time required by CDPM is three orders of magnitude greater than the time

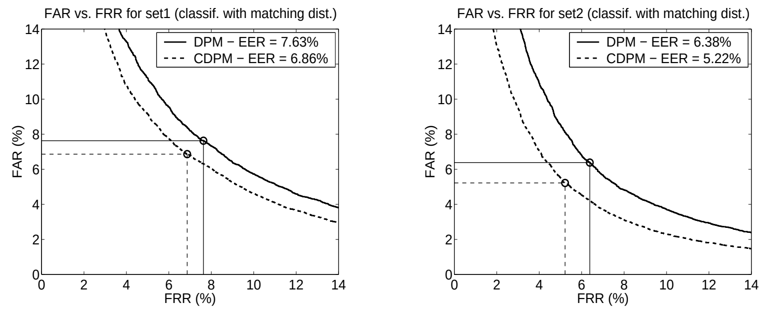


Figure 4.29: Error trade-off curves for CDPM and DPM evaluated on our databases of signatures after sub-sampling the signatures by a factor of four. The distance after matching is used for classification.

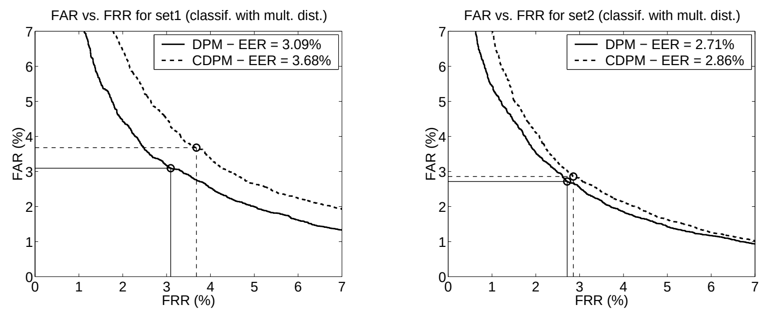


Figure 4.30: Error trade-off curves for CDPM and DPM evaluated on our databases of signatures after sub-sampling the signatures by a factor of four. The harmonic mean of all the distances proposed in chapter 3 are used for classification.

required by DPM. However, the growth of the computational time with signature length is similar for both CDPM and DPM, meaning that the mentioned properties effectively limit the computational complexity of CDPM. The comparison of the performance of DPM and CDPM for signature verification shows that CDPM is better if the residual distance after matching is used as the classification parameter, but DPM is still superior to CDPM when several similarity measures are used for classification. Looking at the reference signatures obtained with DPM and CDPM, there is a clear improvement in terms of smoothness provided by CDPM, although this improvement in smoothness is not directly translated into better performance. It is possible to conclude that the error rates achieved with DPM are so small that it would be difficult to improve them with CDPM, and in fact this latter results in a small increase. It is

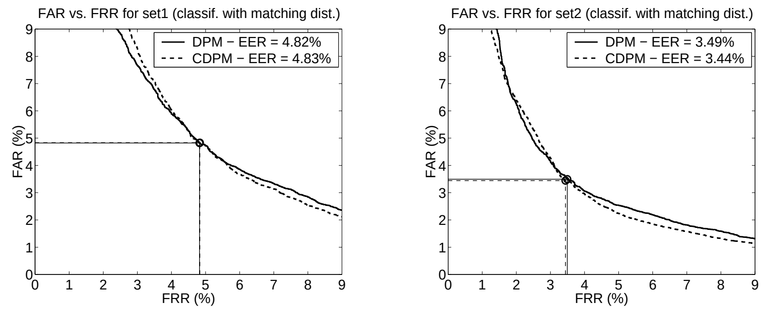


Figure 4.31: Error trade-off curves for CDPM and DPM evaluated on our databases of signatures after adding to the signatures zero mean Gaussian noise with a standard deviation equal to five times the spatial resolution of the visual tracker presented in chapter 2. The distance after matching is used for classification.

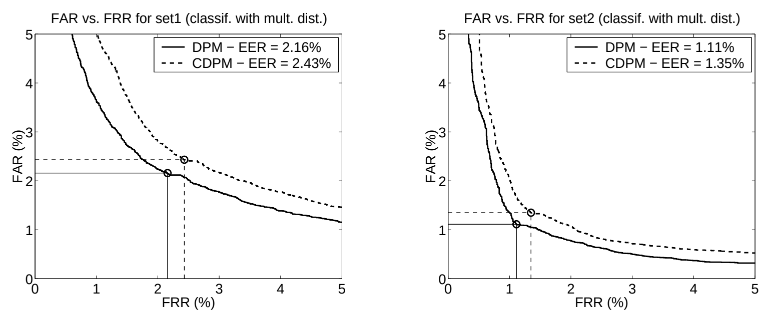


Figure 4.32: Error trade-off curves for CDPM and DPM evaluated on our databases of signatures after adding to the signatures zero mean Gaussian noise with a standard deviation equal to five times the spatial resolution of the visual tracker presented in chapter 2. The harmonic mean of all the distances proposed in chapter 3 is used for classification.

also possible to conclude that the similarity measures used in our experiments do not take full advantage of the matching at inter-sample resolution provided by CDPM, and therefore, this is the reason for not obtaining a better performance with CDPM.

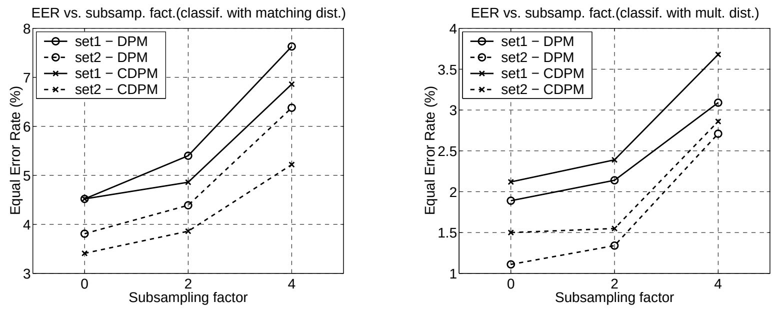


Figure 4.33: Comparison of the equal error rates for DPM and CDPM for different sub-sampling factors. The first column shows the error rates obtained using the distance after matching as the classification parameter and the second column displays the error rates obtained using the harmonic mean of several similarity measures as the classification parameter.

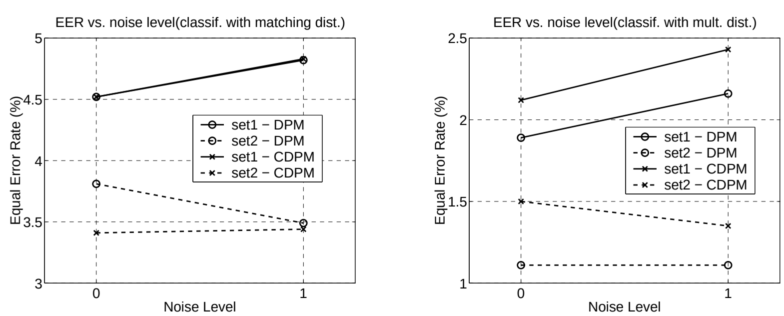


Figure 4.34: Comparison of the equal error rates for DPM and CDPM with and without noise added to the signatures.

Chapter 5 Conclusion and Future Work

This thesis has presented a novel human-computer interface for handwriting. A camera focuses on the user's hand while the user writes with a normal pen on a piece of paper. The handwriting trajectory is successfully recovered from its spatio-temporal representation given by the sequence of images. This trajectory is decomposed into handwritten strokes and pen movements between two strokes. The detection of the points in which the pen is traveling above the paper and not writing is obtained by using local measurements of the brightness of the image at the location in which the writing end of the pen is detected.

The interface is based on consumer hardware and computer vision techniques, and it has been shown to work in real-time at 60 Hz. The pen tip template is automatically acquired once the user places the pen within a predefined region of the image. The position of the pen tip is obtained by correlation with a template of the pen tip. A recursive estimation scheme is used to predict the next position of the pen tip, allowing the size of the search neighborhood to be reduced. Each point of the recovered trajectory is classified as "pen down" or "pen up" based on a local comparison of the brightness value at the ballpoint with the surrounding background. This measure is converted into a probability value. The acquired trajectory is divided into strokes and a measure of the confidence of each stroke being a pen down is provided by integrating the information of the point-wise decisions.

Several parts of the interface are open to improvement. For example, enabling the pen tip to be automatically detected and tracked from the time it enters the field of view of the camera would make the interface simpler to use and more user-friendly. The recursive estimation scheme could be improved by including a more accurate model of the dynamics of handwriting generation. The pen up/down detection could be enhanced by adding a few other measurements, such as the local orientation of the ink at position of the ballpoint, the correlation of this orientation with the local direction of the pen tip's trajectory, etc. These various measurements can be naturally included in the system by increasing the dimensionality of the HMM's observation. Finally, the error rates of pen up and pen down classification could be decreased by performing a better segmentation of the acquired trajectory.

From the experiments, we note that the user needs some minimal training in order to get accustomed to and master the interface. In particular, the visual acquisition system occasionally loses track of the pen tip in the presence of extremely fast strokes when acquiring signatures since they are usually written at a faster pace than normal handwriting. The use of more powerful machines or the implementation of our tracking algorithm in hardware using FPGA's would eliminate this problem since bigger search regions could be explored, allowing users to sign at normal speed.

This visual interface for handwriting is the initial step in order to devise a full pen-based computing system in which not only text but also mouse-like commands would be provided with a pen.

The camera-based interface is able to capture signatures to be used for personal identification. The system does not require any special hardware, unlike fingerprint verification, iris or retina scanning systems. It is comparable to face recognition systems in terms of hardware since it uses a camera for tracking the signature. In this thesis, the performance has been evaluated for a signature verification system that acquires the signatures with the visual interface for handwriting capture. The comparison between signatures is performed by using dynamic programming matching.

The experimental results show that the parameterization of the signatures is quite critical for achieving good verification performance. The best results are obtained by parameterizing the signatures with affine arc-length, using the harmonic mean of several similarity measures as the classification parameter. It can be inferred that shape similarity and causality of the signature's generation are more important than matching the dynamics of signing. This dynamics is not stable enough to be used for signature verification since the subject is trying to reproduce a shape rather than a temporal pattern. However, the causality of the signature, i.e., the order in which parts are produced, is still valuable and is used in the DPM paradigm. This causality is the added information that our on-line system is using to outperform systems that do comparison from still pictures of signatures.

The use of duplicate examples has been shown to provide a better estimate of the generalization error, given that the algorithm has to be invariant with respect to a certain class of transformations. In our experiments, we used only time origin shifting and small scaling in x and y directions as the transformation class. A full affine transformation could be used to generate duplicate examples provided that a reasonable range of the parameters of this transformation could be estimated from the data.

The very good performance of the system when tested with random forgeries indicates that the algorithm is able to discriminate whether a signature belongs to a certain class (or, in other words, to a certain subject) and provides grounds to conclude that the algorithm could be used for signature recognition.

The signature verification algorithm could be made more robust by adding more global descriptors of the signatures that would allow the system to discard coarse forgeries. One problem that is unsolved with the present scheme is dealing with dramatic changes in scale and it is one of the areas of further research, as well as the development of a better similarity measure.

Comparing signature verification with a physiological biometric technique for personal identification, such as fingerprint verification, we could observe that a forger with enough information about the true signature and enough training could deceive the algorithm. This weakness is inherent to all behavioral biometrics and may or may not be important depending on the particular application.

Signature verification could be employed to replace the use of computer passwords. In this case, the daily use of the system could make people sign more consistently and could provide them with a figure that quantifies the variability of the signature. It would also be possible to have a system in which the user signs with an ink-less pen, leaving no trace of the signature in order to block possible forgers from knowing it. This system would overcome the mentioned weakness of the method, but it would make the user feel a bit awkward since there would be absolutely no visual feedback when signing.

Dynamic programming matching (DPM), the leading conventional method for signature matching, is only able to establish *sample-to-sample* correspondence between signatures. This thesis has presented a novel algorithm for establishing *sample-to-inter-sample* correspondence between signatures. This algorithm is the continuous generalization of DPM and belongs to the general class of dynamic programming algorithms. The properties of the continuous dynamic programming algorithm (CDPM) are described in full detail, in particular those that allow one to limit the spatial complexity of the algorithm. The experimental results show that by using these properties, the growth of the computational time with signature length is similar for both CDPM and DPM, meaning that the mentioned properties effectively limit the computational complexity of CDPM. However, whether there is a theoretical bound on the spatial complexity is still a subject of research.

The performances of CDPM and DPM are compared on both synthetic and real data. DPM was used with and without oversampling of the signatures in order to provide better matching spatial resolution. In terms of computational time required to perform the matching, CDPM is three orders of magnitude slower than plain DPM, being equivalent to DPM with an oversampling factor of 25 (approximately).

The performances of CDPM and DPM are compared on signature verification, showing that both algorithms achieve similar results, although DPM outperforms CDPM. The reference signature extracted from the training set is shown to be much smoother when using CDPM to compute it, so, in principle, the performance of CDPM might be improved by using a better set of similarity measures for classification, that take advantage of the matching at subsample resolution provided by the algorithm. The development of a better set of similarity measures as well as the extraction of a better reference signature from the training set are all subjects for further work.

Bibliography

[1] Y.S. Abu-Mostafa. Hints. *Neural Computation*, 7:639–671, 1995.

[2] B.D. Anderson and J.B. Moore. *Optimal Filtering.* Prentice Hall, Inc., 1979.

[3] R. Bellman. *Dynamic Programming.* Princeton Univ. Press, 1957.

[4] E. Di Bernardo, L. Goncalves, and P. Perona. Monocular tracking of the human arm: Real-time implementation and experiments. In *Proc. 13^{th} Int. Conf. Pattern Recognition*, pages 622–626, Wien, August, 1996.

[5] E. Di Bernardo, L. Goncalves, and P. Perona. *Computer Vision for Human-Machine Interaction. R. Cipolla and A. Pentland eds.*, chapter "Monocular Tracking of the Human Arm in 3D", pages 155–169. Cambridge University Press, 1998.

[6] C. Bregler, H. Hild, S. Manke, and A. Waibel. Improving connected letter recognition by lipreading. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1993.

[7] A.M. Bruckstein, R. Holt, A. Netravali, and T. Richardson. Invariant signatures for planar shape recognition under partial occlusion. *CVGIP: Image Understanding*, 58(1):49–65, 1993.

[8] R.S. Bucy. Non-linear filtering theory. *IEEE Transactions on Automatic Control*, 1965.

[9] M. Burl, T. Leung, and P. Perona. Face localization via shape statistics. In *Proc. Intl. Workshop on automatic face and gesture recognition*, pages 154–159, 1995.

[10] M.C. Burl and P. Perona. Recognition of planar object classes. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, San Francisco, 1996.

[11] M.C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *Proc. 5^{th} Europ. Conf. Comput. Vision, H. Burkhardt and B. Neumann (Ed.), LNCS-Series Vol. 1407-1408, Springer-Verlag*, June 1998.

[12] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[13] R. Cipolla and N. Hollinghurst. Human-robot interface by pointing with uncalibrated stereo vision. *Image and Vision Computing*, 14(3):171–178, 1996.

[14] R. Cipolla, N. Hollinghurst, A. Gee, and R. Dowland. Computer vision in interactive robotics. *Assembly Automation*, 16(1):18–24, 1996.

[15] A. Colmenarez, B. Frey, and T.S. Huang. A probabilistic framework for embedded face and facial expression recognition. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, pages 592–597, 1999.

[16] C. Colombo and A. Del Bimbo. Real-time head tracking from the deformation of eye contours using a piecewise affine camera. *Pattern Recognition Letters*, 20(7):721–730, 1999.

[17] J.L. Crowley. Vision for man-machine interaction. *Robotics and Autonomous Systems*, 19(3-4):347–358, 1997.

[18] J.L. Crowley, F. Bernard, and J. Coutaz. Finger tracking as an input device for augmented reality. *International Workshop on Face and Gesture Recognition*, pages 195–200, 1995.

[19] J.G. Daugman. High confidence visual recognition of persons by a test of a statistical independence. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1148–1161, 1993.

[20] S.E. Dreyfus. *Dynamic Programming and the Calculus of Variations.* Academic Press, 1965.

[21] I.L. Dryden and K.V. Mardia. *Statistical shape analysis.* John Wiley & Sons, Ltd., 1998.

[22] I.A. Essa and A.P. Pentland. Facial expression recognition using a dynamic model and motion energy. In *Proc. 5^{th} Int. Conf. Computer Vision*, pages 360–367, Boston, June, 1995.

[23] S. Elrod et al. Liveboard: a large interactive display supporting group meetings, presentations and remote collaboration. In *CHI '92*, pages 599–607, 1992.

[24] M.C. Fairhurst. Signature verification revisited: promoting practical exploitation of biometric technology. *Electronics and Communication Engineering Journal*, pages 273–280, 1997.

[25] K.E. Finn and A.A. Montgomery. Automatic optically-based recognition of speech. *Pattern Recognition Letters*, 8(3):159–164, 1988.

[26] G.D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[27] B. Frohlich, G. Grunst, W. Kruger, and G. Wesche. The responsive workbench - a virtual working environment for physicians. *Computers in Biology and Medicine*, 25(2):301–308, 1995.

[28] K. Fukunaga. *Statistical Pattern Recognition.* Academic Press, 1990.

[29] A. Gee and R. Cipolla. Determining the gaze of faces in images. *Image and Vision Computing*, 12(10):639–647, 1994.

[30] A. Gelb. *Applied Optimal Estimation.* The MIT Press, 1974.

[31] L. Goncalves, E. Di Bernardo, and P. Perona. Reach out and touch space (motion learning). In *Proc. of the Third International Conference on Automatic Face and Gesture Recognition*, pages 234–239, Nara, Japan, April 14-16, 1998.

[32] L. Goncalves, E. Di Bernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3d. In *Proc. 5^th^ Int. Conf. Computer Vision*, pages 764–770, Boston, June, 1995.

[33] I. Haritaoglu, D. Harwood, and L. Davis. Who, when, where, what: A real time system for detecting and tracking people. In *Proceedings of the Third Face and Gesture Recognition Conference*, pages 222–227, 1998.

[34] T. Hastie, E. Kishon, M. Clark, and J. Fan. A model for signature verification. In *Proc. IEEE Conf. on Systems, Man and Cybernetics*, pages 191–196, 1991.

[35] K. Huang and H. Yan. On-line signature verification based on dynamic segmentation and global and local matching. *Optical Engineering*, 34(12):3480–3487, 1995.

[36] A.K. Jain, L. Hong, S. Pankanti, and E. Bolle. An identity-authentication system using fingerprints. *Proceedings of the IEEE*, 85(9):1365–1388, 1997.

[37] A.H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, 1970.

[38] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. of the ASME-Journal of basic engineering.*, 35–45, 1960.

[39] F. Lacquaniti, C. Terzuolo, and P. Viviani. The law relating the kinematic and figural aspects of drawing movements. *Acta Psychologica*, 54:115–130, 1983.

[40] F. Leclerc and R. Plamondon. Automatic signature verification. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(3):643–660, 1994.

[41] G. Lorette and R. Plamondon. Dynamic approaches to handwritten signature verification. *Computer Processing of Handwriting*, pages 21–47, 1990.

[42] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.

[43] J. Luettin and N.A. Thacker. Speechreading using probabilistic models. *Computer Vision and Image Understanding*, 65(2):163–178, 1997.

[44] P. Maes, T. Darrell, B. Blumberg, and A. Pentland. The alive system: Wireless, full-body interaction with autonomous agents. *Multimedia Systems*, 5(2):105–112, 1997.

[45] R. Martens and L. Claesen. On-line signature verification by dynamic time-warping. In *Proc. 13th Int. Conf. Pattern Recognition*, pages 38–42, 1996.

[46] K. Mase. Recognition of facial expression from optical flow. *IEICE Transactions*, 74(10):3474–3483, 1991.

[47] K. Mase and A. Pentland. Automatic lipreading by computer. *Trans. Inst. Elec. Info. and Comm. Eng.*, 73(6):796–803, 1990.

[48] M.E. Munich and P. Perona. Visual-based ID verification by signature tracking. In *Proc. 2nd Int. Conf. Audio- and Video- Based Person Authentication*, 1999.

[49] M.E. Munich and P. Perona. Visual signature verification using affine arc-length. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, pages 180–186, 1999.

[50] V.S. Nalwa. Automatic on-line signature verification. *Proceedings of the IEEE*, 85(2):215–239, 1997.

[51] W. Nelson and E. Kishon. Use of dynamic features for signature verification. In *Proc. IEEE Conf. on Systems, Man and Cybernetics*, pages 201–205, 1991.

[52] W. Nelson, W. Turin, and T. Hastie. Statistical methods for on-line signature verification. *Inter. Jour. of Pattern Recognition and Artificial Intelligence*, 8(3):749–770, 1994.

[53] P. Nesi and A. Del Bimbo. A vision-based 3-d mouse. *International Journal of Human-computer Studies*, 44(1):73–91, 1996.

[54] M. Parizeau and R. Plamondon. A comparative analysis of regional correlation, dynamical time warping and skeletal tree matching for signature verification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(7):710–717, 1990.

[55] V.I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for human- computer interaction: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[56] R. W. Picard. *Affective Computing*. MIT Press, 1997.

[57] R. Plamondon and B. Clément. Dependence of peripheral and central parameters describing handwriting generation on movement direction. *Human Movement Science*, 10:193–221, 1991.

[58] R. Plamondon and G. Lorette. Automatic signature verification and writer identification, the state of the art. *Pattern Recognition*, 22(2):107–131, 1989.

[59] R. Plamondon and F.J. Maarse. An evaluation of motor models of handwriting. *IEEE Transaction on systems, man, and cybernetics*, 19(5):1060–1072, 1989.

[60] F.E. Pollick and G. Sapiro. Constant affine velocity predicts the 1/3 power law of planar motion perception and generation. *Vision Research*, 37(3):347–353, 1997.

[61] F.K.H. Quek. Eyes in the interface. *Image and Vision Computing*, 13(6):511–525, 1995.

[62] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Inc., 1993.

[63] L.R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[64] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, Signal Processing*, 26(1):43–49, 1978.

[65] Y. Sato and K. Kogure. On-line signature verification based on shape, motion and writng pressure. *Proc. 6th Int. Conf. on Patt. Recognition*, pages 823–826, 1982.

[66] B. Serra. *Reconnaissance et localisation d'objets cartographiques 3D en vision aérienne dynamique*. PhD thesis, L'Université de Nice - Sophia-Antipolis, 1996.

[67] B. Serra and M. Berthod. Subpixel contour matching using continuous dynamic programming. *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, pages 202–207, 1994.

[68] B. Serra and M. Berthod. Optimal subpixel matching of contours chains and segments. *Proc. 5th Int. Conf. Computer Vision*, pages 402–407, 1995.

[69] B. Simard, B. Prasad, and M.K. Sinha. On-line character recognition using handwriting modelling. *Pattern Recognition*, 26(7):993–1007, 1993.

[70] C.G. Small. *The statistical theory of shape*. Springer-Verlag, Inc., 1996.

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

[72] C.J. Taylor, T.F. Cootes, A. Lanitis, G. Edwards, and P. Smyth et al. Model-based interpretation of complex and variables images. *Philosophical transactions of the Royal Society of London*, 352(1358):1267–1274, 1997.

[73] A. Tomita, T. Ebina, and R. Ishii. A gui-interaction aiding system for cut-and-paste operation based on image processing for the visually impaired. *IEICE Transactions on Information and Systems*, E81D(9):1019–1024, 1998.

[74] H.L. Van Trees. *Detection, Estimation and Modulation Theory: Part I.* John Wiley & Sons, Ltd., 1968.

[75] M. Turk and A. Pentland. Eigenfaces for recognition. *J. of Cognitive Neurosci.*, 3(1):71–86, 1991.

[76] P. Viviani and G. McCollum. The relation between linear extent and velocity in drawings movements. *Neuroscience*, 10(1):211–218, 1983.

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

[78] J. Vredenbregt and W.G. Koster. Analysis and synthesis of handwriting. *Philips Tech. Rev.*, 32(3/4):73–78, 1971.

[79] P.D. Wellner. Adaptative thresholding for the digitaldesk. *Technical Report EPC-1993-110*, 1993.

[80] P.D. Wellner. Self calibration for digitaldesk. *Technical Report EPC-1993-109*, 1993.

[81] B. Wirtz. Stroke-based time warping for signature verification. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 179–182, 1995.

[82] L. Wiskott, J.M. Fellous, N. Kruger, and C. Von der Malsburg. Face recogniton by elastic bunch graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.