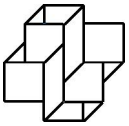


Laboratório Nacional de Computação Científica
Programa de Pós Graduação em Modelagem Computacional

Modelos e Métodos para Interação Homem-Computador Usando Gestos Manuais

Por

Albino Adriano Alves Cordeiro Junior

PETRÓPOLIS, RJ - BRASIL
NOVEMBRO DE 2009

MODELOS E MÉTODOS PARA INTERAÇÃO HOMEM-COMPUTADOR USANDO GESTOS MANUAIS

Albino Adriano Alves Cordeiro Junior

TESE SUBMETIDA AO CORPO DOCENTE DO LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM MODELAGEM COMPUTACIONAL

Aprovada por:

Jauvane Cavalcante de Oliveira, Ph.D.

Jose Ricardo de Almeida Torreão, Ph.D.

José Antônio Apolinário Junior, Ph.D.

Raúl A. Feijóo, Ph.D.

PETRÓPOLIS, RJ - BRASIL
NOVEMBRO DE 2009

XXXX

Tese (D.Sc.) – Laboratório Nacional de Computação Científica – LNCC/MCT, 2009.

CORDEIRO JUNIO
Modelos e M
Albino Adriano Alv
xx, yy p. : il.
Orientadore(s):Jauv

1. Gesture Re
I. LNCC/MC

The simplification of anything is always sensational.
(Gilbert K. Chesterton)

The things that we love tell us what we are.
(St. Thomas Aquinas)

À minha primeira família, meus amados pais e irmão, à nova família que iniciei com Cristianne e a todos os meus avôs e avós, tios e tias, primos e primas que me são tão caros! A vocês dedico este trabalho.

Agradecimentos

I would like to express my sincere gratitude initially to my supervisors Jauvane, Nicolas and Marcelo that, with immense patience and wisdom, supported me and guided me through this long and amazing journey. Dear professors, thank you! The opportunity you gave me will shape all my future. I give a special thank to Prof. Nicolas, who gave me and my wife the wonderful experience of living in Canada, by opening his lab doors for me in Ottawa.

I am deeply grateful to my loving wife. Her sweet presence and words and love and example of strength and everything else would, so many times, turn unproductive moments of despair into very productive moments of enthusiasm.

I want to thank my mother and father. The unconditional love I perceive in their eyes is my courage to always aim high. I am also grateful for my brother and his wife and kids who have always been a light to which I look up to. I specially thank my grandmother (voinha) for her persevering love and prayer that made a world of a difference in every moment in my life.

My friends at LNCC in Petrópolis and DISCOVER in Ottawa. When I grow up, I want to be just like you guys! I learned a lot from you and from our conversations. In particular Fábio, Dilberto, Leonardo, Priscila, Adilson and Magno in LNCC and François, Qing, Ayman and Jilin in DISCOVER, thanks a lot.

Many thanks goes to the Ana's (Paula and Neri) at the CFRH/LNCC for always being ready to help the students, including me.

My incessant gratitude is definitely due to God, who generously gave me so much in life as well as the privilege of life itself. I thank God for revealing Himself in the Person of our Savior uncovering, this way, extraordinary meaning and beauty hidden in everything, even ordinary ones. Thank you my Lord, for everything!

Resumo da Tese apresentada ao LNCC/MCT como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

MODELOS E MÉTODOS PARA INTERAÇÃO HOMEM-COMPUTADOR USANDO GESTOS MANUAIS

Albino Adriano Alves Cordeiro Junior
Novembro, 2009

Orientador(es): Jauvane Cavalcante de Oliveira, Ph.D.

Esta tese aborda o problema de *entender* videos digitais algoritmicamente aplicado ao design de sistemas de Interação Homem-Computador (HCI do Inglês: Human-Computer Interaction) baseados na postura e movimento da mão. Tais sistemas são frequentemente referidos como um tipo de *Interface Perceptual* com o usuário (PUI do Inglês: Perceptual User Interface), que é uma interface que habilita o computador a detectar e reconhecer ações dos usuários de forma ativa. Acredita-se que PUI é um paradigma que irá suplementar o padrão atual, as Interfaces Gráficas com o Usuário (GUI do Inglês: Graphical User Interfaces), que são baseadas em mouses e teclados para entrada do usuário.

A principal motivação da pesquisa feita em HCI por gestos manuais é habilitar as pessoas a interagir de uma forma mais natural com dispositivos computacionais, por exemplo, ao permitir que usuários manipulem programas, arquivos e pastas de computador de uma forma similar ao manuseio de objetos físicos familiares.

Neste trabalho é proposto um ferramental para rastreamento da mão –posição e rotação no plano– assim como para reconhecimento de postura da mão a partir dos contornos da mão. Uma nova abordagem de processamento de pixels baseada em *aprendizagem de máquina* forma o bloco fundamental para um método *level set* de extração de contornos, tão bem como para um módulo de mensuração do rastreador, que é formulado como um problema de filtragem em espaço de estados onde a dinâmica do sistema é modelada com sistemas lineares com saltos markovianos. Baixas taxas de erro de classificação de postura são alcançadas com o uso de um descritor de formas baseados em medidas invariantes de momentos bidimensionais.

Abstract of Thesis presented to LNCC/MCT as a partial fulfillment of the requirements for the degree of Doctor of Sciences (D.Sc.)

MODELS AND METHODS FOR HUMAN-COMPUTER INTERACTION USING HANDS GESTURES

Albino Adriano Alves Cordeiro Junior
November, 2009

Advisor(s): Jauvane Cavalcante de Oliveira, Ph.D.

This thesis addresses the problem of algorithmic *understanding* of digital video applied to the design of Human-Computer Interaction (HCI) systems based on hand posture and motion. Such systems are often referred as a type of *Perceptual* User Interface (PUI), which is an interface that enables the computer to detect and recognize users' actions in an active way. PUI is believed to be a paradigm that is going to supplement the current standard Graphical User Interfaces(GUI), that are based on mice and keyboards for user input.

The main motivation of the research done in hand-gesture HCI is to enable people to interact in a more natural way with computational devices, for example, by letting the users manipulate computer programs, files and folders in a way that resembles the handling of familiar physical objects.

In this work a toolset is proposed for hand tracking -position and in-plane rotation- as well as posture recognition from hand contours. A novel approach to pixel-level processing based on *machine learning* forms the fundamental building block of a level set contour tracking method, as well as for the measurement module of the tracker, which is formulated as a filtering problem in state-spaces where the dynamics is modeled with Markov jumps linear systems. Low error rates are achieved for posture classification using a shape descriptor based on 2D moments invariant measures.

Contents

1 Introduction 1
- 1.1 Study subject, objectives and relevance . . . 1
- 1.2 Contour tracking . . . 5
- 1.3 Posture classification . . . 7
- 1.4 Motion estimation . . . 8
- 1.5 Summary of contributions . . . 8

2 Literature Review 9
- 2.1 Application domains . . . 9
- 2.2 Gesture types . . . 13
- 2.3 Gesture analysis and interpretation literature . . . 15
  - 2.3.1 Appearance-based methods essentials . . . 15
  - 2.3.2 Model-based methods essentials . . . 17
- 2.4 Feature extraction . . . 20
  - 2.4.1 Haar-like features . . . 21
  - 2.4.2 Optical flow . . . 22
  - 2.4.3 Edge features . . . 22
  - 2.4.4 Statistical features . . . 24
  - 2.4.5 Feature selection algorithms . . . 24
  - 2.4.6 Extraction based on machine learning . . . 25
- 2.5 Machine learning theory . . . 27
- 2.6 Bayesian tracking . . . 28
  - 2.6.1 The Kalman filter (KF) . . . 29
  - 2.6.2 The particle filter (PF) . . . 30
  - 2.6.3 Markovian jumps linear systems (MJLS) . . . 30
- 2.7 Level-set methods for contour tracking . . . 30
  - 2.7.1 Robust contour evolution . . . 31
  - 2.7.2 Fast contour evolution . . . 31
- 2.8 Shape features . . . 31

3 Low-level Feature Extraction 34
- 3.1 Motivation . . . 34
- 3.2 The codification of prior knowledge . . . 36
- 3.3 Learning machines . . . 36
- 3.4 Tools for machine learning analysis . . . 37
  - 3.4.1 Vapnik-Chervonenkis (VC) dimension . . . 37
  - 3.4.2 Vapnik-Chervonenkis (VC) confidence . . . 38
  - 3.4.3 Margins . . . 38
  - 3.4.4 The kernel trick . . . 39
- 3.5 SVMs for supervised learning . . . 41

3.5.1 Optimal separating hyperplane . . . 41
3.5.2 The support vector machine . . . 42
3.6 Pixel classification . . . 44
3.6.1 Color spaces . . . 48
3.6.2 Similarity measures . . . 48
3.7 SVM-based pixel classification . . . 48
3.7.1 Feature design . . . 48
3.7.2 Background subtraction for free . . . 49
3.8 Manual image annotation . . . 49
3.9 Segmentation experiments . . . 51
3.9.1 Window size . . . 51
3.9.2 Training-error penalty parameter . . . 53
3.9.3 Covariance features . . . 57
3.9.4 Background features . . . 57
3.9.5 A 3D-grid search . . . 57

4 Level Set Method for Contour Tracking 62
4.1 Implicit curve representation . . . 62
4.2 Level set methods . . . 65
4.3 A fast level set approximation algorithm . . . 66
4.3.1 The data-driven cycle . . . 68
4.3.2 The regularization cycle . . . 68
4.4 Contour tracking experiments . . . 70
4.4.1 Initialization . . . 71
4.4.2 The “hall monitor” sequence . . . 71
4.4.3 The rock-paper-scissors sequence . . . 74
4.4.4 The 3D Navigation sequence . . . 74
4.5 A discussion on the algorithm’s complexity . . . 74

5 Posture Recognition 76
5.1 Hu’s moments invariants . . . 76
5.2 Implementation and Experiments . . . 77
5.2.1 The rock-paper-scissors game . . . 78
5.2.2 A 3D navigation system . . . 78
5.3 A discussion on alternatives to Hu’s invariants . . . 78

6 Hand Motion Estimation 84
6.1 Contour representation and the tracking problem formulation . . . 84
6.1.1 A B-splines contour representation framework . . . 84
6.1.2 State space filtering formulation . . . 86
6.2 Modeling dynamics with Markov jump linear systems and the Costa filter . . . 87
6.3 The measurement module . . . 89
6.3.1 Contour fitting . . . 90
6.4 Important implementation issues . . . 90
6.4.1 Data normalization . . . 90
6.4.2 System identification . . . 92
6.4.3 Mode transition probability modeling and Markov chain generation . . . 93
6.4.4 Observation noise modeling . . . 94
6.5 Monte Carlo simulations . . . 94

7 Conclusion 99
7.1 Summing up 99
7.2 Future work 100

Referências Bibliográficas 101

List of Figures

Figure

1.1 Smartphone Embeded Projector . . . 2
1.2 3D Connexion SpacePilot . . . 3
1.3 3D navigation gestures. . . . 4
1.4 Main blocks of a contour framework GPUI. . . . 6
1.5 Hand contours. Note that we can understand that they are hand contours and we can also tell their postures, nevertheless, there is no texture nor color information. 6

2.1 An organisation diagram for categorizing and classifying Gesture-based HCI research literature (from Karam [2006]). . . . 10
2.2 Augmented Reality Basic Setting in Buchmann et al. [2004]. . . . 11
2.3 CAVE Examples. . . . 12
2.4 HONDA's famous ASIMO Robot understands simple hand gestures (`http://world.honda.com/ASIMO/technology/intelligence.html`). . . . 12
2.5 Virtual Keyboard Examples. . . . 13
2.6 Gestures involving a combination of body parts. . . . 14
2.7 Examples of static and dynamic gestures accordingly with the convention adopted here. . . . 15
2.8 A Gesture Taxonomy (from [Pavlovic et al., 1997]). . . . 16
2.9 Basic Gesture Recognition Schemes. . . . 18
2.10 Examples of 2D and 3D geometrical hand models. . . . 19
2.11 The value of the integral image at point (x, y) is the sum of all the pixels above and to the left (from Viola and Jones [2001b]). . . . 21
2.12 Haar-like Features Representation. . . . 23
2.13 Usual visualization optical flow (from `http://www.cs.umd.edu/~sontran/733/small_flow.png`). . . . 23
2.14 Canny edge detector output example. . . . 24
2.15 Hand tracking based on optical flow (from Kölsch and Turk [2005a]). . . . 25
2.16 Feature extraction based on linear search on contour normals (from Isard and Blake [1998a]). . . . 26
2.17 In the state-estimation problem the true states are hidden only accessible through the noise observation process. . . . 28
2.18 The 2D contour is defined as the intersection of the surface with the zeroth-level plane. . . . 32

3.1 Prior knowledge and generalization ability are essential in visual understanding. . 35
3.2 Three points in $\Re^2$, shattered by oriented lines (from Burges [1998]). . . . 37
3.3 Linear classifier (thick line) and margin (the minimal distance between any training points to the hyperplane). . . . 39
3.4 Two-dimensional classification example (from Schölkopf and Semola [2001]). . . . 40
3.5 Illustration of a non-linear partition of data with SVM. . . . 42

3.6 Low-level Features Extraction 45
3.7 The pixel values distribution for a hand image and the background in the RGB and CIE Lab color spaces. Only two thousand randomly sampled pixels are displayed. The bigger red balls are hand values and the smaller green ones are background values. 47
3.8 A color sample is used for computing the covariance matrix used in the calculation of the Mahalanobis distance. 48
3.9 Hand annotation by hand. 50
3.10 Images used in the experiments. 52
3.11 Window size influence on error rates. Every point in the plots is the arithmetic mean between the error obtained using variance and the one using covariance features. 53
3.12 Effect of neighborhood size on classification robustness. C fixed equal to 10^3. . . 54
3.13 False positive versus C 55
3.14 Time used for the classification of all pixels in an image versus C 55
3.15 Effect of the penalty parameter on classification. Using a 16×16 fixed window. . 56
3.16 Effect of covariance features for various windows sizes. C fixed equal to 10^3. . . . 58
3.17 Effect of background subtraction features. C fixed equal to 10^3 and window dimensions equal to 16×16. 59
3.18 Total error (false negatives plus false positives). The error data was rescaled so that the smaller errors are represented by the bigger red spheres and as the error increases the sphere gets smaller and its color shifts towards the blue. 60
3.19 False positives. The error data was rescaled so that the smaller errors are represented by the bigger red spheres and as the error increases the sphere gets smaller and its color shifts towards the blue. 61

4.1 Examples of a simple and self-intersecting curve. 63
4.2 Level-set-like contour evolution (from Shi and Karl [2005a]) 67
4.3 Posture Classification Information Flow 71
4.4 Level set initialization. Red is outside, dark blue inside region, yellow L_{out}, baby blue L_{in}. 72
4.5 Results of the hall monitor sequence. The right column are the results obtained with the machine learning method proposed in this thesis and the left column the result obtained in Thida et al. [2006]. 73
4.6 Contour tracking with an "easy" background and approximately constant appearance. 75
4.7 3D Navigation sequence. Varying illumination demands training data that contains representative samples of the various illumination conditions. 75

5.1 Rock-paper-scissors game hand-postures 79
5.2 The classification rate in this sequence was around 98.5%. 80
5.3 More classes and noisy contour extraction still yields a fairly good accuracy (88.7%) 81
5.4 Bin distribution for shape context computation (from Belongie et al. [2002]). (a) and (b) are the sampled edge points of the two shapes. (c) is the diagram of the log-polar bins used to compute the shape context. (d) is the shape context for the circle, (e) is that for the diamond, and (f) is that for the triangle. As can be seen, since (d) and (e) are the shape contexts for two closely related points, they are quite similar, while the shape context in (f) is very different. 83

6.1 Basic elements of the tracking system. I_k is the k^{th} input video frame, $\mathbf{y}_k$ is the measured state and $\hat{\mathbf{y}}_k$ is the filtered measurement which will hopefully have a lower level of noise on average. 86

6.2 Visualization of the contour points measurement scheme in action. 90
6.3 Our simulation platform can generate an arbitrary number of random paths for the hand template with stochastic properties conveniently defined by the user. . 96
6.4 Root mean square error comparison between filters with fixed path. REF, KJ, CF, IMM are respectively the unfiltered measurement, the Kalman filter, Costa filter and IMM average errors. 96
6.5 Root mean square error comparison with random paths (θ generated with Algorithm 6). 97
6.6 The average RMS errors for 400 time steps long MJLS random walks for different values of the mode jump probability. 97
6.7 Shows the influence of process noise on the filters performances. 98

List of Tables

Table

3.1 Kernel Functions 44

6.1 Hand crafted dynamic modes. We used $c = 100$, $v = 10$, $a = 0.1$ and $t \in [0, 2\tau]$ (with $\tau = 100$) which assures that the motion begins and finish at the same point and the visual representation remains inside a 500×500 window. ρ was fixed equal to 2, which gives a maximum of twice the original scale. 95

Chapter 1

Introduction

1.1 Study subject, objectives and relevance

This thesis addresses the problem of algorithmic interpretation of digital video applied to the design of Human-Computer Interaction (HCI) systems based on hand posture and motion. Such systems are often referred as a type of Perceptual User Interface (PUI), which is an interface that enables the computer to actively detect and recognize users' actions. PUI is believed to be a paradigm that is going to supplement the current standard Graphical User Interfaces (GUI), that are based on mice and keyboards for user input. The main motivation of the research done in hand-gesture HCI is to enable people to interact in a more natural way with computational devices, for example, by letting the users manipulate computer programs, files and folders in a way that resembles the handling of familiar physical objects.

PUIs based on hand motion and gesture recognition, which I refer to as GPUI hereafter, have a wide range of applications such as enabling very young children to interact with computers, gaming, navigation in virtual environments, sign language recognition and many others [Mitra and Acharya, 2007]. As a general interface to computing devices, GPUIs are particularly useful in situations where people are in transit, opposed to being at their work or home desk, for example, navigation interface for multimedia kiosks in auto shows, art galleries, museums, tourist attractions, congresses and other events. GPUIs are likely to play an important role adding a new dimension to standard digital signage –the interaction dimension. Basically, digital signage is the fast growing market of advertisement using flat screen digital displays that we frequently see nowadays in shop windows, airports, snack bar's wall menus, university campi, and other public spaces. Its current state of the art is characterized by high quality commercial software for content creation and network content distribution that lacks, however, channels for user feedback.

There are various signs of an increase in relevance of GPUI R&D, for instance, powerful mobile phones aiming at Internet navigation and email communication, called smartphones, are selling more than 100 million units per year and seeing their processing power increase every year. The clear trend here is that society is moving toward ubiquitous computing in a fast pace [Abowd and Mynatt, 2000], that is, there is a great demand for technology to enable people to use services associated with computers anywhere and on the move. This trend has two immeadiate consequences:

- More people will use computers or will use computers more often;
- Individuals will increasingly need computers in situations where mouse, keyboard and standard monitor are non-practical.

We can deduce from this that there will be an even greater consumer demand for the development of new HCI technologies that can be based on speech recognition, touch-screens, motion sensors, micro projectors (see Figure 1.1), and specially GPUIs.

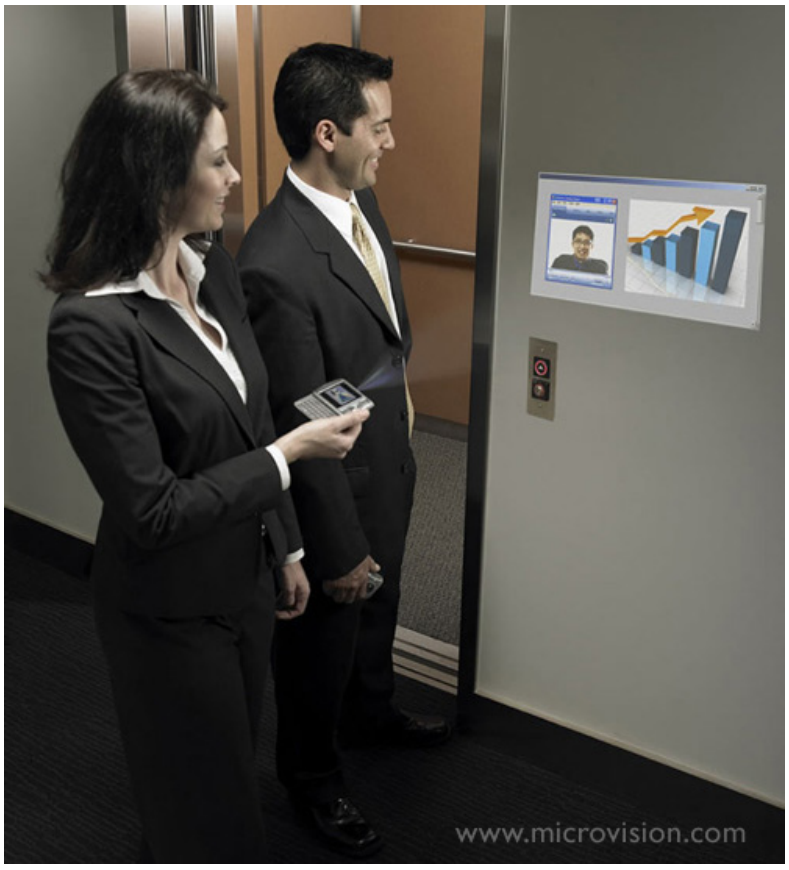


Figure 1.1: Smartphone Embeded Projector

Vision-based hand tracking (and gesture recognition) theory is still immature, in the sense that there is no core mathematical formulation that is broadly adopted by the expert community. The problem itself can be modeled with rather different perspectives and emphases in mind, and each of these can be approached with completely different methodologies and mathematical tools.

Figure 1.2: 3D Connexion SpacePilot

This thesis focus on gestures for natural interaction with computational devices. To help form a more precise picture of the type of hand motion and postures[1] considered here let us consider a hypothetical example of a vision based replacement for a 3D mouse like the 3D Connexion SpacePilot (see Figure 1.2) as a study case.

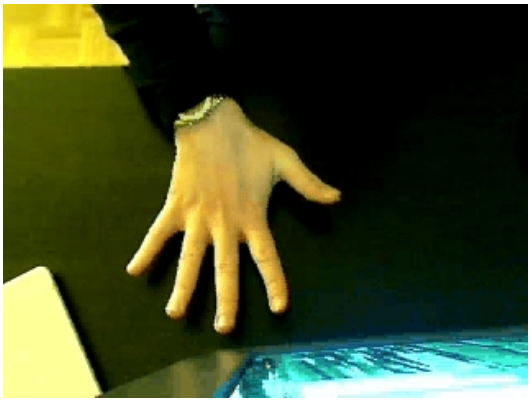
3D mouse devices are growing in importance nowadays, as more and more computer applications in 3D CAD modeling, 3D Virtual Worlds like Second Life, 3D Animation industry, Google Earth, e-commerce (for 3D visualization of the products), etc., witness a popularity boost. The basic idea is that the user is not restricted to navigation in the plane of the monitor, but by pushing down or pulling up the knob you gain a third free control dimension. Added to that, the user can also twist one way or the other for rotation commands -a fourth dimension.

In order to replace the basic functions of the 3D mouse mentioned above, we propose the gestures in Figure 1.3. Note that these gestures (five total) are, in fact, defined by a single posture.

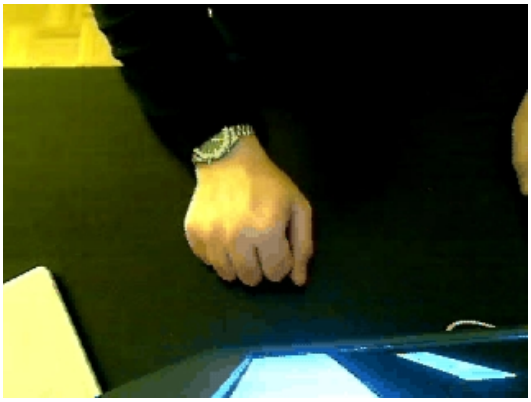
For the sake of clarity, throughout this thesis whenever motion estimation, trajectory or posture state is mentioned it is considered the composition of orientation, 2D position and scale (distance from the camera) of the hand of an arbitrary posture, that is, given a posture (pointing for instance) we associate a 4-dimensional vector that defines the hand major axis' orientation, position of the center of mass and scale relative to a referential posture state.

Depending on how the user is expected to use the hands, or the type of gestures the system is supposed to recognize, emphasis might be needed on robust and accurate posture classification or on robust and accurate motion estimation. For the 3D navigation study case considered here, the time the user spend in each posture is roughly random, in contrast with sign language recognition for sign language gestures have very characteristic timing and motion, actually resembling the way we use vocal words in conversations one after the other in a certain rhythm in order to build a sentence.

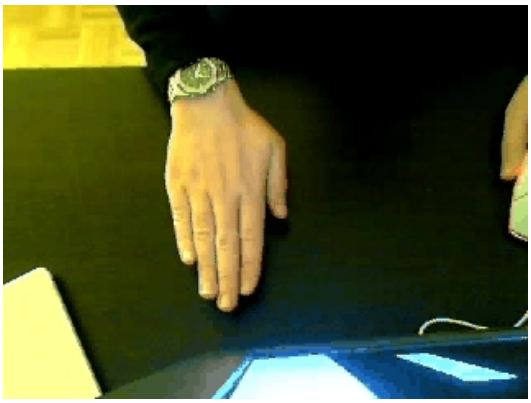
[1] Posture is defined in our work as the configuration of the articulations of the hand or, in other words, how the fingers are arranged.



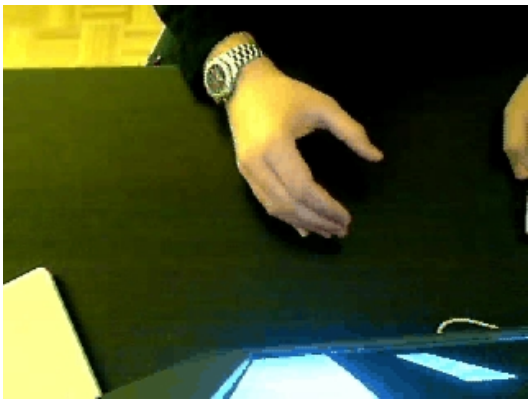
(a) Down (perpendicular going into the monitor).



(b) Up (perpendicular going out the of monitor).



(c) Move the mouse pointer around with the hand.



(d) Rotate counter-clockwise

(e) Rotate clockwise

Figure 1.3: 3D navigation gestures.

With the type of gestures of the 3D navigation system in mind, I advocate here the use of contours as the basic feature for gesture recognition, that is, a GPUI framework based on a contour classifier; Its main building blocks are illustrated in Figure 1.4. The choice of contours is inspired in Nature. The human vision system is able to extract a large amount of visual information from contours. An intuitive demonstration of this is that most people are able to understand and recognize the hand posture of Figure 1.5 even though they lack texture and color. This signalizes two things: first it shows that the human visual cognition system is extremely powerful and it also shows that contours can carry a large amount of visual information about the object.

Another advantage of contours is that it is a fairly compact representation of the object. Computer representation of contours is a list of 2D points, usually with well less than one percent of the total number of points in the image. Moreover, there is a solid body of theory for the extraction of high-level features from contours that are invariant to orientation, 2D position and relative distance from the camera (scaling). Notice that our framework uses contours as an intermediate feature, such that for final posture classification we compute a vector of features from contour points that has a very small and fixed dimension (much less than the number of points in the contour). Moreover, trajectory measurements can be made from contours.

Considering the contour-based framework briefly described above, the problem of gesture recognition is put mainly in terms of four main subproblems: contour extraction or tracking, posture classification, motion estimation and sequence analysis (gesture recognition). My contributions were for the first three subproblems and the remaining of this chapter discusses briefly each of these.

1.2 Contour tracking

Feature extraction is the most critical part in most computer vision systems. Notice that many gesture recognition systems reported in the literature will work in controlled laboratory environments, and fail completely when taken *outside* and that is because variation in illumination conditions and cluttered background affect image features in a very complex and unpredictable way.

It is important to remind that in, our framework, the input to both the posture classification and the motion estimation modules are contours, therefore, the gesture recognition correct classification rate is doubly dependent on the level of noise distortion of the contour extraction process.

Another important issue at this stage is that contour extraction, and feature extraction in general, is in part an operation on raw-data which makes it likely to be the bottleneck in overall processing times. This is particularly critical in real-time video analysis applications, such as gesture recognition for which data throughput is extremely high.

For single images, deformable-objects contour extraction is equivalent to object segmentation or just segmentation. Contour extraction of a video sequence, however, is usually called *contour tracking* in the literature. There are different kinds of object segmentation and deformable-object contour tracking approaches, however, the aimed final result is the same, even if implicitly: the classification of pixels in the image as belonging to the object or to the background. Indeed, pattern classification based contour extraction is one of the innovative aspects of our work, given that the current standard approach to segmentation does not go through pattern classification theory and tools for pixel-level classification as it is done here.

In part, researchers have avoided using strong classifiers, such as support vector machines, for raw-data processing, due to worries about processing times being too high, or yet, because pattern classification tools have been traditionally associated with more high level patterns such as shapes, faces, human gait, etc.. In summary, pattern classification has been used mostly at the end of the pipeline of the computer vision system to classify information that is acquired using

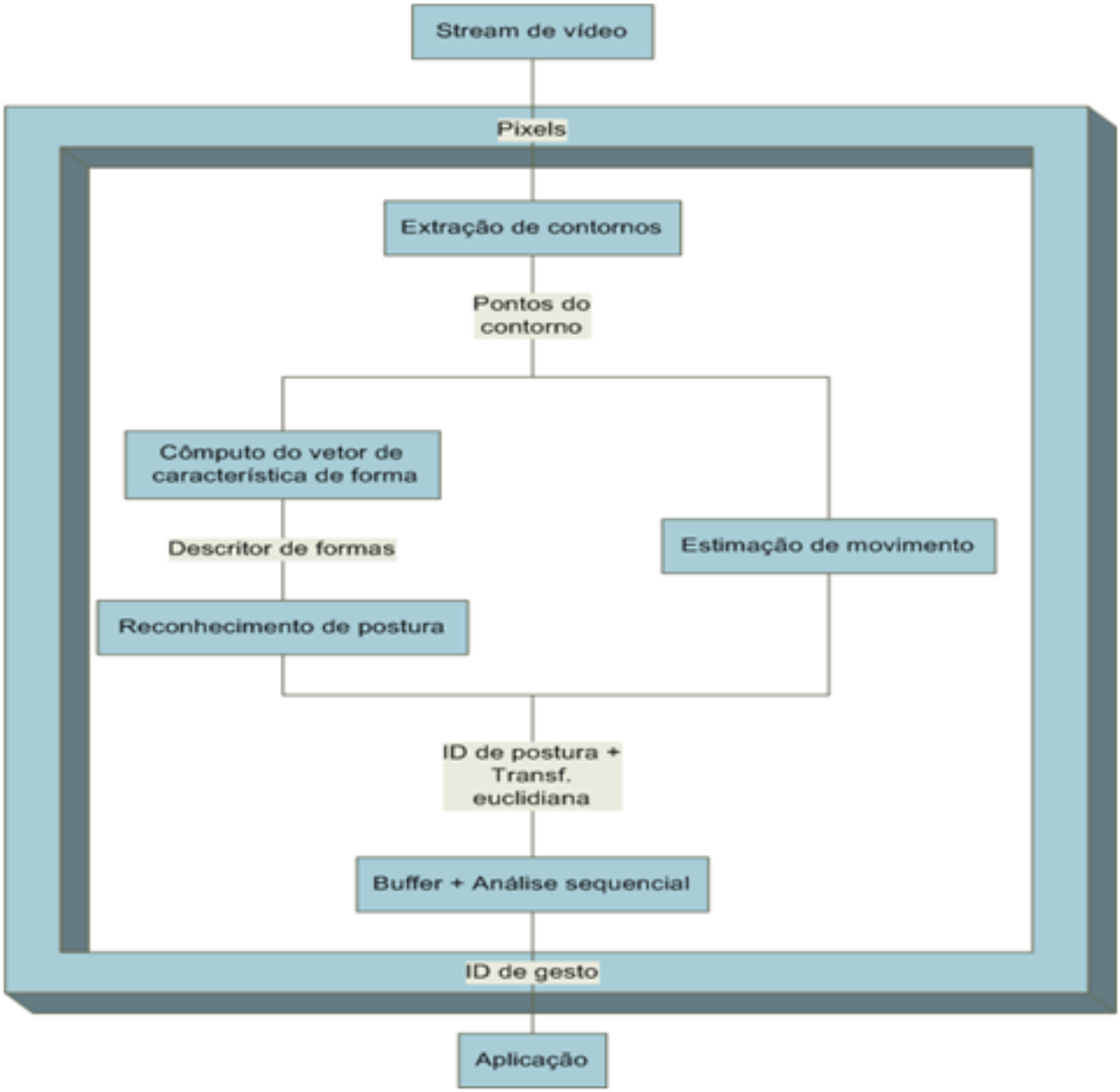


Figure 1.4: Main blocks of a contour framework GPUI.

Figure 1.5: Hand contours. Note that we can understand that they are hand contours and we can also tell their postures, nevertheless, there is no texture nor color information.

pure computer vision techniques. Although some attempts to use pattern classifiers to produce a feature map for tracking can be found in the literature [Avidan, 2004], there is clearly much more to be explored in this line.

One of this thesis' claims is that a real-time contour tracking using a strong classifier for pixel-level classification is viable and that our proposed contour tracker proves just that.

We can always apply an object segmentation algorithm to every frame in a video sequence and therefore obtain a contour tracking algorithm. In video streams, there is a strong correlation between frames close in time. For instance, objects tend to change position in a continuous fashion and in consecutive frames variability of postures and poses tends to be small.

Our tracker combines a support vector machine (SVM) classifier with a very fast level-set-like algorithm to exploit inter-frames correlation in order to minimize the number of classifications executed every frame, obtaining this way a robust and fast algorithm, notwithstanding, simple to understand and implement.

1.3 Posture classification

In general, posture classification is a problem of *pattern classification* theory, which is a problem in the field of *computational learning theory*. The later is the theory that explores ways of estimating functional dependencies from a given data collection [Vapnik, 2000].

In practical terms, when we speak of a pattern classification method here, we are referring to an underlying machine learning algorithm that can scan a collection of example patterns and their correct classification (labels) and learn a classification rule from it. The *learned* or *trained* machine can, then, be applied to previously unseen or *test* patterns. This is what the machine learning community generally calls supervised inductive learning.

Independently of the classification method used, the first challenge to be addressed is the choice of a set of features that are invariant to orientation, 2D position of the center of mass, and scale. This is necessary to make the learning and classification process more manageable since it avoids the need to train the classifier for a great number of different views (poses) of the same posture. Our approach make use of the theory of moment invariants as developed by Hu and revised by Reiss [Hu, 1962] [Reiss, 1991]. The concept of moments had been used extensively in classical mechanics and statistics, nevertheless, Hu extended the theory of 2D moments making it a very useful tool in vision pattern recognition. 2D moments are weighted measures, which in our case, is given by the simplified expression:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q dx dy \tag{1.1}$$

where p and q are the orders, $f(x, y)$ is the image function. This is a rather general formula. In practice since we will be working with contours in real digital images, the image function is going to be equal to zero everywhere except in the contour points for which it is defined equal to one, thus the equation can be simplified to

$$m_{pq} = \sum_{(x,y)\in C} x^p y^q \tag{1.2}$$

where C is the set of contour points. One important entity in moments theory is the characteristic function

$$\phi(\upsilon, \nu) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{ix\upsilon + iy\nu} f(x, y) dx dy \tag{1.3}$$

where $i = \sqrt{}$which can be written as a power series of the moments

$$\phi(\upsilon, \nu) = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} m_{pq} \frac{(i\upsilon)^p}{p!} \frac{(i\nu)^q}{q!} \tag{1.4}$$

That established, it can be proved that $f(x, y)$ can be recovered from moments via

$$f(x, y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-ix\upsilon - iy\nu} \phi(\upsilon, \nu) d\upsilon d\nu \tag{1.5}$$

That means that the total set of moments carries all there is to know about $f(x, y)$. There is a solid body of theory for deriving invariant measures from moments, for instance, Hu [1962] devised a method for computing invariant measures from moments based on the theory of algebraic invariants. In this work, Hu's moment invariants will be used for shape classification and evaluated against other features that can be extracted from contours such as Zernike's moment invariants [Arvacheh and Tizhoosh, 2005] and shape contexts [Belongie and Malik, 2000].

In the case of a small sets of easily distinguishable postures, more complex pattern classification tools are actually unnecessary and direct matching (naive nearest neighbor procedure) of the feature vectors will do the job. Different combinations of machine learning algorithms and features will be later compared for performance.

1.4 Motion estimation

If on one hand feature invariance to affine transformation is necessary to make the classification problem manageable, to discard trajectory information is not wise. Motion itself can communicate a good deal of information and can help substantially in providing naturalness to the gesture interaction.

There are different types of gestures. We reason here that natural gestures with the purpose of communicating a specified command will present in many cases very characteristic dynamic behavior, in a way that the dynamic signature of the gesture is inseparable from its meaning, for example, waiving good-bye is defined by the characteristic motion as is by the palm posture. We expect every gesture to be associated with a certain hand motion and in regular use the hand is likely to be changing abruptly from one type of motion to the other (as state changes in a Markov chain). We assume that each motion type dynamic can be modeled by a linear state-space system, therefore, the overall dynamics can be modeled with Markovian Jumps Linear Systems (MJLS).

Based on the MJLS model, we propose a motion estimation system based on a linear optimal filter for MJLSs [Costa et al., 2005]. The motion estimation system can be seen also as rigid-object contour tracking algorithm.

1.5 Summary of contributions

In our work we attempt to further advance the state-of-the-art by proposing solutions to some problems associated with GPUIs based on contour tracking and posture classification. Our main contributions are:

i. A contour based framework for GPUIs

ii. A novel object color model based on machine learning (SVMs) for pixel level feature extraction

iii. A real-time deformable-object contour tracking method based on the model from ii and approximated Level set method.

iv. A motion estimation algorithm based on a Markovian Jump Linear Systems formulation and the Costa filter

Arising from the thesis, we have to the moment one proceedings publication [Cordeiro Junior et al., 2008] which comprises the results related with pose tracking using MJLS modeling and filtering and two journal papers in process of submission.

Chapter 2

Literature Review

Among all human body organs, hands are the ones that people use the most and with the most ease to interact with the world around. Hands are highly articulated objects with roughly 27 degrees of freedom (DOF). Hands are able to translate brain commands into action in a timely and seamless fashion, often in order to deliver very complex performances as displayed by people communicating in sign language or a virtuoso pianist at work.

It is common knowledge that the credit for the personal computer revolution experimented in the last decades was due in part to the introduction of the popular GUI paradigm using the mouse. The current reality of unprecedented high level of computational power and low-cost digital cameras available for the masses promises to be the perfect environment for the introduction and popularization of novel human-computer interfaces that use hands in an intuitive fashion. That prospect has driven a great deal of R&D effort in the last two decades, both in industry and academia.

Not surprisingly, there has been a surge of technical publications on hand tracking, gesture recognition and related topics in the same period. An exhaustive literature review in gesture interpretation is, thus, a daunting challenge due to the immense number of published works and variability of approaches; however, this is not my goal here. This section presents some trends in the research literature as well as representative papers which were influential to this PhD project.

For a more exhaustive analysis of the literature I suggest the seminal work by Karam [2006], in which the author proposes, among other things, a matrix (Figure 2.1) of common high-level characteristics and categories for comparing gesture interaction research and systems, which was methodically used to analyze a vast amount of papers on the topic.

2.1 Application domains

Gesture recognition has a great number of potential applications, ranging from Augmented Reality interaction through sign language translation for mobile platforms to Board game control. At the current state of the technology, the underlying application often dictates the design approach and tools.

A natural application ground for gesture recognition technology is in augmented reality (AR). Augmented or mixed reality is an emerging technology that deals with visual combination of the real-world and computer generated data. The state-of-the-art AR system usually involves *ad-hoc* head-mounted displays (HMD) connected to a computer that enables the user to see the world around as well as superimposed computer imagery (see Figure 2.2). For instance, Yao et al. [2004] proposes a bare-hand controlled augmented reality (AR) map navigation system. A set of symbolic hand gestures is defined as controlling commands. Users can directly move their hands on a real map. When certain hand gestures are recognized the relative geography information, video clip or 3D scene will be played based on some display devices such as the HMD and computer screen. Lee and Hollerer [2008] describes a novel markerless camera tracking

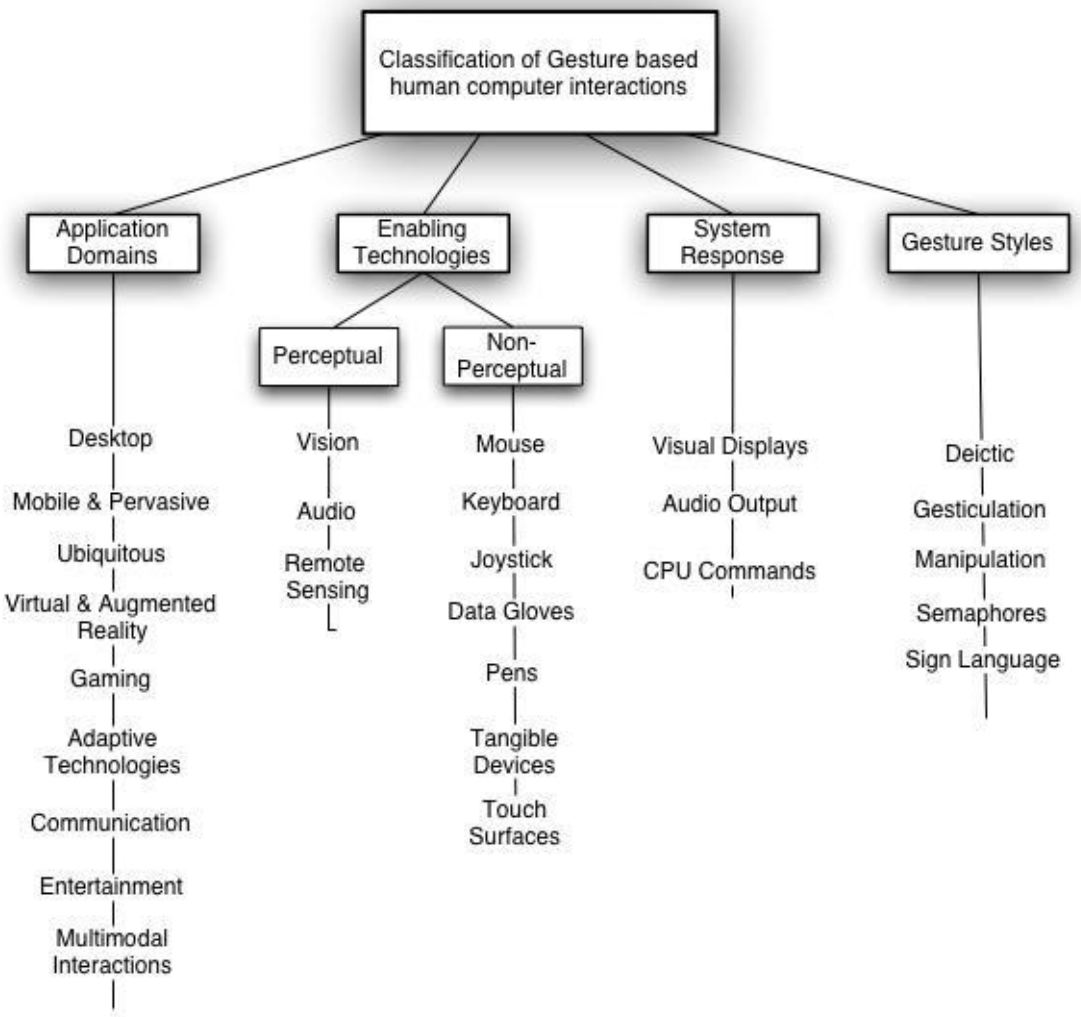
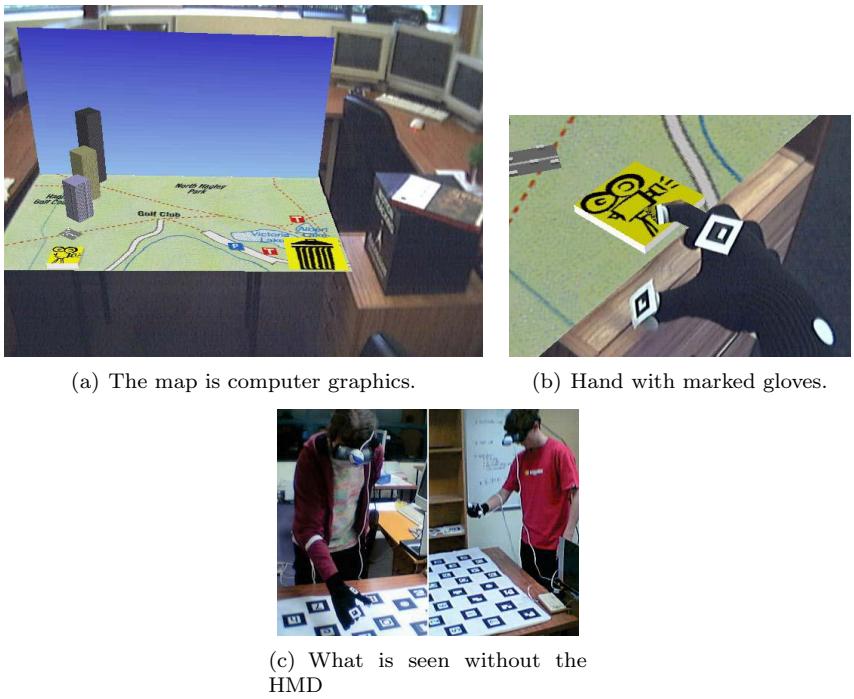


Figure 2.1: An organisation diagram for categorizing and classifying Gesture-based HCI research literature (from Karam [2006]).

approach and user interaction methodology for augmented reality (AR) on unprepared tabletop environments. In the same fashion Sato et al. [2004] proposes an augmented desk interface. Buchmann et al. [2004] present a fingertip-based interaction with virtual objects in Augmented Reality (AR) environments.



(a) The map is computer graphics.

(b) Hand with marked gloves.

(c) What is seen without the HMD

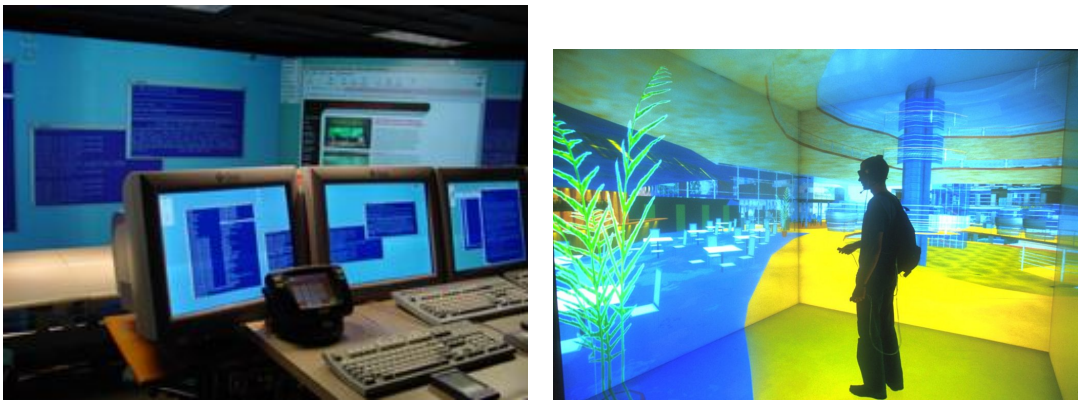
Figure 2.2: Augmented Reality Basic Setting in Buchmann et al. [2004].

Very close to the concepts present in AR, Virtual Reality (VR) (or Virtual Environments) aims at total immersion of the user in a synthesized world like the one provided by CAVEs (see Figure 2.3). Cabral et al. [2005] discuss usability issues of gestures and multimodal (gestures and speech) interaction in VR. Kaiser et al. [2003] proposes a system that deals with symbolic and statistical fusion, 3D gesture recognition, spoken language and referential agents information sources, and Chen et al. [2007b] presents a novel system for Navigating 3D virtual environments by hand gestures solely.

Another traditional area of application for gesture interaction systems is Human-Robot symbiosis or interaction (see Figure 2.4) with a vast number of published material (see [Al-Amin Bhuiyan, 2006], [Bhuiyan et al., 2007], [Brooks and Breazeal, 2006], [Hong et al., 2006], [Hu et al., 2003], [Lee, 2006], [Singh et al., 2005b],[Rao and Mahanta, 2006], [Ehreumann et al., 2001] and references therein). Some other applications can be included in the human-robot interaction category, for example, companion robots for the aging population [Menezes et al., 2006] [Chen et al., 2007a] [Chen et al., 2004a] and house robot-aids as the cooking robot proposed by Fukuda et al. [2004]. GPUIs have also been proposed for wheelchair control by Kang et al. [2003]

GPUIs have been proposed a number of times for operational board interfaces such as for video surveillance environments [Iannizzotto et al., 2005] and central traffic control stations [Chen et al., 2005]. Close to this is the application in board-games [Sriboonruang et al., 2006].

Commercial GPUIs products are not very common to date, however, some applications count on commercial systems, for instance, intelligent surfaces such as the Microsoft Surface technology (`http://www.microsoft.com/surface/index.html`) and Collaboration Walls by Perceptive Pixel (`http://www.perceptivepixel.com/`). There are, as well, a number of academic papers addressing these applications (see [Tse et al., 2006], [Chan et al., 2007] and references therein). Close to this is weather narration system with gesture automatic perception [Sharma



(a) DISCOVER Lab CAVE system (b) from www.intuition-eunetwork.net

Figure 2.3: CAVE Examples.

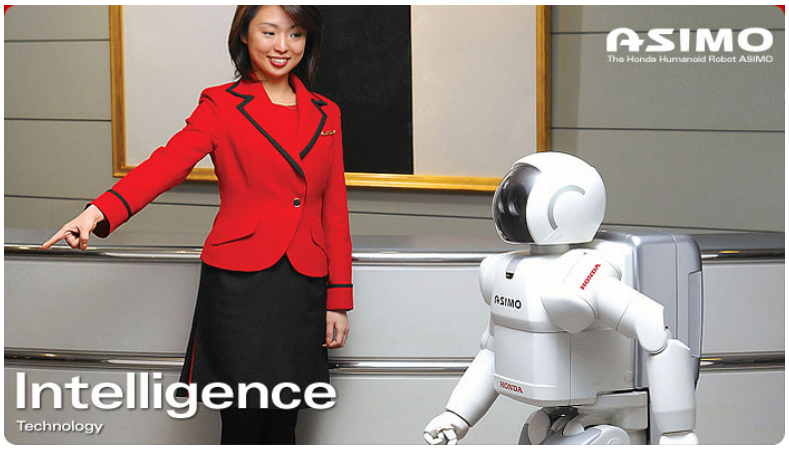


Figure 2.4: HONDA's famous ASIMO Robot understands simple hand gestures (`http://world.honda.com/ASIMO/technology/intelligence.html`).

et al., 2000].

Other promising application areas are unmanned vehicle control and aircraft marshalling as proposed by Singh et al. [2005a] and Choi et al. [2008]).

Je et al. [2007a] and Je et al. [2007b] proposes automatic gesture interpretation for analysis of music conducting action, Montero and Sucar [2006] for Video conference environments and Costanzo et al. [2003] for *smart* offices.

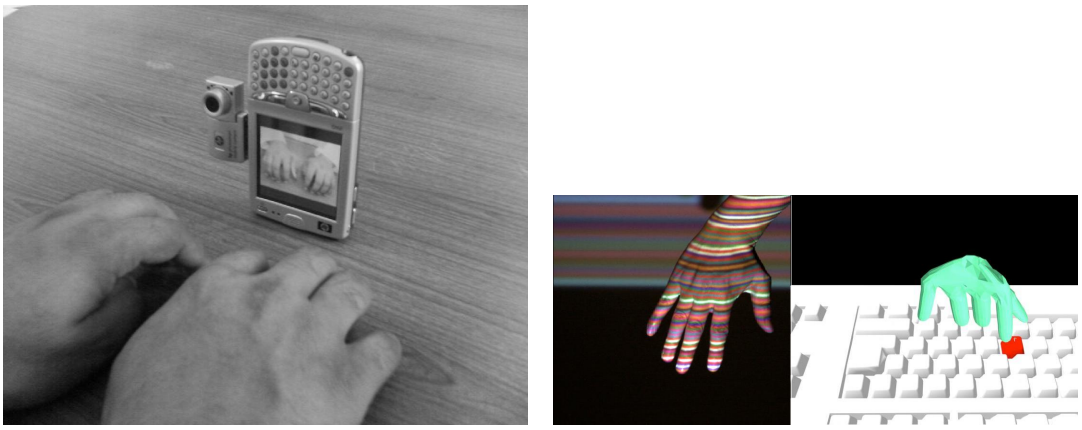
Computer-Assisted surgery (CAS) is a modern reality. In [Grange et al., 2004] the authors propose a multimodal interaction framework that lowers the cognitive load of the surgeon when dealing with operating room computers, while giving him more direct control over such equipment. Furthermore, in their work, is described a real-time surgeon detection and tracking, gesture recognition and activity monitoring. The authors note that:

- The operating room presents a controlled, well defined environment from a computer vision perspective, since illumination and color variations are not a significant problem;
- Modern CMOS cameras are small, lightweight, and easily movable, thus easily integrated into the operating room; and
- Visual gesture recognition does not require the surgeon to wear additional hardware (e.g. electromagnetic trackers).

One of the main themes in gesture recognition research, for example, is the application in sign language automatic translation. Sign language translation is a daunting challenge for vision based gesture recognition for its relatively large number of hand postures often with similar appearances -yet stark different meanings- as well as self-occlusion and other difficulties. Nonetheless, some important breakthroughs have been reported in this niche, see Liang and Ouhyoung [1998a], Travieso et al. [2003] , Henderson et al. [2005], Caridakis et al. [2008] and references therein.

Interestingly, sign language automatic translation to text has been already developed for mobile platforms as well [Elbouz et al., 2007] [Majoe et al., 2007] empowering its use.

Virtual keyboards using GPUI techniques (Figures 2.5(a) and 2.5(b)) have already been proposed for both desktop and mobile platforms [Du and Charbon, 2008] [Habib and Mufti, 2006].



(a) Mobile Phone System (from Habib and Mufti [2006])

(b) Desktop System (from Du and Charbon [2008])

Figure 2.5: Virtual Keyboard Examples.

Remarkable and perhaps hilarious was the development of a complex augmented reality and gesture interpretation system for human-chicken interaction by Lee et al. [2006].

2.2 Gesture types

There are different kinds of hand gestures. A first broad distinction we can make is between hand-only gestures and gestures that need other body parts to convey meaning (Figure 2.6). Note that the recognition of the later involve extra difficulties, since there will be the need to analyze scenes that are intrinsically more complex presenting occlusion, multiple regions with similar texture and color (skin regions), as well as complex correlation between these different regions.

(a) Shush sign: Meaning be silent.

(b) Cut Throat: Meaning someone is metaphorically dead

Figure 2.6: Gestures involving a combination of body parts.

Here we will deal solely with the simpler case of hand-only gestures, for our focus is on the robust extraction of information from video data as opposed to semantic analysis of complex contexts. In general, hand gestures are an effective way of communication which is deeply influenced by culture, the individual's emotional state, social environment and many other factors. There are innumerous facets of Human-Human gesture communication one can dedicate a study to. In fact, the study of gestures has proven to be a rich scientific discipline and a great number of technical publications about it is available (vide the International Society for Gesture Studies at www.gesturestudies.com).

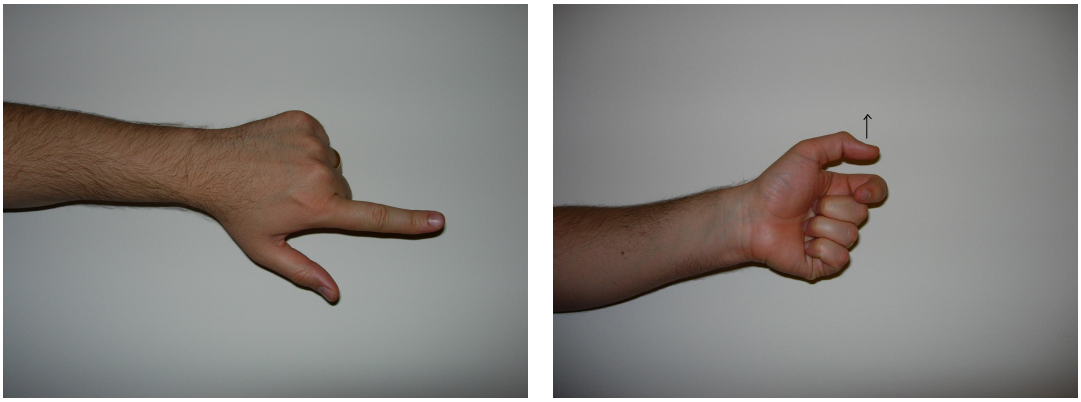
For our purposes, that is, from a HCI perspective, hand gestures can be categorized as communicative, manipulative or multipurpose. Communicative are those gestures that represent a well defined message or command. Sign language gestures, for instance, are clearly within the class of communicative gestures. The manipulative gestures have the intent to interact with an inanimate or virtual object, that is, to control, move, rotate, poke etc. Very common in HCI, multipurpose gestures are those that convey a command and manipulate an object simultaneously.

Another distinction of gesture types I make here is, perhaps, one of the most important when it comes to the actual implementation of a GPUI. Gestures can be divided into

i. **static** gestures which are defined by a single posture, that is, finger configuration (Figure 2.7(a));

ii. **dynamic** gestures that are characterized by the evolution of postures and/or poses (Figure 2.7(b)).

It is important to clarify that rigorous gesture ontology is more complex than the one depicted above and is often heavily influenced by the application context. For instance, a more detailed categorization of gesture types motivated by human-robot interaction was made by Nehaniv [2005], who divided gestures in five groups:



(a) Static gesture - pointing (b) Dynamic gesture - coin tossing

Figure 2.7: Examples of static and dynamic gestures accordingly with the convention adopted here.

Manipulative : manipulation of objects

Expressive behavior : associated with the affective state of the individual

Symbolic : communicative of semantic content (language like)

Interactional : used to initiate, maintain, regulate, synchronize, organize, or terminate various types of interactions

Referential : pointing of all kinds, referential, attention-directing.

Other examples of gesture typification are provided by Pavlovic et al. [1997] (Figure 2.8), Karam [2006], and Kölsch [2004]. The latter addresses specifically the topic of gesture and human comfort.

2.3 Gesture analysis and interpretation literature

Pavlovic et al. [1997] present not only a taxonomy for gestures types, but a comprehensive gesture modeling framework that includes a characterization of gestures' spatial properties that divides gestures interpretation in 3D hand model-based and appearance-based.

I use here a slightly modified version of that characterization in order to fit the systems developed in this thesis more clearly. Briefly, systems where it is attempted to infer postures and gestures *directly* from images or visual features (such as hand contours and fingertips positions) are said to use an appearance-based approach. Otherwise, systems that attempt first to estimate a *geometrical* representation of the hand configuration (by providing joint angles of a 3D model, for example) from every frame, and then interpret the resulting sequence of parameterizations are called model-based (3D model-based originally).

In the following chapters contributions to both appearance-based (chapter 5) and model-based (chapter 6) gesture interpretation will be proposed. The next sections present a brief characterization of the two approaches.

2.3.1 Appearance-based methods essentials

Appearance-based methods in the literature are composed basically of a feature extraction step (see Section 2.4) at every video frame, followed by a pattern classification module [Kurita and Hayamizu, 1998] [Gutta et al., 1996] [Iwai et al., 1999] [Lementec and Bajcsy, 2004]. That is, appearance-based methods are usually seen as an instance of the general problem of Pattern Recognition, which involves three clearly identifiable tasks, namely:

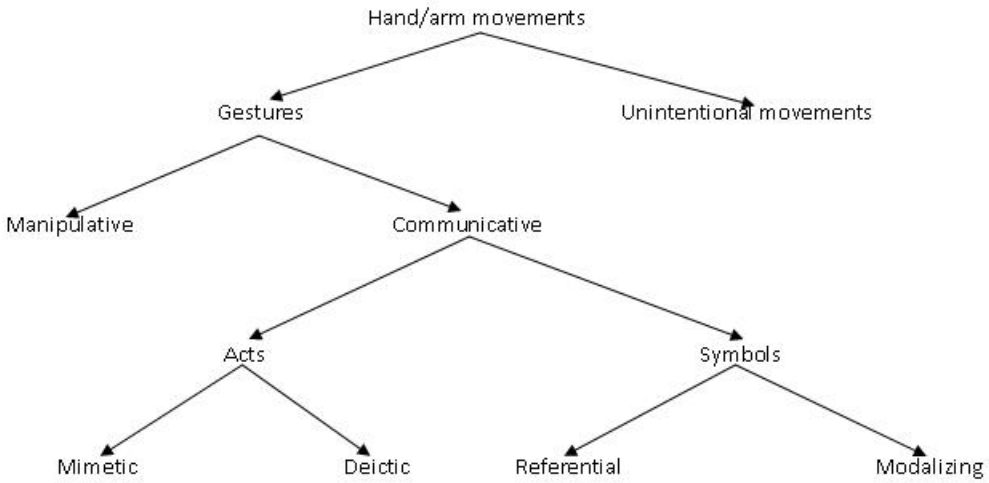


Figure 2.8: A Gesture Taxonomy (from [Pavlovic et al., 1997]).

i. Feature Extraction

ii. Classifier learning using training samples (known label)

iii. Classification of unseen samples

A, perhaps natural, assumption in the design of a gesture recognition system is that gestures can be decomposed as a characteristic sequence of hand postures and a trajectory as it is explicitly done by Bobick and Wilson [1997], Murakami and Taguchi [1991], Bulzacki et al. [2008], Joslin et al. [2005] and many others.

A good number of gesture recognition algorithms proposed in the literature fall in the scheme of the block diagram in Figure 2.9(a), as is the case of Wilson and Bobick [1999], Travieso et al. [2003] and Chen et al. [2003], for instance. These works are characterized by the presence of a module for *feature extraction* that outputs a feature vector that compactly describes the hand and/or hand dynamics for a single frame *buffer* module that represents the procedure of storing feature vectors associated with an interval $(k - N, k]$ of N frames that contain the execution of a single gesture; and a module for the *classification* of the set of N buffered feature vectors.

I adopted an architecture somewhat different from the above in which, instead of classifying sets of feature vectors at a time, it is assumed that only a known *finite* set of hand postures would be allowed, thus, *posture* classification is done every frame resulting in a posture code. A certain number of consecutive codes are merged into a single vector, and this *gesture-feature-vector* can, then, be classified resulting finally in a gesture code (Figure 2.9(b)). This scheme has been adopted by various other researchers as well [Liang and Ouhyoung, 1998a] [Ng and Ranganath, 2000] [Chen, 2008] [Habib and Mufti, 2006].

2.3.2 Model-based methods essentials

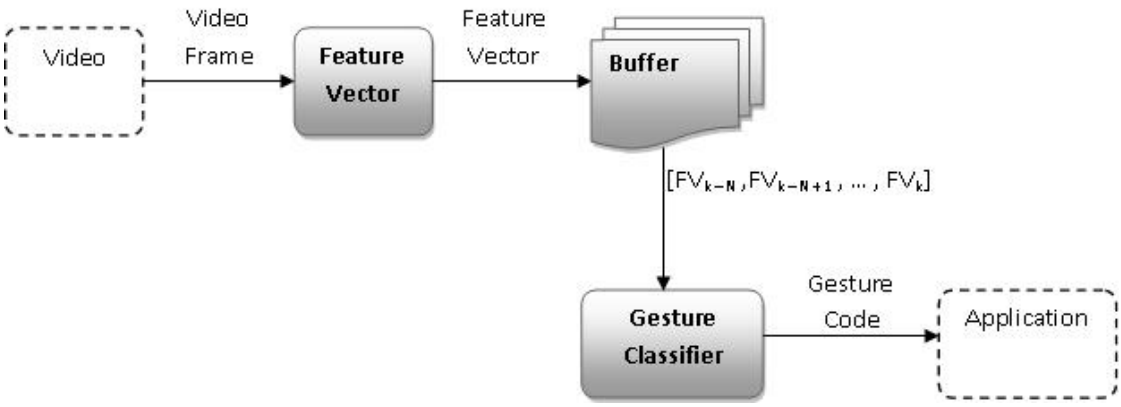
In this class of methods for hand gesture interpretation, fall those methods that rely fundamentally on the sequential fitting of a geometrical hand model to the hand in the image. This translates to sequentially estimating (tracking) the model parameters. In many cases, model-based systems will not make any use of machine learning procedures. An emblematic example of non-recognition GPUI is given by Tosas and Li [2004], where an augmented reality virtual touch screen (VTS) is displayed using a head-mounted display and the VTS is made touch-sensitive through unadorned (markeless) visual articulated hand tracking. A similar non-recognition GPUI example was proposed earlier by MacCormick and Isard [2000].

Model-based methods can use 3D or 2D hand models (Figure 2.10). For each of those there are different dominant approaches. In 3D hand tracking the immediate goal is usually to render a 3D model of the hand with a computer graphics program.

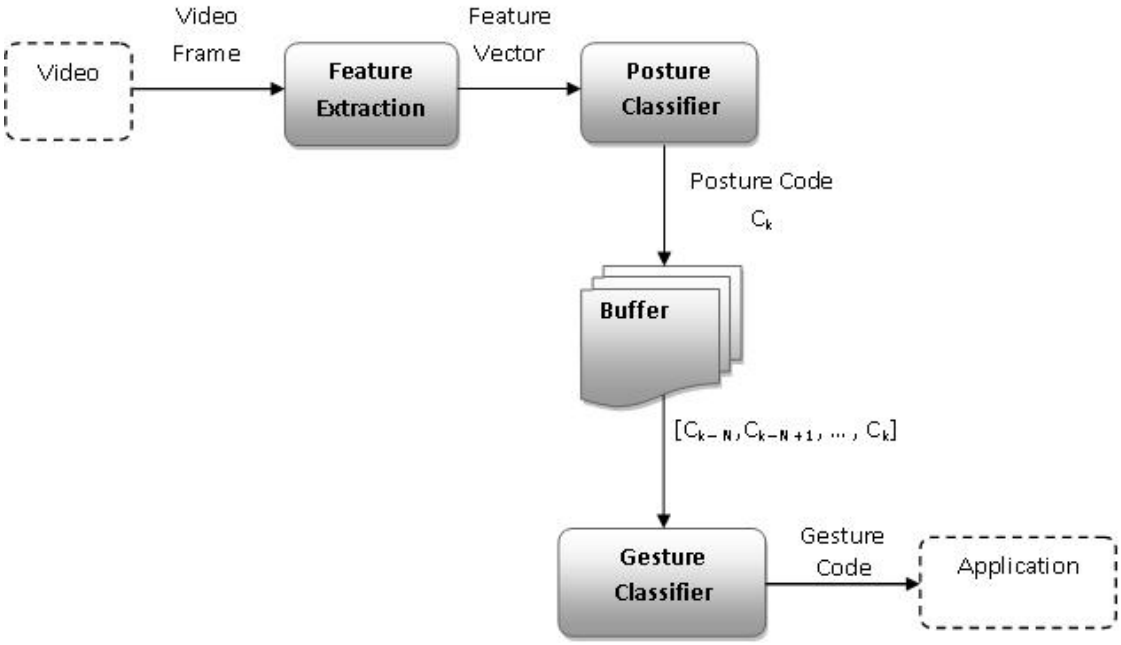
A strong trend in 3D hand tracking in the literature is to formulate the problem as a matter of database retrieval. Tomasi et al. [2003] inaugurated this methodology. Their investigation was based on the assumption that what happens too quickly we just cannot see. They built a 3D hand tracker for these types of motions that relies on the recognition of familiar configurations in 2D images (classification - for *database retrieval*), and fills the gaps in-between (interpolation).

Malric [2008] built a tracker based on the recognition of familiar configurations in 2D images, as well, using a bank of neural networks working in a voting system. Athitsos and collaborators [Athitsos and Sclaroff, 2003] [Athitsos and Sclaroff, 2003] [Athitsos et al., 2004] [Athitsos, 2006] worked in the same line, however, they addressed the use of direct matching using specialized metrics for database retrieval instead of traditional machine learning algorithms, that is, given the image measurement (a feature vector) how to compare it with every element of the database in order to find the most similar instance in a reasonable time.

El-Sawah et al. [2007] introduces and evaluates an integral framework for 3D hand tracking and dynamic gesture recognition. They present the 3D vision-based hand tracking and posture estimation as a nonlinear optimization problem, where the search space is the high dimension

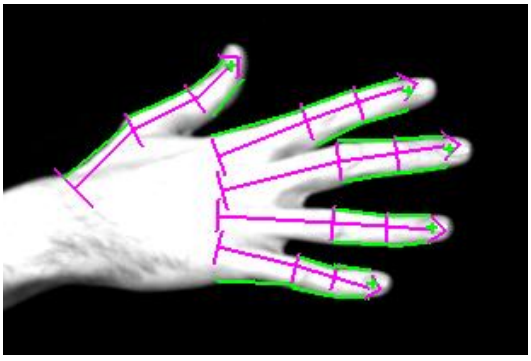


(a) A Generic Scheme for Gesture Recognition



(b) Gesture Recognition through Posture Classification Scheme

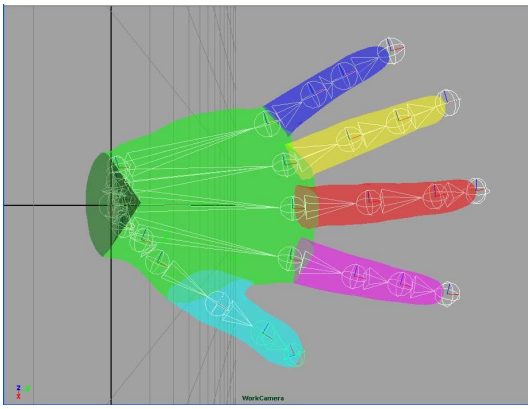
Figure 2.9: Basic Gesture Recognition Schemes.



(a) A 2D articulated segments hand model (from Rehg and Kanade [1994]).

(b) The cardboard 2D hand model (from Wu et al. [2005])

(c) 2D model made of ellipsoidal patches (from Bretzner et al. [2002])



(d) A 3D articulated model (from Malric [2008])

Figure 2.10: Examples of 2D and 3D geometrical hand models.

hand articulation (finger joints) and the objective function is a probabilistic observation model derived from generic features in the acquired video.

Any general 3D articulated-object tracking algorithm can, in theory, be used for hand tracking. Some important contributions in this topic were done by Wu et al. [2003], Rehg and Kanade [1994] and Plankers and Fua [2003]. A comprehensive review on the subject of articulated 3D hand motion estimation is provided by Erol et al. [2005].

The tracking of the twenty seven parameters of a 3D model from bare hand 2D images is an ill-posed problem in the sense that, from certain perspectives, different postures will look exactly the same. That happens because some parts of the hand may get occluded by others, therefore, making the configuration of the hidden parts no more than guessable, in a situation said to be of self-occlusion. Inter-frame correlation in a video sequence and the fact that some postures are much more frequent than others can be exploited however, to lessen the effect of self-occlusion-generated ambiguity in the extraction 3D articulated-model parameters from monocular video.

Another solution would be the use of special hardware, that is, cameras based on infra-red technology able to capture depth information. This is however a more costly path.

Although, from 3D articulated-models one can -in theory- recognize any gesture the hand is able to perform, in many applications, one only needs to recognize a small set of predefined hand gestures. From a 2D articulated hand model a large set of gestures can be recognized, at least large enough for most human computer interaction aims.

The development made in this thesis is all for 2D models. The model variability perveived within the 2D category in the literature is certainly greater than the variability perceived in the 3D category. For example, Wu et al. [2005] use the *cardboard model*, in which each finger is represented by a set of three connected planar patches (Figure 2.10(b)); Bretzner et al. [2002] use a representation of the hand based on circles and ellipses to cover the hand parts (Figure 2.10(c)). In fact, the choice of the model in 2D hand tracking systems is not the most important question, what really matters is how to track the hand, or how to find the model parameters that sequentially fit the model (template) to the hand in the image.

In terms of tracking procedure, the most widely adopted line of research for 2D models was the one initiated by the remarkable work of Blake and collaborators [Blake et al., 1993] [Blake et al., 1994] [Blake et al., 1995] [Isard and Blake, 1998a] [Isard and Blake, 1998b], done for rigid object tracking in the beginning and then extended to articulated objects [Blake and Isard, 1998] [Deutscher et al., 2000] [MacCormick and Isard, 2000] [Tosas and Li, 2004]. The methodology in this line is based on a *state-space* formulation, and Sequential Monte Carlo algorithms are applied for filtering (tracking) the model parameters.

Due to the importance of *tracking* algorithms to my work, an entire chapter is dedicated to a detailed analysis of the topic (Chapter 6).

2.4 Feature extraction

Feature extraction is the processing and analysis of lower-level information (pixel values) needed to produce higher-level information (objects contours, for instance). Conceptually, feature extraction is a way of reducing data dimensionality by encoding relevant information in a compressed representation, pruning away less discriminative information. Evidently, how relevant or discriminative information is varies with the object of interest and the goals of the application.

The word *Pixel* comes from the merge of *picture* and *element*. Pixels of gray-level images are scalar values representing the brightness intensity, and pixels of color images are usually represented by a three-component vector, called r,g and b, respectively. These components define the level of brightness of red, green and blue for that pixel, therefore, defining the color and brightness a human's eyes will perceive in that picture element. Computer representation of digital images consists of 2D matrices of pixels, therefore, color images are accessed as three

matrices of similar dimensions. Usually, when a reference is made to a pixel, one might be referring to its *value* or the (i, j) position - or *point*- in the matrix.

For a complex task such as gesture interpretation, it is very convenient to divide the feature extraction operation in levels. For example, in my work there are three levels. The lower level features are the points with high probability of being on the object, opposed to the points on the background. From these features, middle level features are extracted, which, in this, case are hand contours. Then, from contours, different types of high level features are investigated here: shape features for the appearance-based approach; and model parameters for the model-based approach. From high-level features, postures and poses are determined which can be utilized by the user interface application or be stored so as to be analyzed in sets for dynamic gesture recognition. Throughout this text the feature level will not be explicitly mentioned most of the time, and it will depend on the context or it will mean feature as an abstract concept. Nevertheless, middle level features (contours) extraction will be refered as *contour tracking* hereafter. In this literature review section I will focus on low-level feature extraction.

Feature extraction is paramount to gesture recognition performance, thus, the choice of which features to work with as well as the extraction algorithm are likely to be the most important design decisions in GPUI development. What is perceived from the literature is that the way feature extraction is done in each work depends on the authors' understanding of what characterizes the object's appearance accurately enough for the underlying application context.

The authors' understanding of the object appearance can be synthesized by an appearance model. An appearance model has to carry information about the object (the hand in our case) sufficient to differentiate the image regions that belong to the object from those that belong to the background, and/or enough to infer the status (posture and position, orientation and scale).

Although this thesis focus on the specific problem of *contour* tracking, an excellent source of bibliography in *Feature Design* can be found in the *object* tracking literature. Note that tracking contours is a much more challenging problem than object (position) tracking; nevertheless, both problems have many similar aspects. Color has been, probably, the property most utilized for feature extraction [Manjunath et al., 2001] [Sigal et al., 2004] [Yao and Zhu, 2004]. Nonetheless, texture, luminosity, and motion -defined by two or more frames- have been studied as well [Wu et al., 2000] [Blake and Isard, 1998] [Cutler and Turk, 1998].

Few feature extraction frameworks persist throughout the computer vision literature. There are, however, honorable exceptions of very well designed feature extraction technology that have been recurrently adopted, of which some will be briefly introduced in this review. In the following subsections some well known feature types and their computation methods will be discussed.

2.4.1 Haar-like features

The object detection algorithm developed by Viola and Jones [2001b] (see also Viola et al. [2005]) was certainly one of the greatest successes in computer vision technology to date. It manages to be extremely fast and reliable for many objects in many adverse situations. The algorithm's success is due in part to the innovative features and features computation procedure they developed.

Viola and Jones base their work on a novel image representation called *Integral Images* which allows the features used by their detector to be computed very quickly. The integral image at location (x, y) contains the sum of the pixels values above and to the left of (x, y):

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x, y) \tag{2.1}$$

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image (see Figure 2.11). Using the

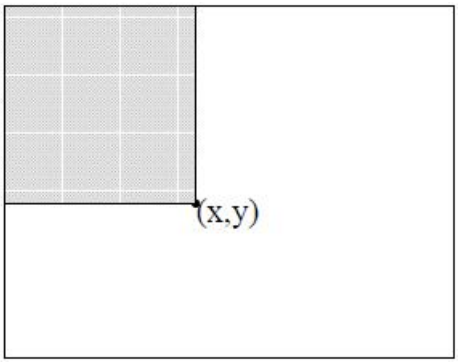


Figure 2.11: The value of the integral image at point (x, y) is the sum of all the pixels above and to the left (from Viola and Jones [2001b]).

following pair of recurrences:

$$s(x, y) = s(x, y \ldots 1) + i(x, y) \tag{2.2}$$

$$ii(x, y) = ii(x \ldots 1, y) + s(x, y) \tag{2.3}$$

where $s(x, y)$ is the cumulative row sum, $s(x, \ldots 1) = 0$, and $ii(\ldots 1, y) = 0$. The integral image can be computed in one pass over the original image.

The features used in the Viola&Jones algorithm are called *Haar-like* features or rectangular Haar-like features. They are called that way for the similarities they share with the Haar *wavelets* basis functions [Chui, 1992]. In general, Haar-like features highlight simple patterns of color and luminosity variations in the image reducing the influence of noise.

These features are computed from the image pixels by scanning the image with specific rectangle patches that are divided in white and black subrectangles (Figure 2.12(a)). The feature at every point is computed by subtracting the weighted sum of the pixels in the white areas from the weighted sum of the pixels in the dark areas.

Using the Integral Image representation, the computation of that difference of sums resembles a simple table look-up operation. Only four array references to the Integral Image are needed to compute any of the needed sums.

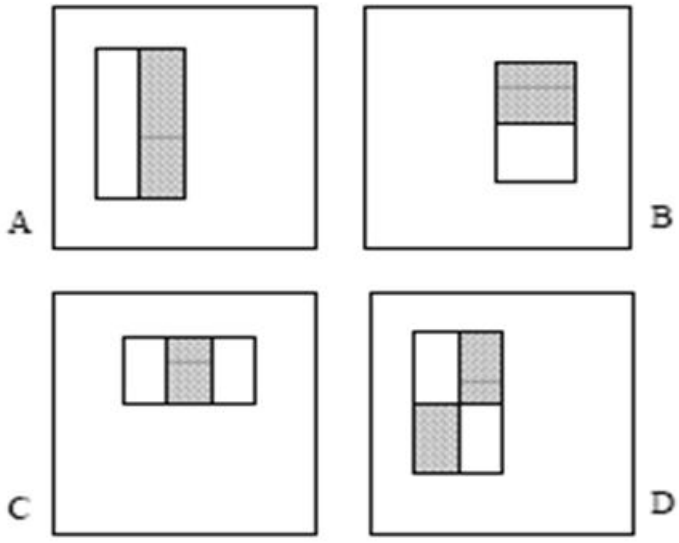
Lienhart and Maydt [2002] extended the original set of rectangular features to include tilted Haar-like features (Figure 2.12(b)), which are as fast to compute as the original ones, but help to concretely improve performance. Recently, Messom and Barczak [2006] generalized that result using the concept of rotated Integral Images.

2.4.2 Optical flow

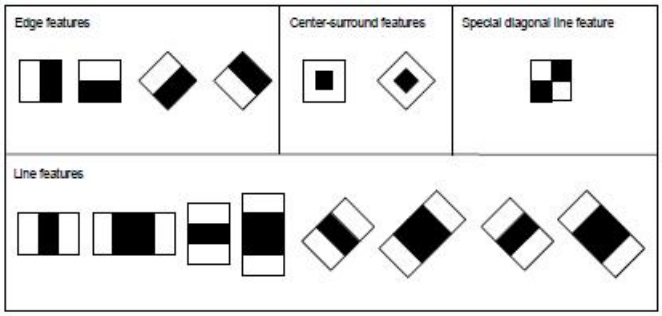
The study of the estimation of the apparent motion of image pixels is generally called *optical flow*. The basic assumption is that the appearance of objects does not change (at least significantly) between consecutive frames, therefore, there is *conservation* of pixel intensities for regions of that object wherever the object is placed in the image. This problem can be formulated as a set of differential equations known as the *aperture problem* of optical flow, which is insovable as given. Solutions to this problem necessarily add further restrictive assumptions.

In the historical paper Lucas and Kanade [1981], it was assumed that motion between consecutive frames is approximately linear (or affine) and proposed a solution to the apperture problem based on this assumption. Their algorithm is still one of the most used today.

In the same year, Horn and Schunck [1981] launched a second line of development assuming that movement throughout the image would probably be *smooth*. This smoothness was expressed in his work by rewriting the optical flow problem in the form of a functional minimization problem



(a) Original Haar-like features used by Viola and Jones [2001a].



(b) The extended set of Haar-like features (from Chen [2008]).

Figure 2.12: Haar-like Features Representation.

and adding a smoothing (or regularization) term to it. Following this work there were a series of proposed modifications of the regularizing term as well as the addition of other terms (see Hinterberger et al. [2002]). Other different approaches to optical flow estimation were proposed, I suggest Beauchemin and Barrón [1995] for an overview.

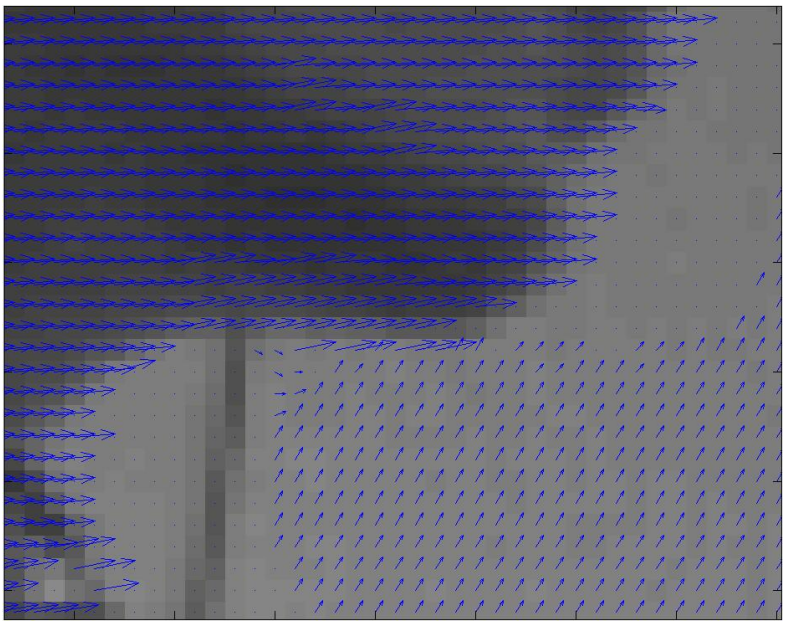


Figure 2.13: Usual visualization optical flow (from `http://www.cs.umd.edu/~sontran/733/small_flow.png`).

Optical flow computation results in a field of velocities (or displacements) which are usually represented by an arrow field like the one in Figure 2.13.

2.4.3 Edge features

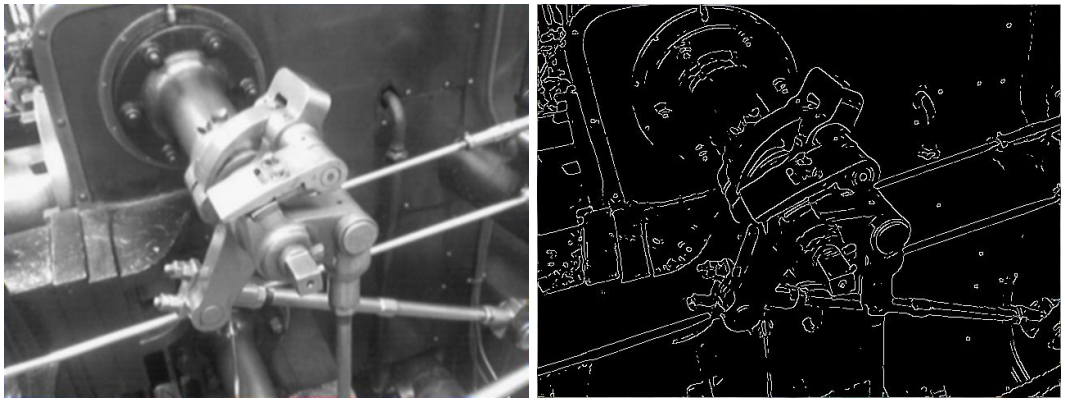
Edge features are meant to highlight an image's regions of accentuated variation of luminosity, color, or texture. Local maxima of these properties will probably coincide with object's boundary points. Traditionally, edge features are computed through the convolution of the image with a spatial gradient operator (Figure 2.14) or using higher level algorithms like the famous Canny's edge detector [Canny, 1986] or the boundary tensor [Köthe, 2006].

In computer vision literature, edge feature computation is called edge detection. That is probably because the ideal result from an edge computation operation would be a binary image of the same dimensions of the original image, however, only edge points would be different than zero. It is also the case, that edge highlighting and edge point selection are attempted to be done simultaneously or in a non-distinguishable fashion.

2.4.4 Statistical features

As it is the case for my work, most of the time you have some concrete prior knowledge of the appearance of the object. For example, in a GPUI system, the user is asked to place the hand somewhere in the scene, where a sample of the hand color or texture may be acquired.

I call statistical features the type of features that are computed by scanning the image and building a map that gives at each point a measure of the likelihood of that point being part of the object. In this sense *blob* tracking [Carson et al., 2002] techniques are extraction methods for this type of feature.



(a) Engine-original image. (b) Engine-Canny edge detector output

Figure 2.14: Canny edge detector output example.

2.4.5 Feature selection algorithms

The most simple selection algorithm is thresholding. A threshold value is set, usually, in an empirical or intuitive way, such that a binary decision is made by direct comparison. That is, given the feature value (or vector) x_p associated with point p in the image and the threshold value τ, the following is an example of a thresholding rule:

$$\begin{cases} \|x\| < \tau, & p \in \Omega \\ \text{otherwise}, & p \notin \Omega \end{cases} \tag{2.4}$$

where Ω can be the object contours, the object internal region, etc.

Useful object features can however, be extracted from optical flow. As noticed by Shi and Tomasi [1994], although tracking the center of mass of the object is by and large a solved problem selecting features that can be tracked well and correspond to physical points in the world is still hard. Shi and Tomasi [1994] present an optimal selection criterion and a feature tracker based on it, and Tommasini et al. [1998] builds upon their work. Furthermore, developments in the area of *interest point detection* (equivalent to features selection), such as SIFT [Lowe, 1999] and SURF [Bay et al., 2006], can be used in conjunction with optical techniques for object motion analysis and gesture recognition [Ke and Sukthankar, 2004] [Zhou et al., 2007]

Cutler and Turk [1998] approach the problem of analysis of optical flow for body-gesture recognition. Kölsch and Turk [2005a] shows the results of hand tracking with *Flocks of Features* [Kölsch and Turk, 2005b], a tracking method that combines motion cues (a.k.a. optical flow) and a learned foreground color distribution to achieve fast and robust 2D tracking of highly articulated objects (Figure 2.15), moreover, HandVu a computer vision system that recognizes hand gestures in real-time is proposed using that hand tracker. Shah [1995] provides a comprehensive survey on motion-based recognition in general.

The model-based hand tracking algorithms presented in later chapters use a feature extraction procedure evolved from the technique explained in detail in Blake and Isard [1998]. The technique takes advantage of the fact that 1D line processing is usually much faster than 2D region processing. The main idea is to draw a number of line segments crossing the current estimated object contour and perpendicular to it. Then iterate through each line segment searching the feature points usually by thresholding the output of an edge detector applied to the points of the line (Figure 2.16). By this method, the search space is reduced to a region close to the last object's known position, in fact, to line segments in that region; therefore, the amount of computation needed tends to be extremely small.

Naturally, most of the papers developed on the condensation framework [Isard and

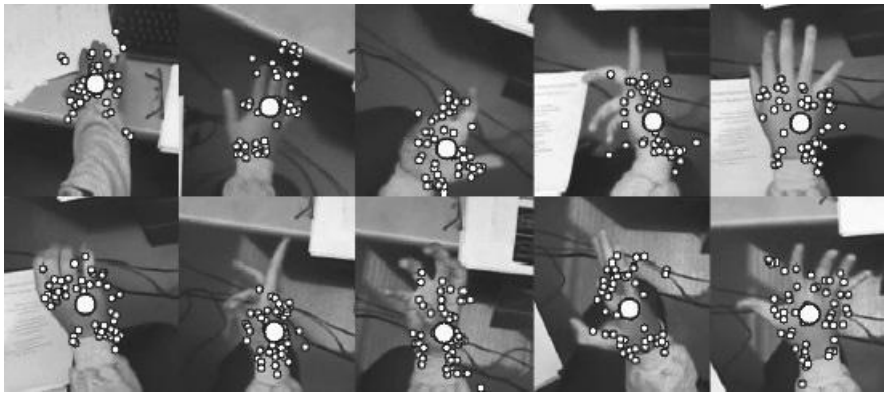


Figure 2.15: Hand tracking based on optical flow (from Kölsch and Turk [2005a]).

Blake, 1998a], that aimed at tracking contours instead of boxes, ended up using normal-lines search scheme for feature extraction as well. This was true in particular for those papers that worked on an extension of the CONDENSATION for articulated objects [MacCormick and Isard, 2000] [Tosas and Bai, 2007] [Zhou et al., 2008] .

Nascimento and Marques [2004] also uses linear search of contour normals, however, it proposes a mechanism for reducing the damage generated by outliers through the elimination of features points that don't fit in natural groupings they called *strokes*.

Principal components analysis (PCA) is a general technique for dimensionality reduction. The use of PCA as a feature extraction framework in computer vision has proven to be very versatile, vide Vidal et al. [2005] and references therein for examples.

2.4.6 Extraction based on machine learning

The approach I adopted for feature selection is still very little explored in the context of tracking, yet very promising results have already been published. The underlying idea is to use the power and flexibility of machine learning algorithms in order to model object appearance and approach the problem of feature extraction as a pattern classification procedure.

Feature selection using machine learning algorithms is not new, however, researchers usually avoid this tactic assuming it too time-consuming for most practical applications. The most common way of doing things is using simpler methods for feature selection, leaving machine learning algorithms for the classification of higher level patterns only, such as shapes, sequences of shapes, etc.

Viola&Jones object detection algorithm is a sheer proof that today's available computing power is more than enough for the use of machine learning algorithms in low-level features extraction in real-time systems.

The straightforward form of object detection is to analyze every image sub-window (with superposition) of a certain size and decide if there is an instance of the object inside that window or not. Considering that there is a certain range of distances the object is allowed to be from the camera, there will be as well a range of window scales allowed. The point here is that object detection, in its basic form, demands a huge number of classification operations, even for small images. For example, given a base resolution of 20x20 of the detector, the exhaustive set of rectangle features is 160,000, much bigger than the number of pixels.

Although some research has been already done in tracking through machine learning, I could not find works that attempted to track *contours* specifically. That is, the trackers using general classifiers for feature selection aim at the tracking of a box containing the object. For example, Avidan [2004] elegantly integrated a Support Vector Machine (SVM) into an optic-flow-based tracker. Avidan's tracker worked by maximizing a SVM score instead of an intensity

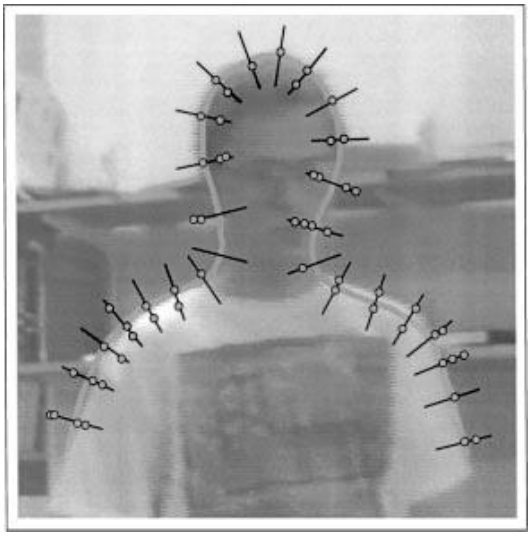


Figure 2.16: Feature extraction based on linear search on contour normals (from Isard and Blake [1998a]).

difference function.

Shen et al. [2006] specifically integrated a general classifier (SVM) into a model-based CONDENSATION tracker. They reasoned that the performance of a Bayesian tracker such as the CONDENSATION (or particle filter) is intrinsically dependent on the likelihood function, that is, the function that measures how well a hypothesis fits the image data.

Grabner et al. [2006] developed an improved version of Avidan's work that uses *Adaptive Boosting* to learn variations of the object appearance online. Williams et al. [2003] and Williams et al. [2005] are also studying ways of improving Avidan's tracker, however, they focus on exploring temporal correlation in a principled fashion to boost speed and robustness.

There are segmentation algorithms that could be adapted to contour tracking but are extremely slow [Hosaka et al., 2007]. Kotropoulos and Pitas [2002], for instance, proposed the segmentation of ultrasonic images using SVMs. In their paper it was proposed the use of SVMs to segment lesions in ultrasound images and their lesion detection ability was thoroughly assessed.

2.5 Machine learning theory

One of the main contributions of my work is the innovative way in which machine learning algorithms are integrated into the contour tracking procedure. Machine learning is also used for posture and gesture classification. Thus, this entire section is dedicated to discuss literature on machine learning theory as well as on the application of machine learning algorithms to feature extraction and vision-based gesture recognition.

Machine learning algorithms have been applied successfully to credit card fraud detection, internet search engines, medical diagnoses, bioinformatics, handwritten recognition, speech recognition, stock market analysis, classification of DNA sequences, to solve many computer vision problems, and others. This just shows how powerful and general machine learning theory is.

The basic idea of machine (inductive) learning is to make computers able to learn patterns from available data, so as to be capable to *recognize* these patterns later on.

For an all-embracing introduction to the main topics in machine learning for pattern classification I suggest the popular book by Duda et al. [1973], specially for its accessible language. For a theoretical background of the modern statistical learning theory, including the mathematical tools for formally assessing learning machines performance and the concepts of learning capacity, VC-dimension, structural risk minimization and many others I strongly suggest starting with the tutorial by Burges [1998] or the small book of Vapnik [2000], in order to get a bird's eye view first, before jumping into Vapnik's bigger book [Vapnik, 1998].

The theory of Perceptron learning and Artificial Neural Networks (ANN) [Anderson, 1995] provided the first successful machine learning paradigm, which still one of the most used today. Two of the main advantages of ANNs are the existence of well tested implementation and relatively fast classification time (as opposed to training time). ANNs have been applied to gesture recognition recurrently in the last decades [Murakami and Taguchi, 1991] [Kim et al., 1996] [Liang and Ouhyoung, 1998b] [Yuan et al., 2005] [Malric et al., 2008].

Support vector machines are the most notorious product of the modern *statistical learning theory* initiated by Vapnik and Chervonenkis [Vapnik, 1998]. An outstanding advantage of SVMs over ANNs is that, except from a few parameters to set, there are no worries related with the definition of neuron topology as it is the case with ANNs. The SVM learning process will find an optimal number of support vectors, in terms of structural risk. SVM has been evaluated in late years, and in many applications, it has outperformed ANNs; SVM will often generalize better.

Hidden Markov Models (HMM) is perhaps the most popular machine learning tool in gesture recognition literature [Kurita and Hayamizu, 1998] [Wilson and Bobick, 1999] [Iwai et al., 1999] [Stenger et al., 2001] [Fei, 2004] [Rajko et al., 2007]. HMM offers concrete advantages compared to other classifiers when it comes to dynamic gesture recognition (in contrast to posture

recognition), that is because, temporal gesture segmentation and recognition take place simultaneously, and HMMs can be trained by a number of training samples similar to Neural Networks [Eickeler et al., 1998]. The first popular application of HMM technology was speech recognition and the best introductory literature for the topic was produced with that application in mind [Rabiner and Juang, 1986] [Rabiner, 1989] [Juang and Rabiner, 1991].

Other machine learning methods have also been used in hand tracking and gesture recognition: decision trees [Gutta et al., 1996] [Hang and Qiuqi, 2004] (Amit and Geman [1997] developed a work on shape classification using decision trees that can be applied directly to posture classification in a contour tracking framework), syntactic grammars [Chen, 2008], Bayesian networks [Hua and Wu, 2004] [Kettebekov et al., 2005] [El-Sawah, 2008], AdaBoost [Ong and Bowden, 2004], and Fuzzy models [Su, 2000] [Rao and Mahanta, 2006] [Habib and Mufti, 2006]. There is however lack of papers making comparisons between classifiers in terms of vision-based gesture interpretation.

2.6 Bayesian tracking

As it was mentioned earlier, this thesis addresses two modalities of hand tracking. The first one is the base for the model-based approach to gesture interpretation. As it was mentioned in previous sections, model-based hand tracking is the sequential estimation of a set of model parameters. This set of model parameters define at every time step k the **state** or status of the model x_k. This section presents some relevant literature on recursive state estimation which is traditionally developed in a Bayesian framework.

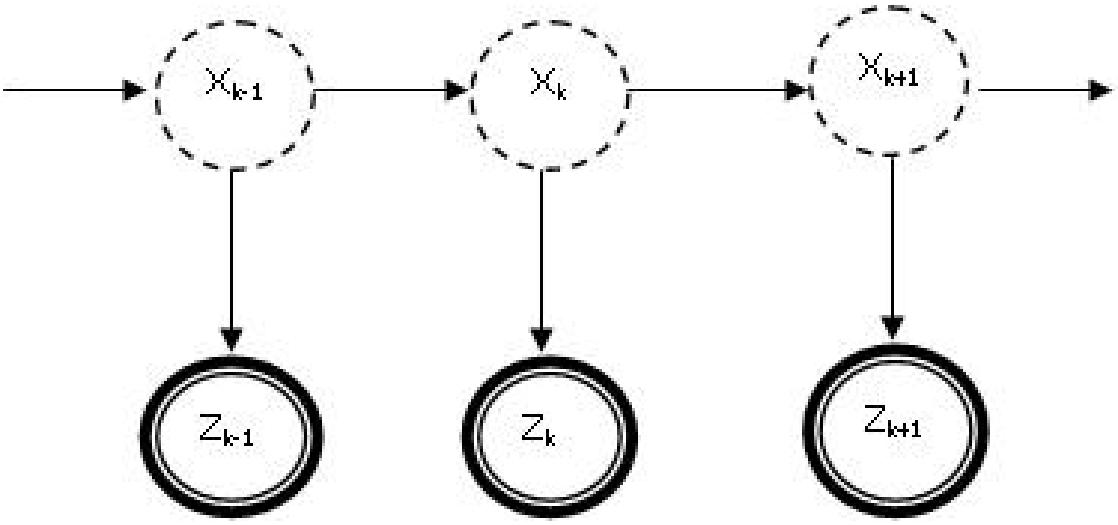


Figure 2.17: In the state-estimation problem the true states are hidden only accessible through the noise observation process.

The basic assumption is that the state x can be considered a Markov process, that is:

$$p(x_k|x_{k-1}, xk-2, \ldots, x_1) = p(x_k|x_{k-1}) \tag{2.5}$$

which means that the probability distribution of the state at step k given the immediately previous one is conditionally independent of the other earlier states. This assumption might seem a strong restriction at first sight, but it is not, in fact, this model is suitable for a large number of real-life processes. It is also part of the Markovian assumption that the initial probability distribution of x is also known.

Moreover the states are not observed directly. Instead, a measurement process z is acces-

sible (Figure **??**), which is modeled as a function of the state distorted by noise:

$$z_k = h_k(x_k, \eta_k) \tag{2.6}$$

where η_k is a white noise process that accounts for *measurement noise*.

The Bayesian framework integrates prior knowledge of the state process dynamics in a principled way through a dynamic model:

$$x_k = f_k(x_{k-1}, \omega_k) \tag{2.7}$$

where ω_k is white noise, the *process noise*, which accounts for intrinsic uncertainty or randomness of the tracked process.

The optimal state estimate $\hat{x}_k$ is determined by the probability distribution $p(x_k|z_k)$, which is computed in a two-stage procedure. The first stage is the *prediction* stage, in which the dynamic model is used to produce a predictive distribution of the state in the current time step, based on the previous state x_{k-1} and the knowledge of the conditional distribution $p(x_{k-1}|Z_{k-1})$, through the known Chapman-Kolmogorov equation:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1}dx_{k-1} \tag{2.8}$$

where $p(x_k|x_{k-1})$ is determined by (2.7) and the known statistics of ω.

In the second stage the prediction produced by (2.8) is updated with the information introduced by the data measurement acquired at the current time step z_k through the Bayes' rule:

$$p(x_k|z_k) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_1 : k-1)} \tag{2.9}$$

where

$$p(z_k|z_1 : k-1) = \int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k \tag{2.10}$$

The term $p(z_k|x_k)$ is called the likelihood function and is defined by (2.6) and the known statistics of η.The implementation of equations (2.8) and (2.9), a.k.a. the recursive propagation of the *posterior* density is not trivial. In general it cannot be determined analytically.

2.6.1 The Kalman filter (KF)

Kalman [1960] derived, in the discrete case, the analytical solution for the simplified, but very important situation where the dynamic and observation models are linear, that is,

$$x_k = Fx_{k-1} + B\omega_k \tag{2.11}$$

$$z_k = Hx_k + C\eta_k \tag{2.12}$$

where F, H, B, and C are matrices, and the state has a Gaussian initial distribution. It can be proved that the state distribution will remain Gaussian throughout time, in the above conditions. Gaussian distribution is compactly represented by a mean vector and a covariance matrix. The Kalman Filter is with no doubt the most popular state estimator of all times, due to its easy implementability and robustness.

In case the dynamic and observation models are not both linear but still differentiable, an extension of the Kalman filter (EKF) can be easily derived by means of the Jacobian matrix computed in terms of the current predicted state at each time step. The Jacobian is used to build linearization of the nonlinear functions through Taylor series expansion. Note that the linearized versions of the models are only needed for the propagation of the posterior covariance matrix and the mean is propagated non-linearly without problems.

Julier and Uhlmann [1997] reasoned that the linearization provided by EKF results in poor performance when the dynamic or the observation model is highly non-linear. Julier and Uhlmann proposed a modified Kalman filter for non-linear models that avoid the burden of computing Jacobian matrices, the Unscented Kalman Filter (UKF). The UKF stands on the principle that it is easier to approximate a probability distribution than a general non-linear function. It uses a set of discretely sampled points (sigma points) used to parameterize the mean and covariance. The discrete representation can be propagated nonlinearly and then used to approximate the (Gaussian) posterior distribution.

2.6.2 The particle filter (PF)

The particle filter breaks with the restriction of Gaussian distributions. The basic idea is to work with a set of weighted samples (particles) $\left\{x^i_{0:k}, w^i_k\right\}_{i=1,\ldots,N}$ that is propagated and used to estimate the posterior in the following way

$$p(x_k|z_k) = \sum_{i=1}^{N} w^i_k \delta(x_{0:k} - x^i_{0:k}) \tag{2.13}$$

where N is the number of particles, w^i_k the particle weights. The direct sampling of the posterior, however, is not viable most of the times and the way around this problem is given by the theory of Sequential Monte Carlo methods, in particular, the sequential importance sampling method. In fact, Isard and Blake [1998a] main contribution was to successfully implement that method for contour tracking. In that work, the method was called CONDENSATION as a reduction of CONditional DENSity propagATION. There is a wealthy of literature on the subject of sequential Monte Carlo methods (particle filters). An excellent start is the tutorial of Arulampalam et al. [2002] or the overview paper of Doucet et al. [2000a]. A more extensive overview is given by Doucet et al. [2001a].

2.6.3 Markovian jumps linear systems (MJLS)

In this thesis it is argued that there exists a class of dynamic hand gestures that is characterized by the succession of characteristic movements of the hand, where each of these movements can be fairly emulated by a linear model such as (2.11). This type of system is said to make Markovian Jumps because the rate of change (jump) between models is governed by a Markov chain. MJLS models are, in spite of the *linear* in the title, a kind of non-linear system. It is well known that optimal non-linear solutions are based on a bank of Kalman filters that grows exponentially with time, proving to be impractical [Costa et al., 2005]. Blom and Bar-Shalom [1988] proposed the most famous suboptimal solution that attempted to control or prune the size of the bank of Kalman filters. The majority of works on the last two decades seems to be in the direction of adapting the particle filter for the MJLS [Isard and Blake, 1998b] Doucet et al. [2000b] Doucet et al. [2001b] Maggio and Cavallaro [2005] Andrieu et al. [2003] Andrieu et al. [2002] Blom and Bloem [2004].

Another way of approaching the problem is through the linear optimal solution (Linear Minimum Mean Squared Error-LMMSE) proposed by Costa [1994]. This is in fact the basic idea of one of the developments of this thesis [**?**].

2.7 Level-set methods for contour tracking

The second main modality of object contour tracking is suitable for appearance-based posture classification and gesture interpretation. I was inspired to use level-set methods [Sethian, 1997] for contour tracking specially by impressive recent results in literature, for example, Yilmaz

et al. [2004] and Cremers' [Cremers et al., 2007] [Cremers et al., 2003] Giraldi et al. [2007]. Osher and Paragios [2003] provides a valuable overview of the topic.

2.7.1 Robust contour evolution

Osher and Sethian [1988b] proposed the level-set method motivated by the problem of numeric hypersurface propagation. The basic idea is to evolve the desired hypersurface implicitly by explicitly evolving instead, an embedding function in a higher dimensional space (Figure 2.18). At any time, the aimed hypersurface can be recovered, for the hypersurface is defined as the zeroth-level set of the embedding function. Although this may seem the longer way around, in fact, this implicit representation comes with concrete advantages:

i. There is no need to parameterize the embedded hypersurface

ii. Level-set makes it *natural* to track the evolution of contours that undergo significant *topological* changes

In the level-set framework, the numerical evolution of the contour is given by the solution of a Hamilton-Jacobi partial differential equation using, for example, finite differences. Although this would not be considered a slow process in itself, it should be reminded that the iterative solution of this equation generally implies non real-time performance.

There is an ample literature on improving the robustness of level-set methods against local minima (distractions) in the image (vide Giraldi et al. [2007] and references therein).

Most of the work done in improving speed of the level-set method has been done in the sense of limiting the evolution at each time step to a narrow band around the current zero level set, however, this approach carries difficulties such as narrow-band definition and re-initialization.

2.7.2 Fast contour evolution

Shi and Karl [2005b] observe that the traditional approach of evolving the surface accordingly to a PDE is unnecessary at pixel resolution in computer vision problems where the halt point of the evolution is what really counts and the actual way the contour evolves to that point is irrelevant. Based on this assumption, Shi and Karl [Shi, 2005] [Shi and Karl, 2005a] [Shi and Karl, 2005b] developed a contour tracking algorithm that holds to the main positive characteristics of level-set methods (topology freedom and no contour parameterization), but carries out pixel-level contour evolution in a much simplified manner, in fact, by executing only simple operations with two linked lists. Their algorithm is very easy to implement and runs on a standard PC in real-time, still leaving plenty of CPU cycles for other processes.

This approach, however, lacks effective mechanisms for local minima avoidance and depends heavily on the accuracy of pixel-level feature processing as will be seen in the Chapter 4.

2.8 Shape features

In order to recognize postures in the appearance-based framework, it is still necessary to compress the shape information contained in the contour points in a compact shape feature-vector. Therefore, the machine learning algorithm that recognizes the hand postures deals with those shape feature-vectors instead of a large amount of contour points.

A needed property of this compact shape descriptor is invariance to hand position, scale and orientation, that is, it has to be a *pure* description of the posture -the finger configuration. Promising invariant shape descriptors or features are the recently proposed *shape context* by Belongie and collaborators [Belongie and Malik, 2000] [Belongie et al., 2002].

Another option is to use the technology introduced by Manay et al. [2006], called *integral invariants*, for shape comparison directly from the points.

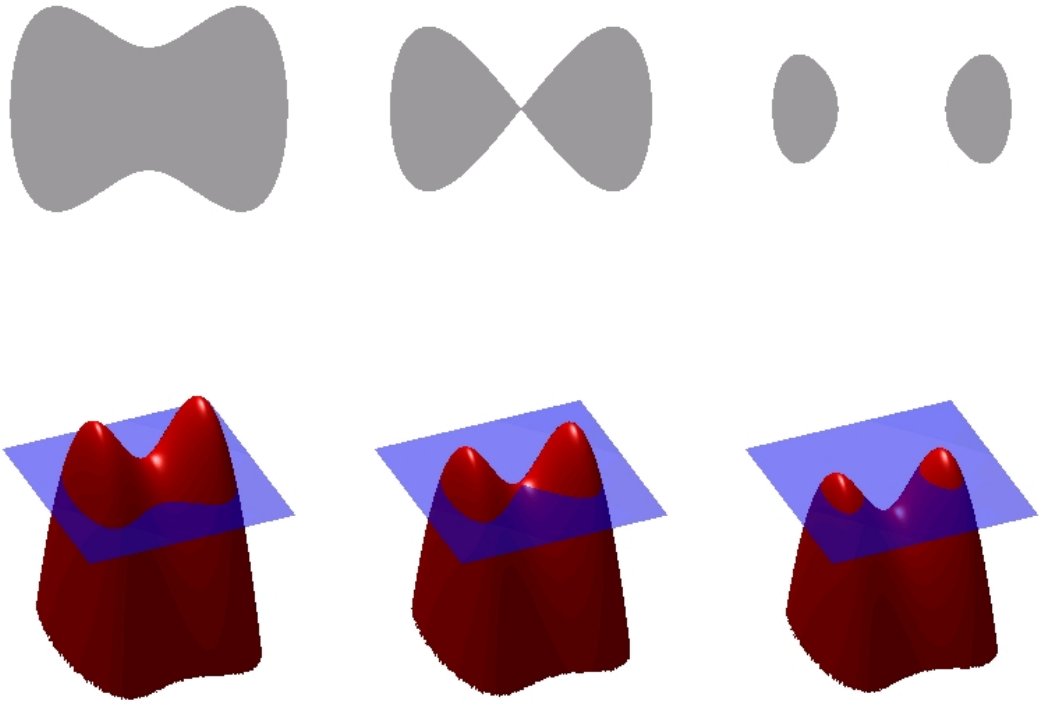


Figure 2.18: The 2D contour is defined as the intersection of the surface with the zeroth-level plane.

Hu [1962] derived a compact set of invariant measures based on moments. His measures can be used as a shape feature-vector for shape recognition. In fact, this was the method used in some of the experiments presented in this thesis.

There is, nevertheless, a vast literature in shape matching and recognition, which is out of the scope of this thesis.

Chapter 3

Low-level Feature Extraction

3.1 Motivation

This chapter discusses the methodology I used for pixel-level analysis. The human vision system relies heavily on two basic intelligence functions: *memorization* and *generalization* ability. An artificial vision system like the ones investigated here will perform proportionally to how well it can emulate those human abilities which are fundamental for the recognition process. These two cognitive properties (memory and generalization) will be a recurrent underlying theme throughout this text.

An intuitive illustration of how memory (prior knowledge) and generalization ability participate in the process of understanding visual information is given by Figure 3.1. Most people will find it difficult to understand the scene until some hint is given (dog). The hint is enough for the brain to go fetch in the memory the right prior visual knowledge needed to understand the scene. Even if the observer has never seen that specific scene before, the human brain ability to generalize will fill in the gaps.

The most general mathematical theory to model the problem of accumulating knowledge (memorizing) and generalizing knowledge to new instances of data is the modern statistical learning (or inference) theory.

Conceptually, the statistical learning theory is concerned with the problem of learning dependencies on empirical data [Vapnik, 1998]. This definition of the problem may sound too abstract at first sight; however, if we look into its two major instances, things get more familiar.

i. *parametric statistical inference problem*: Given that the basic stochastic law (type of the probability distribution function) of the data source is known, estimate the parameters that specify the probability function using the data. The basic method for solving this problem was devised by Fisher [1952], called generally the Maximum Likelihood method.

ii. *nonparametric or general statistical inference problem*: There is no reliable way of modeling the data source with any known stochastic law. The goal is to infer an approximating function (of the data source behavior) from given examples.

Vapnik [1998][p.1] observes that

> ...after the appearance of the first generation of computers capable of conducting multidimensional analysis in the 1960s researchers soon realized that existing classical approaches to low-dimensional function estimation problems do not reflect singularities of high-dimensional cases.

Richard Bellman, the inventor of *dynamic programming*, pointed out that the problems arising in higher-dimensional spaces were due to the exponential increase in *hypervolume* associated with the increase in dimension [Bellman, 1961]. The number of samples evenly distributed in space needed for a function approximation is directly proportional to the hypervolume of the

Figure 3.1: Prior knowledge and generalization ability are essential in visual understanding.

space; therefore it also increases exponentially with the space dimension. This *drama* was dubbed the *curse of dimensionality* by Bellman.

Higher-dimensional feature vectors are, thus, usually avoided by applied researchers because they know (sometimes intuitively) that performance is seriously challenged in high dimensions. This trend may lead, however, to oversimplification of the data representation or overcompression, with consequent loss of useful information.

My approach to pixel-level analysis is motivated by the belief -acquired from experimentation- that the object-pixel appearance is better modeled as a general statistical inference problem, opposed to a parametric one, and that, although pixel color itself is defined by a three component vector, object-pixel identification is more robust if a higher-dimensional pixel representation is used.

3.2 The codification of prior knowledge

Both the parametric and the general statistical inference paradigms deal ultimately with the codification of knowledge extracted from empirical data. Another terminology widely utilized for codification is *modeling*. The verb *to model* is more often used, however, meaning the codification of the designer's prior knowledge of the problem, as opposed to the codification of knowledge contained in data. Note that I could have used the wording *information contained in data* here, but it would be misleading because what I am interested is in extracting actual knowledge about the data source behavior.

In this chapter a method is shown for object-pixel appearance modeling from examples. More specifically, the idea is to depart from the traditional way of modeling object-pixel appearance (color) based on the parametric learning paradigm, by employing a full-fledged general (nonparametric) machine learning algorithm, in this case, support vector machines. The following sections present machine learning theory and support vector machines.

3.3 Learning machines

In the context of binary classification, that is, pattern classification with only two possible classes, the solution to the general statistical learning problem is given by methods that are generally called today as learning machines. Formally, a learning machine is defined as a family $\mathcal{F}$ of $f:\ \Re^F \times \Re^P \longmapsto \{1,-1\}$ mappings, such that, $\Re^F$ is the feature vector space and $\Re^P$ is the machine parameter space, or equivalently the training parameter space. The goal is therefore to find the machine parameter vector $\alpha \in \Re^P$ that enables the machine to emulate (by memorizing) the mapping defined by the training data set

$$\mathcal{T}:\quad \{\mathbf{x}_i \longmapsto y_i\}_{i=1,\ldots,l}\,, \mathbf{x}_i \in \Re^F, y_i \in \{1,-1\} \tag{3.1}$$

where the set $\{1,-1\}$ is called the set of labels.

Moreover, we do not only want the machine to plainly memorize the training data, we want it to generalize well. In other words, the machine is supposed to *learn the behavior* of the data source from which the training data is generated, therefore being able to map (label) elements of the feature space $\Re^F$ not in the training set (3.1) in the same way the original data source would do it (at least most of the time). Let us say the data source behavior, which is stochastic in nature, is described by a joint probability distribution $P(\mathbf{x}, y)$ defined in $\Re^F \times \{1,-1\}$. This way, the learning goal is equivalent to finding the machine parameter-vector α which minimizes the general test error

$$R(\alpha) = \int \frac{1}{2}|y - f(\mathbf{x},\alpha)|dP(\mathbf{x},y) \tag{3.2}$$

which will be called here *true* or *generalization* error (or risk) as opposed to the *empirical error*

defined below

$$R_{emp}(\alpha) = \sum_{i=1,\ldots,l} |y_i - f(\mathbf{x}_i, y_i)| \tag{3.3}$$

Note that $P(\mathbf{x}, y)$ is not known and difficult to estimate reasonably. The true error $R(\alpha)$, therefore, is not very useful in practice, although very important conceptually. On the other hand, the empirical error $R_{emp}(\alpha)$, which is defined in terms of the training data at hand, is easy to compute. In fact, more naive machine learning algorithms usually assume that the true error can be approximated by the empirical one.

3.4 Tools for machine learning analysis

The ultimate goal of machine learning research is to develop methods to find and train learning machines that generalize well. For this, some important theoretical tools have been proposed in the last four decades that serve as a guide in the process of analysis and development of machine learning methods. We briefly discuss some of these important theoretical results in this section. For a more comprehensive overview of these concepts I refer the reader to the tutorial of Burges [1998], the overview paper of Vapnik [1999] and/or the one by Müller et al. [2001], as well as to the paper of Giraldi et al. [2008].

3.4.1 Vapnik-Chervonenkis (VC) dimension

The VC dimension is defined in terms of the concept of point shattering by a function set. Let us understand this concept with an illustration.

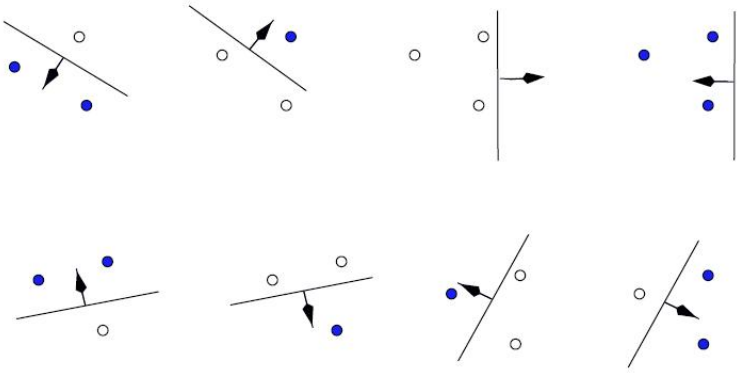


Figure 3.2: Three points in $\Re^2$, shattered by oriented lines (from Burges [1998]).

Suppose that the space in which the data live is $\Re^2$, and the set $\{f(\cdot, \alpha)\}$ consists of oriented straight lines, so that for a given line, all points on one side are assigned the class 1, and all points on the other side, the class -1. The orientation is shown in Figure 3.2 by an arrow, specifying on which side of the line points are to be assigned the label 1. While it is possible to find three points that can be *shattered* by this set of functions, it is not possible to find four. Such that, in general, we say that a set of functions is able to shatter a data set (of feature vectors) with l elements if, and only if, for all 2^l possible labeling of the data set, there is at least one $\alpha \in \Re^P$, such that, $f(\mathbf{x}, \alpha)$ shatters it.

The VC dimension h is a property of the learning machine $\{(\mathbf{x}, \alpha) \longmapsto f(\mathbf{x}, \alpha) \in \{1, -1\}\}$, and is defined as the maximum number of points in $\Re^F$ that can be shattered by it. In the case of the illustration above ($\Re^2$), $h = 3$.

A result that might prove useful when computing the VC dimension for a given machine is stated in the following theorem

Theorem 3.4.1. *Consider some set of m points in $\Re^F$. Choose any of the points as origin. Then the m points can be shattered by oriented hyperplanes if and only if the position vectors of the remaining points are linearly independent.*

Corollary 3.4.2. *The VC dimension of the set of oriented hyperplanes in $\Re^F$ is $n+1$, since we can always choose $n+1$ points, and then choose one of the points as origin, such that the position vectors of the remaining n points are linearly independent, but can never choose $n+2$ such points (since no $n+1$ vectors in $\Re^F$ can be linearly independent).*

The VC dimension is a measure of the *capacity* of a learning machine. Capacity is a concept closely related to generalization, although, not trivially. Burges would summarize the concept of capacity with the following metaphor

> A machine with too much capacity is like a botanist with photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree. Neither generalizes well.

3.4.2 Vapnik-Chervonenkis (VC) confidence

As it was mentioned earlier, there is no easy way of accessing $P(\mathbf{x}, y)$ (knowledge of the data source). However an important result in statistical learning theory establishes a bound that relates the empirical risk with the true risk, fair to say, in a rather loose way. Let us be reminded that the loss $\frac{1}{2}|y - f(\mathbf{x}, \alpha)|$ in the expression of the empirical error (3.3) assume values 0 or 1 in the binary classification case we consider here. It can then be proved that

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(log(2l/h)+1) - log(\eta/4)}{l}\right)} \tag{3.4}$$

is valid with probability $1-\eta$, where $\eta \in [0,1]$. The second term on the right hand side of the equation is called the *VC confidence*.

Although this is an upper bound (with some probability) of the true risk, this does not prevent a particular machine with the same value for empirical risk, and whose function set has a higher (even infinite) VC-dimension, from having better performance. Bounds, nevertheless, offer helpful theoretical insights into the nature of learning problems.

3.4.3 Margins

If the training data is classifiable by linear functions (hyperplanes), that is, functions of the form

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{3.5}$$

which are defined by the hyperplane normal vector $\mathbf{w}$ (called weight vector) and the offset b, Each of the halfspaces defined by this hyperplane correspond to one class, that is, the classification is given by $sgn(\mathbf{w} \cdot x + b)$. It can be proved that the VC dimension itself can be bounded in terms of the *margin* (Figure 3.3). The margin is defined as the minimal distance of a sample to the decision surface. It turns out that the margin can be measured by the length of the weight vector $\mathbf{w}$ in (3.5), in the following manner: again, given that the training data is separable, $\mathbf{w}$ and b can be rescaled such that the points closest to the hyperplane satisfy $|\mathbf{w} \cdot \mathbf{x}_i + b| = 1$, i.e., obtain the so called canonical representation of the hyperplane. Now consider two samples $\mathbf{x}_2$ and $\mathbf{x}_2$ from different classes with $\mathbf{w} \cdot \mathbf{x}_1 + b = 1$ and $\mathbf{w} \cdot \mathbf{x}_2 + b = -1$, respectively. Then the margin is given by the distance of these two points measured perpendicular to the hyperplane

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2)/\|\mathbf{w}\| = 2/\|\mathbf{w}\|. \tag{3.6}$$

Margins are related to the VC-dimension h through the inequality[Vapnik, 2000]:

$$h \leq \|\mathbf{w}\|^2 R^2 + 1 \tag{3.7}$$

where R is the radius of the smallest ball around the training data, which is fixed for a given data set.

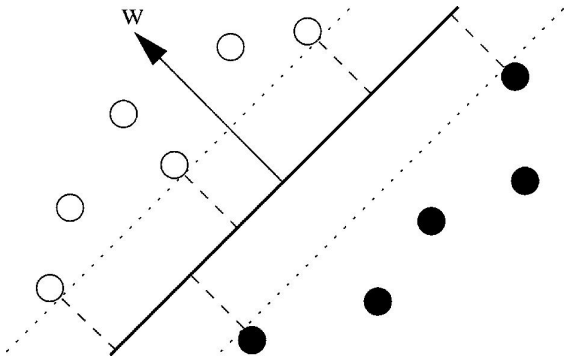


Figure 3.3: Linear classifier (thick line) and margin (the minimal distance between any training points to the hyperplane).

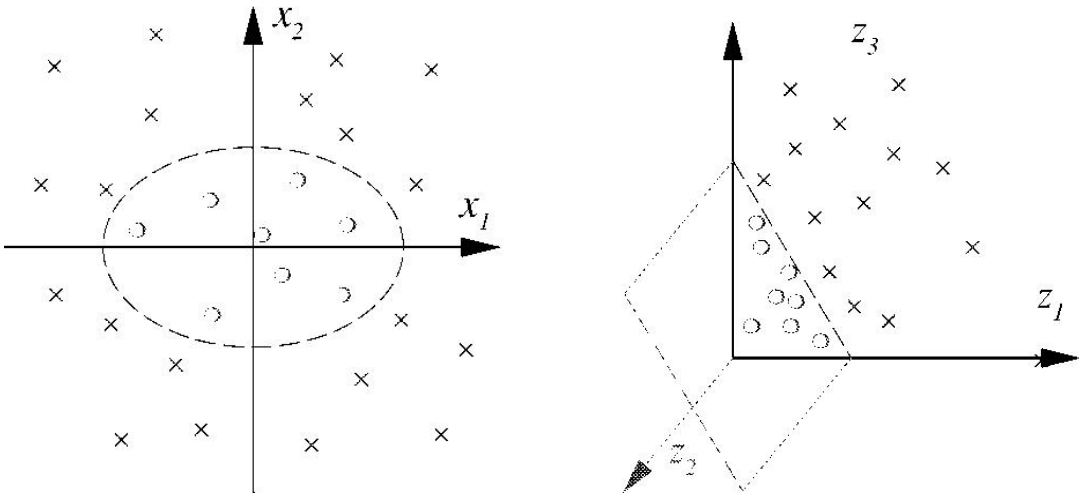
Margin is an important concept in learning algorithm development. It provides a criterion of optimality for the learning algorithm. The optimal separating hyperplane for a given training data set will be the one that maximizes the margin. There is a sound mathematical formulation for the problem of finding the optimal separating hyperplane (training the linear classifier) which will be addressed later (Section 3.5).

3.4.4 The kernel trick

In general, complex real-world problems require more expressive hypothesis spaces than hyperplanes, that is, linear classifiers are just too simplistic to express $P(\mathbf{x}, y)$ realistically. One way to address this problem is the use of multiple *layers* of threshold linear classifiers, which is the case of multi-layer neural networks. Another solution would be mapping the data to a higher-dimensional space in which the data source non-linearity would not "seem" non-linear anymore, then find the optimal hyperplane in this new space (Figure 3.4).

The curse of dimensionality says that learning $P(\mathbf{x}, y)$ gets dramatically more difficult as the dimension F of the feature space $\Re^F$ increases, that is because the number of patterns needed to sample the space properly increases exponentially with F and this can be a serious practical impediment to the implementation of learning algorithms in higher-dimensional spaces. However, in certain learning methods, training data elements never appear isolated but always in the form of *inner* products $\mathbf{x}_i \cdot \mathbf{x}_j$ between pairs of elements, that is, elements appear comparatively to other elements since inner products are a measure of *similarity* in the feature space. This is the case for linear support vector machines, Fisher discriminants methods and Principal Component analysis, for example. In these specific cases, it is sometimes possible to have the best of both worlds, that is, a **non-linear classifier** in the input data space $\Re^F$ by solving a **linear classification problem** in the induced feature space, and this is done by means of the kernel method also known as the kernel trick.

In a way, kernel techniques formalize the notion of *similarity* of data, kernels provide a representation of the data in an induced *reproducing kernel Hilbert space (RKHS)* that can be even infinite-dimensional. It is not the scope of this thesis an in-depth investigation of kernels, their properties, and RKHS theory and one will not be given here. For further reading on kernels I suggest the book of Schölkopf and Semola [2001] and references therein.



(a) Using second-order monomials.

(b) The points in a) are mapped to a higher dimensional space and their images are classifiable by a hyperplane now.

Figure 3.4: Two-dimensional classification example (from Schölkopf and Semola [2001]).

The basic idea of kernel methods is that, although it is tremendously difficult and most of the time computationally unviable to come up with a mapping Φ of the training data into some higher-dimensional induced feature space $\mathcal{H}$, that is a $\{\mathbf{x}_i \longmapsto \Phi(\mathbf{x}_i) \in \mathcal{H}\}_{\mathbf{x}_i \in \Re^F}$, it is sometimes straightforward to come up with an operator that carries out the dot product of the images -through Φ- of any pair of feature vectors $(\mathbf{x}_i, \mathbf{x}_j) \in \Re^F \times \Re^F$, that is, an operator or kernel $k : \Re^F \times \Re^F \longmapsto \Re$ such that

$$k(\mathbf{x}, \mathbf{y}) = \langle \Phi(x), \Phi(y) \rangle \tag{3.8}$$

but never explicitly compute $\Phi(x)$ nor $\Phi(y)$.

3.5 SVMs for supervised learning

The most celebrated machine learning algorithm inspired by the modern statistical learning theory is the support vector machine. Basically, the support vector machine constructs a solution for the maximal margin separating hyperplane problem with a very convenient properties, as we will see in the following.

3.5.1 Optimal separating hyperplane

I will start by assuming the simpler linear case with a separable training data set. Therefore, given the training data set (3.1), we are interested in constructing the decision rule

$$f(\mathbf{x}) = sgn(\mathbf{w} \cdot \mathbf{x} + b) \tag{3.9}$$

by determining a pair $(\mathbf{w}, b)$ that lower the upper bound on the generalization error given by (3.4) as much as possible. The upper bound is the sum of the empirical risk and the VC-confidence term. This is called the Structural Risk Minimization Induction Principle.

We can guarantee a minimum empirical risk by restricting the solution space to the solutions that satisfy:

$$y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \quad i = 1, \ldots, l. \tag{3.10}$$

Note that the VC-confidence term is a monotonically increasing function of the VC-dimension, which turns out to be upper bounded by $\|\mathbf{w}\|^2$. Therefore, as was expected, we want a solution that maximizes the margin which is equivalent to minimize $\|\mathbf{w}\|^2$ because of (3.7).

The problem can then be formally put as a standard quadratic minimization with constraints

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.11}$$

$$\text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, i = 1, \ldots, l. \tag{3.12}$$

which can be rewritten, through the introduction of Lagrange multipliers α_i for each training data element, as finding the saddle point of the Lagrangian functional

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) \tag{3.13}$$

by minimizing it with respect to $\mathbf{w}$ and b and maximizing it with respect to the α_is. As a consequence we need to assure

$$\frac{\delta \mathcal{L}}{\delta b} = 0 \text{ and } \frac{\delta \mathcal{L}}{\delta \mathbf{w}} = 0 \tag{3.14}$$

at the optimal point. This translates to

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \text{ and } \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{3.15}$$

Which shows that $\mathbf{w}$ is a linear combination of the training inputs. By substituting (3.15) into (3.14), the problem is again rewritten, now in a form called the *dual quadratic optimization* problem

$$\max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{3.16}$$

$$\text{subject to } \alpha_i \geq 0 \;\; i = 1, \ldots, l \tag{3.17}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.18}$$

Solving the dual provides the α_is that substituted in the right hand-side of (3.15) yields the optimal $\mathbf{w}$. The offset b can then be easily estimated as will be seen in the next section.

3.5.2 The support vector machine

Support vector machine is the generalization of the above results for non-linear optimization on data space and non-separable training data. That is, situations like the one illustrated in Figure 3.5.

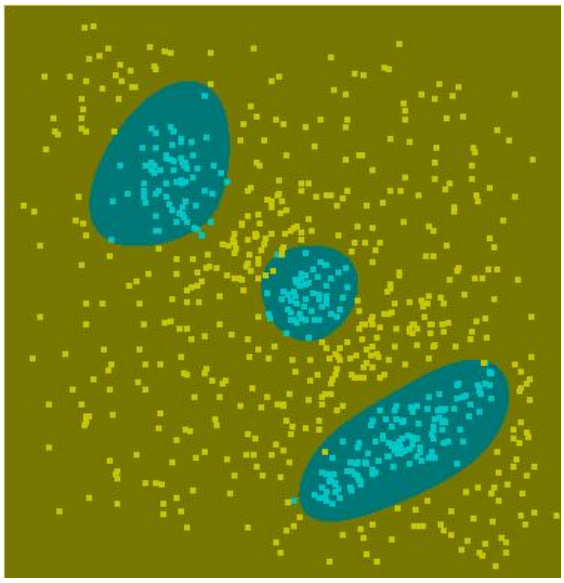


Figure 3.5: Illustration of a non-linear partition of data with SVM.

The non-linearization is easily obtained through the kernel trick. The dual formulation

makes this really simple, since we only need to substitute the inner product by the kernel operator

$$\max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{3.19}$$

$$\text{subject to} \quad \alpha_i \geq 0 i = 1, \ldots, l \tag{3.20}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.21}$$

and the decision rule becomes, with the help of (3.15):

$$f(\mathbf{x}) = sgn \left(\sum_{i=1}^{l} y_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b \right) \tag{3.22}$$

The transition from the separable data to non-separable data is made by alleviating the restriction of null empirical risk. This is done via the introduction of slack-variables to relax the hard-margin constraints

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \psi_i \ \psi_i \geq 0, i = 1, \ldots, l. \tag{3.23}$$

in the linear case or

$$y_i(\mathbf{w} \cdot \Psi(\mathbf{x}_i) + b) \geq 1 - \psi_i \ \psi_i \geq 0, i = 1, \ldots, l. \tag{3.24}$$

in the non-linear one. This way, room is made for noisy outliers (misclassifications) in the training data. The basic minimization problem (3.11) turns into

$$\min_{\mathbf{w}, b, \psi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{l} \psi_i \tag{3.25}$$

$$\text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, i = 1, \ldots, l. \tag{3.26}$$

where the regularization constant $C > 0$ determines the tradeoff between the empirical risk and the capacity (VC-confidence) term. This leads to the dual problem

$$\max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{3.27}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \ \ i = 1, \ldots, l \tag{3.28}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.29}$$

this differs from the standard version only in that it adds a new constraint to the Lagrange multipliers.

Most optimization methods are based on the second-order optimality conditions, so called Karush-Kuhn-Tucker (KKT) conditions which state necessary and in some cases sufficient conditions for a set of variables to be optimal for an optimization problem. For the dual (3.27), it turns out that the KKT conditions imply that:

$$\alpha_i = 0 \implies y_i f(\mathbf{x}_i) \geq 1 \text{ and } \psi_i = 0 \tag{3.30}$$

$$0 < \alpha_i < C \implies y_i f(\mathbf{x}_i) = 1 \text{ and } \psi_i = 0 \tag{3.31}$$

$$\alpha_i = C \implies y_i f(\mathbf{x}_i) \leq 1 \text{ and } \psi_i \geq 0 \tag{3.32}$$

which shows that most of the α_is are equal to zero, that is, SVMs are sparse in α. Equations

(3.30) mean that only the α_is connected with a training datum $\mathbf{x}_i$ on the margin (or inside the margin area) will be non-zero, that is, only the support vectors count for the construction of the decision rule.

Moreover, for all support vectors $\mathbf{x}_i$ with $0 < \alpha_i < C$, the slack variable ψ_i is zero. Thus, for any support vector $\mathbf{x}_i$ with $i \in I \cong \{i|0 < \alpha_i < C\}$

$$y_i \left(b + \sum_{j=1}^{l} y_j \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right) = 1. \tag{3.33}$$

from which a numerically stable estimation of b can be obtained from

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_i - \sum_{j=1}^{l} y_j \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right). \tag{3.34}$$

A support vector machine learning or training method is the numerical solution of the Lagrangian (3.27) in order to obtain the non-linear decision rule (3.22). Various numerical methods exist for solving the SVM minimization problem (quadratic programming problem - QP), for further reading on this topic one may find useful the book of Nocedal and Wright [2006] or the one of Bertsekas [1995].

Table 3.1 shows a set of usual choices for the kernel function $k(\cdot, \cdot)$. Each kernel presents different characteristics and behavior. In initial exploratory tests, the RBF kernel has performed consistently better than the others, thus, all experiments using support vector machines in the thesis were done using RBF kernel functions.

Name	Function
Polynomial (homogeneous)	$k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^d$
Polynomial (inhomogeneous)	$k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + 1)^d$
Radial Basis Function (RBF)	$k(\mathbf{x}_1, \mathbf{x}_2) = e^{-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2}$, for $\gamma > 0$
Gaussian RBF	$k(\mathbf{x}_1, \mathbf{x}_2) = e^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}}$
Sigmoid	$k(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\kappa \mathbf{x}_1 \cdot \mathbf{x}_2 + c)$, $\kappa > 0$ and $c < 0$

Table 3.1: Kernel Functions

This point concludes the presentation of the statistical learning theory and support vector machines, which is the core building block of the pixel classification method proposed in this thesis. The remaining sections of the chapter will briefly discuss the problem of pixel classification and describe in detail the tasks involved in this SVM-based pixel classification method.

3.6 Pixel classification

In this section, the problem of pixel classification is discussed and a SVM-based method is described in detail. Pixel classification is an essential part of the contour tracking algorithms that will be proposed in the following chapters, Figure 3.6 depicts the main low-level (pixel-level) processing tasks involved before the contour module takes over.

Pixel classification is based on an object pixel-model (or appearance model) which yields, somehow, a criterion for classification. The appearance model is extracted from a set of training examples, in the case, annotated images, i.e., images of which the label (class) of every pixel is known (see Section 3.8 on the method for image annotation). One of the simplest appearance models there is, is a Gaussian centered at the mean skin-color value. Gaussians are determined by a center value ν and a standard deviation σ, and can be used to determine if a pixel p is on

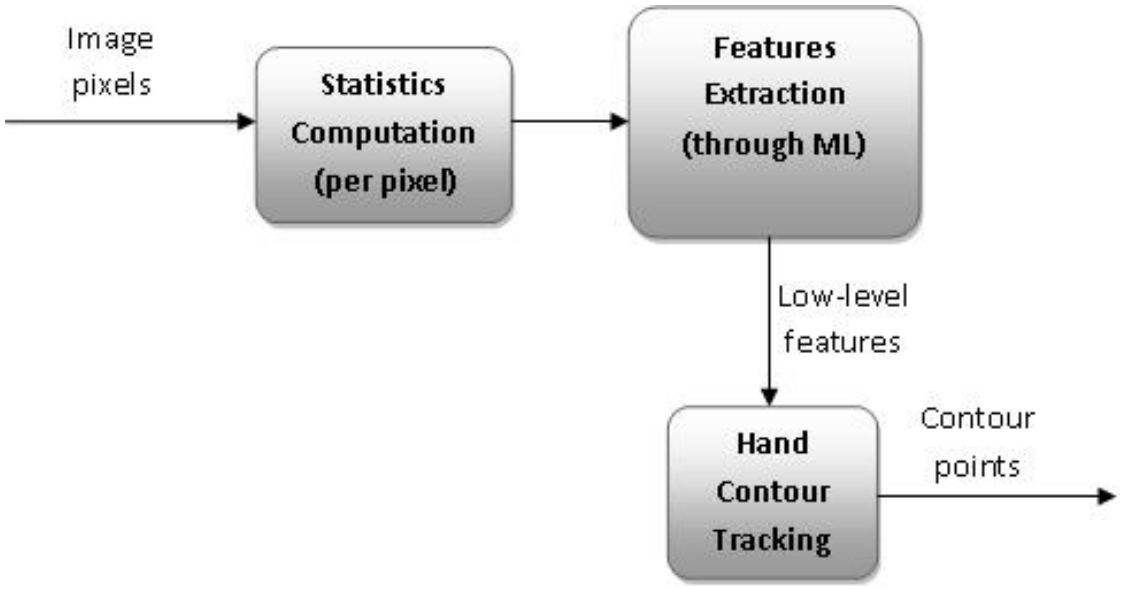


Figure 3.6: Low-level Features Extraction

the object or the background with the test:

$$\begin{cases} \|p - \nu\| < c\sigma, & p \in \text{object} \\ \text{otherwise}, & p \in \text{ background} \end{cases} \tag{3.35}$$

where c is a constant that dictates how restrictive or how loose the criteria should be, generally called color threshold. The acquisition of a Gaussian model is trivial, since it reduces to computing an average pixel value and the covariance matrix or just its main diagonal, the variance vector, if each pixel component is going to be analyzed separately.

A much more accurate appearance model is attained with a Mixture of Gaussians. That is, a number of Gaussians is used to model the different dominant nuances of the object appearance. This model is usually obtained from the training data via Expectation Maximization (EM) algorithms. This type of modeling is, in fact, the base for state-of-the-art algorithms for which it is necessary to achieve high generalization from a relatively small amount of available training data, the case considered here. In practice, the user of the GPUI will have the opportunity for initializing the system with his hand color information by following some predetermined steps which provide a few sample frames that can be used for *initialization* before the actual gesture recognition process starts. From these initialization frames, training data will be extracted for the hand appearance modeling.

Figure 3.7 shows the distributions of pixel values for the foreground (object) and background of a hypothetical initialization image in two different color spaces. In the initialization step it is necessary to train a pixel classifier to be used in the contour tracking process. The figure gives a notion of the difficulty of the problem, for the data is not trivially separable.

The standard computer representation of color reserves eight bits for each color component; therefore, each component (red, green and blue) may assume 256 intensities. In a way or another, pixel classification is a kind of, or involves, *color comparison*. Direct pixel comparison in RGB space, that is, by means of L_2 Euclidian norms like $\|\mathbf{p}_1 - \mathbf{p}_2\|_{L_2}$, where $\mathbf{p}_1$ and $\mathbf{p}_2$ are pixels, usually results in very poor results. That is because the RGB color system was designed aiming at light emission by image rendering hardware not the non-linearity of the human visual system, consequently, some colors that are easily distinguishable by an individual are very close numerically in the RGB space. On the other hand, sometimes subtle variations in illumination over an object of some color, for example, may be barely noticed by people, but have clearly different RGB values.

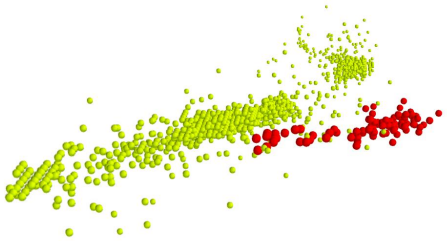
In this context, we can readily identify three approaches to avoid these difficulties:

i. In RGB space, devise a metric (a norm) that reflects the human vision.

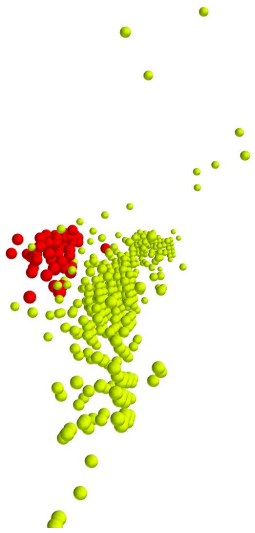
ii. Use some other color space in which standard Euclidian metric will emulate human vision.

iii. In RGB, devise an appearance model based on a strong classifier like SVMs that accurately accounts for complex (non-linear) intra-object color variability and can be used for color comparison without the use of a metric in color space.

The path iii has not been fully explored in the past for two main reasons: There was a tendency to believe that color characterizations of objects were more easily separable in nature; and the application of a strong classifiers for an intrinsically CPU-intensive process such as pixel classification may be painfully slow. I reason that, once embedded in contour tracking methods that minimize the amount of pixel classification operations per frame, path iii yields a feasible and efficient solution. Nonetheless, solutions based on strategy i and ii will be discussed and used as a means of evaluation, by comparison, of the performance of the SVM-based classifier proposed here.



(a) RGB



(b) CIE Lab

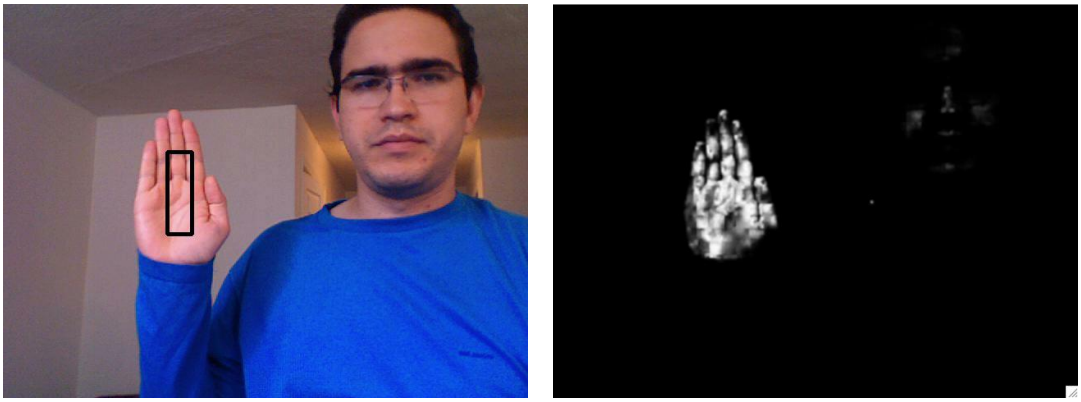
Figure 3.7: The pixel values distribution for a hand image and the background in the RGB and CIE Lab color spaces. Only two thousand randomly sampled pixels are displayed. The bigger red balls are hand values and the smaller green ones are background values.

3.6.1 Color spaces

The CIE Lab color space is a color-opponent space with dimension L for lightness and a and b for the color-opponent dimensions, based on nonlinearly-compressed CIE XYZ color space coordinates. The CIE XYZ color space is obtained from the RGB color space through a standard linear transformation. The CIE Lab color space is meant to be *perceptually* uniform, that is, numerical differences should be perceived by people on average as a visual difference of the same *importance*. CIE stands for *Commission Internationale d'Eclairage* (International Commission on Illumination), was established in 1913 and based in Vienna, Austria. The CIE is the international authority on light, illumination, color, and color spaces.

3.6.2 Similarity measures

Another way to approach the problem of pixel comparison is to use a measure (a norm) that would better mimic the human vision non-linear characteristics. A candidate measure, for example, would be the Mahalanobis statistical distance, which was, in fact the base of the solution proposed in Cordeiro Junior et al. [2008]. The Mahalanobis distance



(a) Original picture showing a rectangular color sample. (b) A Mahalanobis distance based intensity map that highlights the skin regions of the hand.

Figure 3.8: A color sample is used for computing the covariance matrix used in the calculation of the Mahalanobis distance.

3.7 SVM-based pixel classification

3.7.1 Feature design

The main inspiration for the design of the feature vector adopted here is the way the human visual system works. Given an isolated pixel from an image, a person is likely to find it really hard to determine if it belongs to the object, however, given a little block of pixels it is easier to determine if the pixels in it belong to a skin region for example. The hint the human system gives is then: Not to look at the pixel alone, but use information from its vicinity as well. This approach may increase the dimension of the pixel description, but certainly reflects more realistically the human vision behavior.

There are various ways of characterizing the content of a pixel neighborhood, for instance, by truncating the frequency content (via FFT), Wavelets transformations, or Haar features. One very simple characterization, and yet effective in practice, is to use the first and second moments, that is, to use as the feature vector the average and the variance of the color value of the pixels in the neighborhood. Note that the neighborhood is defined by a window of small dimensions, typically 10×10 (a 100 pixels square) centered on the pixel being classified. In summa, the

feature vector associated with a pixel p_{ij} would be given by

$$\mathbf{x}_{-}(i,j) = (r(i,j), g(i,j), b(i,j), \overline{r(i,j)}, \overline{b(i,j)}, \overline{b(i,j)}, \widehat{r(i,j)}, \widehat{g(i,j)}, \widehat{b(i,j)})^T \tag{3.36}$$

where, $r(i,j), g(i,j), b(i,j)$ are the pixel color components, the over line means the average taken in a neighborhood and the wide hat means the variance taken in the same neighborhood. One might choose to work with the nine elements of the covariance matrix instead of the three elements of the variance with pros and cons. The $-$ subscript is to indicate that this feature vector does not include the background subtraction features discussed next.

3.7.2 Background subtraction for free

Background subtraction is the common operation in computer vision of using a stored image of the background to eliminate background clutter. An intuitive description of the background operation is: loop through all frame pixels p_{ij} in the image and set equal to zero those pixels that satisfy

$$\|background_{ij} - p_{ij}\| < Th \tag{3.37}$$

i.e., those pixels that did not change significantly from the background. The threshold constant Th generally incompasses the idea of significant change. This simplistic methodology is not very useful the way it stands, for illumination fluctuations can seriously hamper performance or make the determination of a reasonable value for Th impossible. Evidently, much more complex methods were proposed in the literature for more useful background subtraction (see review of Piccardi [2004]). To lessen the effect of illumination, the background can be modeled not only with a single image but with sequence of images that contain acceptable levels of illumination variation, with, Gaussians, Mixture of Gaussians, HMM and others utilized as modeling tools. Although background subtraction is extremely powerful in noise elimination, it can be of difficult implementation, time-consuming and often CPU-hungry.

The machine learning framework adopted in this thesis provides, however, a natural and efficient form for integrating background subtraction into the process of pixel classification in a rather seamless fashion, that is, without significant changes to the algorithm structure (or source code). This is achieved by adding new features to the feature vector $\mathbf{x}$ of the form $\Lambda(bg) = (r(i,j) - r_{bg}(i,j), g(i,j) - g_{bg}(i,j), b(i,j) - b_{bg}(i,j))$ for every available background image bg.

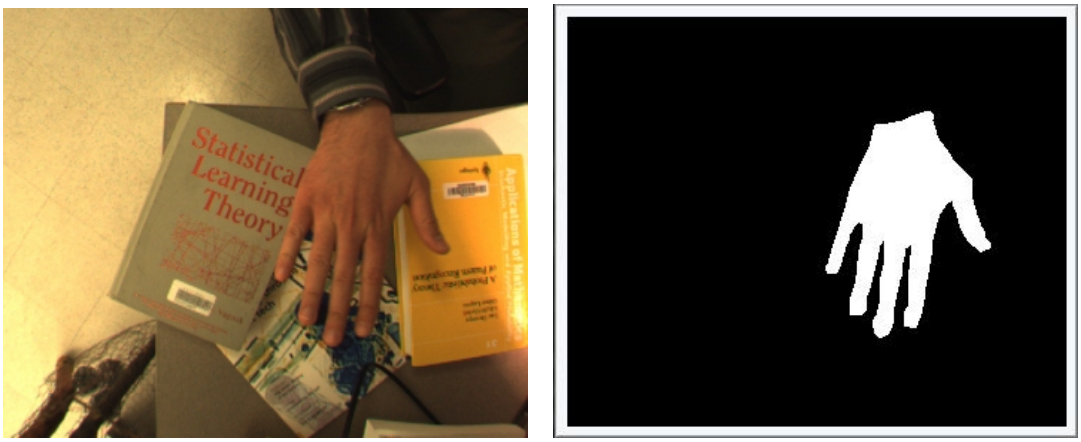
$$\mathbf{x} = (\mathbf{x}_{-}^T, \Lambda(bg_1), \Lambda(bg_2), \ldots, \Lambda(bg_s)) \tag{3.38}$$

where s is the number of available background images.

3.8 Manual image annotation

As mentioned before, the training process requires annotated images. The process of image annotation is very tedious and needs someone to do it manually. Therefore, producing large amounts of annotated images is impractical; actualy, for the production of larger databases of annotated images, it is usually made use of *ad-hoc* software tools that facilitate the job [Jones and Rehg, 2002].

The image annotation method I adopted takes about a minute per frame. Basically, the person is prompted to provide points on the border of the hand by clicking on the image in order to define a contour polygon. Then, a matrix of the same height and width of the original image is produced, such that the elements inside the contour polygon are set to one and the ones outside are set to zero (see Figure 3.9), this matrix is usually called annotated image or label matrix because every element labels a pixel. The function `cvPointPolygonTest` of the OpenCV library was used for determining if an element of the label matrix is inside or outside the contour polygon.



(a) Original image. Points on the hand borders are taken with the mouse.

(b) Resulting label matrix.

Figure 3.9: Hand annotation by hand.

Note that, since the annotation process depends on the user's ability, the result is far from perfect. Therefore, mislabeled pixels will invariably be introduced in the training set, increasing the challenge for the classifier.

3.9 Segmentation experiments

A good way of assessing the performance of pixel classification is through segmentation of previously annotated images. There are a few performance parameters that one can use for measuring performance of the pixel classifier. The first one is the error rate, which is the sum of the number of times a pixel in the background is misclassified (false positives) and the number of times a pixel on the object is misclassified (false negatives). Another performance measure we are interested in is the classification time. Note that SVM classification times can be extremely large, depending on the number of support vectors. Although there are in the literature SVM algorithms that can minimizes the problem of classification computational cost [Burges and Schölkopf, 1997], in the experiments reported here a standard SVM implementation was used, therefore, there is no classification speed maximization whatsoever. Finally, the most effective way of determining a classifier is efficiency is through visual inspection of the segmentation results.

It is important to point out that the quantitative measure that is most directly related to the overall performance of the contour trackers presented in the next chapters is the false positive rate. Pixels wrongly classified as belonging to the object are the ones that most easily distract the contour tracker. False negatives, which are perceived as holes in the object in segmentation experiments, are likely not to damage contour tracking unless they come in large groups.

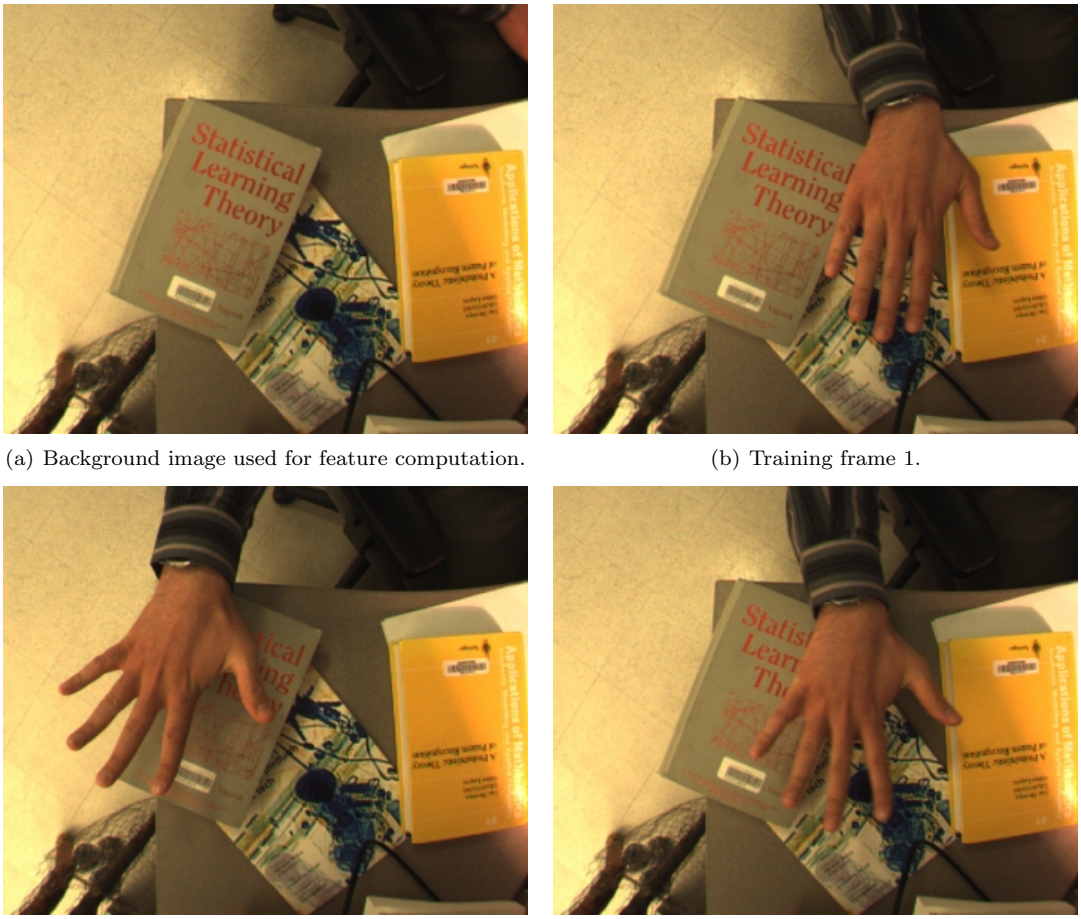
In the following, experiments designed to understand the influence on performance of the neighborhood sizes, the SVM training parameter C (3.25), and other factors are reported. The basic setting for most of the experiments is:

- A single background image (Figure 3.10(a)) for background feature computation;
- Training data is taken from the background image, and images in Figure 3.10(b) and 3.10(c). Note that only a small fraction of labeled pixels from each of those images are used for training because of memory issues since the training was done using 32-bit software. In brief, $74,348$ background and foreground pixels were used for training in a proportion of $3:1$;
- Testing was done on the $101,376$ pixels of a third image (Figure 3.10(d)).

The feature vector is computed for every pixel in all images, even though a good part of them was discarded because of memory constraints of the training software. The average time for the feature computation of one 288×352 image ($101,376$ pixels) is around 10 seconds (0.1frames/sec) in a non-optimized PYTHON implementation running in a $2.00GHz$ laptop with $2GB$ of memory.

3.9.1 Window size

One of the experiments consisted of fixing the SVM parameter C equal to 10^3, using a feature vector that integrated the pixel value, average and variance of the color in the neighborhood as well as the background subtraction features, while varying the size of the neighborhood window from a square of side 4 by increasing the side by steps of two units until reaching 22×22 windows. The time for feature vector computation increases linearly with window size (going from around 8 to approx. 20 sec/frame). The most critical time measurement, however, is the time spent on classification, which depends on the number of support vectors of the trained machine. The number of support vectors, on its turn, is related to the complexity of the classification problem and the capacity of the machine. For a larger neighborhood window the information contained in the average pixel value and the variance (or covariance) becomes more abstract by decreasing



(a) Background image used for feature computation.

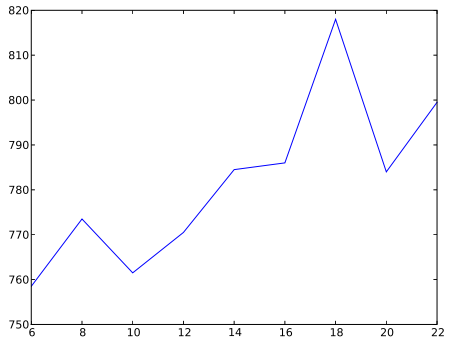
(b) Training frame 1.

(c) Training frame 2.

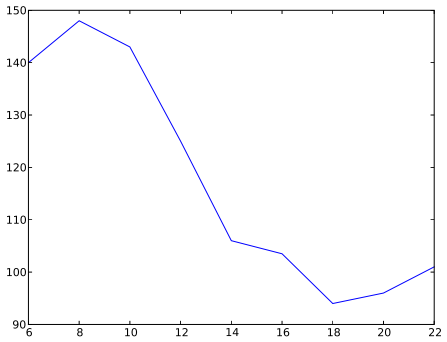
(d) Test image.

Figure 3.10: Images used in the experiments.

the amount of intra-class variability and smoothening the classification boundaries, consequently, a smaller number of support vectors are needed in the trained machine.



(a) False positives count versus window size.



(b) False negatives count versus window size.

Figure 3.11: Window size influence on error rates. Every point in the plots is the arithmetic mean between the error obtained using variance and the one using covariance features.

Regarding classification accuracy, although a window too small induces a more noisy training set, a window too big loses its ability to express useful details of the object becoming irrelevant. This can be verified in the plots in Figure 3.11. The false positives plot shows a decline in performance when windows start to grow larger than 16×16 which indicates that average and variance (and covariance) in larger windows become irrelevant in distinguishing hand details from the background. Note that an opposite behavior is observed in the false negatives plot in the range of window sizes studied. This can be explained by the fact that the regions in the background have larger homogeneous surfaces that are accurately described by the first two moments for windows larger than it is appropriate for the hand.

The images in Figure 3.12 allow for a visual inspection of the influence of the window size in the quality of the classifier. Notice the false positives which are identified as red spots outside the hand, especially the ones on the center of the figure; from the figure it is possible to deduce that a window of size around 16×16 yields a better performance.

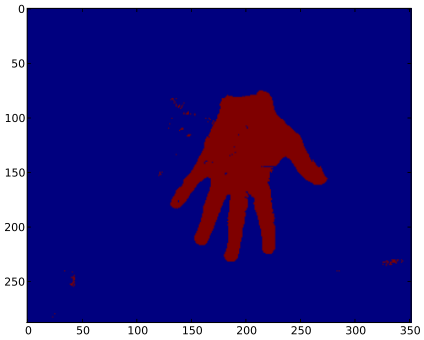
3.9.2 Training-error penalty parameter

In order to find the influence of the C training parameter, an experiment similar to the one above was carried out. In this case, however, the size of the window was fixed at 16×16, and the C parameter assumed the values $10, 10^2, 10^3, ..., 10^6$. C has a strong influence on both training time and testing error as can be seen in the image in Figure 3.15. The plots in Figure 3.13 synthesize the output.

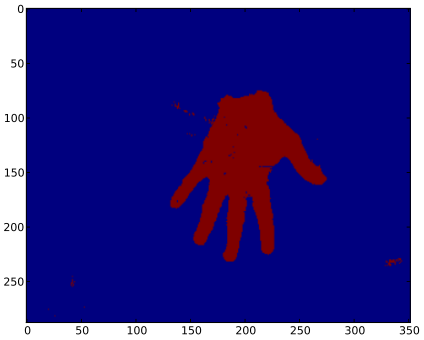
Larger Cs force smaller empirical risk (error on the training data) that can result in poor generalization on the testing data, which was not the case in this experiment. Arbitrarily large values of C, however, may incur in numerical difficulties and unviable training times.

C showed, as well, a strong influence on the number of support vectors of the trained machine, as is revealed by the plots in Figure 3.14. The plots shows a decrease in the time for pixel classification as C increases and this means a smaller number of support vectors is being utilized. Note that time measurements performed in this section are not very reliable because some experiments were run simultaneously, therefore sharing machine resources at times.

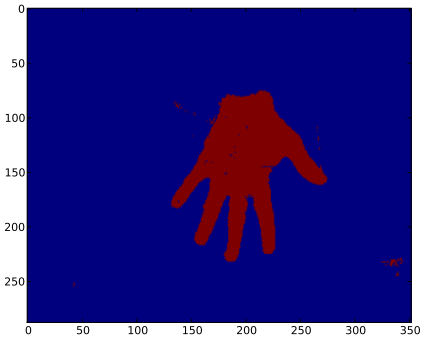
Notice that, the penalty-parameter is represented by a capital c in this chapter. In other chapters, C will represent the contour point set, unless stated otherwise.



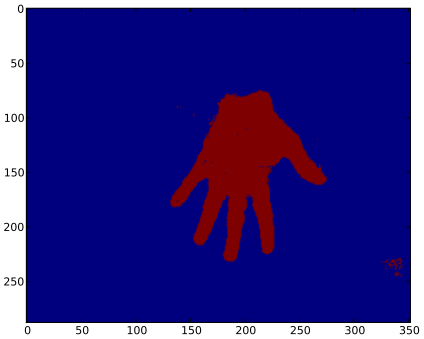
(a) 6×6 window.



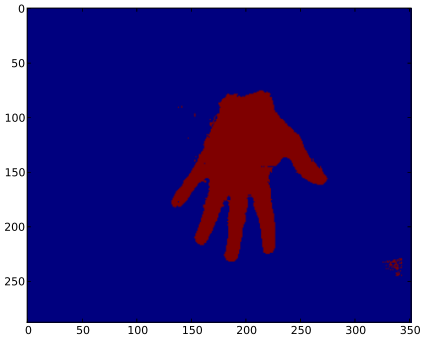
(b) 10×10 window.



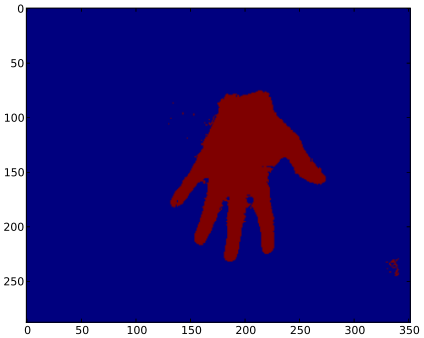
(c) 12×12 window.



(d) 16×16 window.

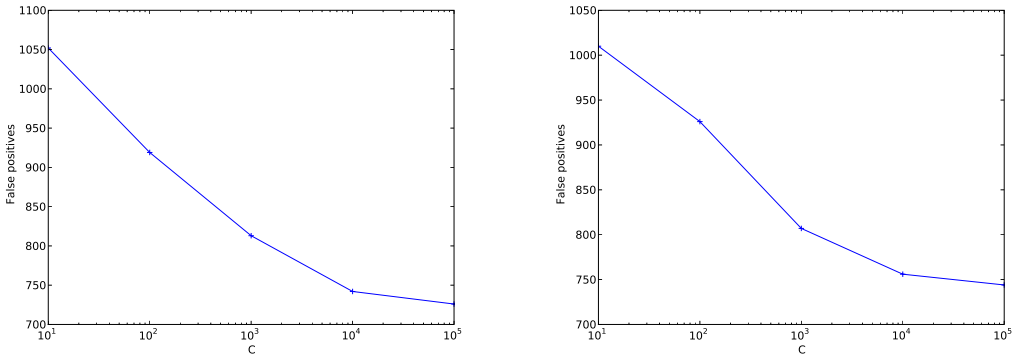


(e) 18×18 window.



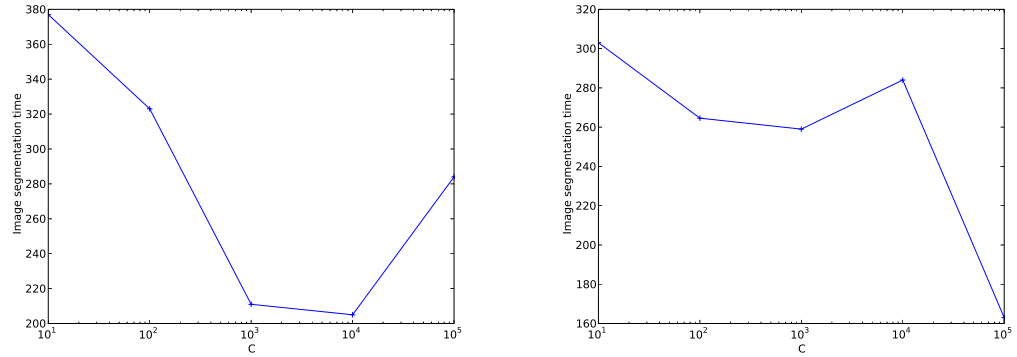
(f) 22×22 window.

Figure 3.12: Effect of neighborhood size on classification robustness. C fixed equal to 10^3.



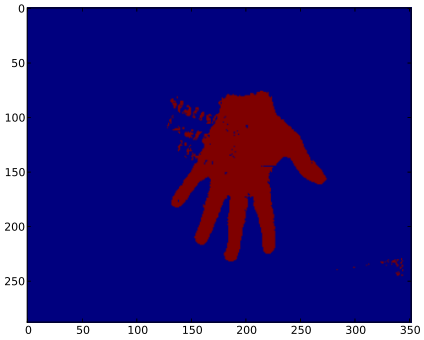
(a) 6×6 window (b) 16×16 window

Figure 3.13: False positive versus C

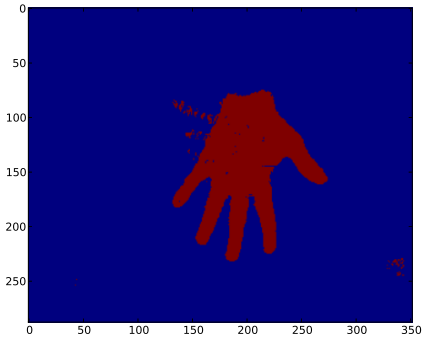


(a) 6×6 window (b) 16×16 window

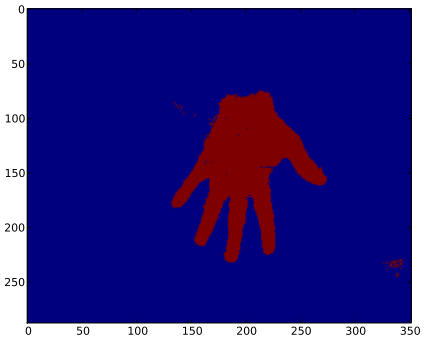
Figure 3.14: Time used for the classification of all pixels in an image versus C



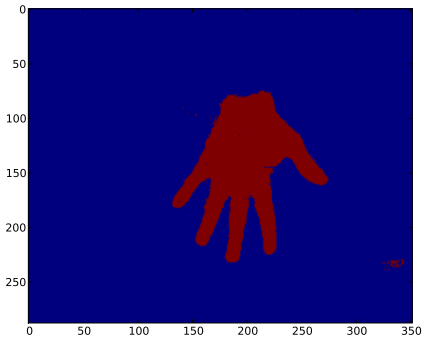
(a) $C = 10$.



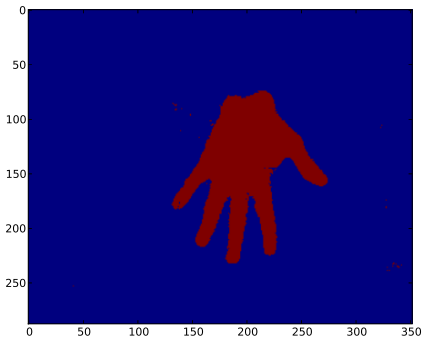
(b) $C = 10^2$.



(c) $C = 10^3$.



(d) $C = 10^4$.



(e) $C = 10^5$.

Figure 3.15: Effect of the penalty parameter on classification. Using a 16×16 fixed window.

3.9.3 Covariance features

Covariance features introduce a *new* dimension to the feature vector. However, covariance features carry all information contained in the variance, since the variance is the main diagonal of the covariance matrix , furthermore, the non diagonal elements of the covariance matrix carry information about the correlation between color components (red and blue, red and green, and green and blue) and this is potentially a good discriminating feature in the presence of varying illumination as it is revealed by the discussion on the Mahalanobis distance in an earlier section.

The images in Figure 3.16 can be visually compared with the corresponding ones in Figure 3.12, and some improvement can be noticed. Quantitatively, the use of covariance features represented an average improvement of 3.73% in relation to variance features, when only false positive error rates are considered. The overall improvement (on the total error rate which is the sum of the positives and negative errors) due to the covariance features was of 3.7%.

3.9.4 Background features

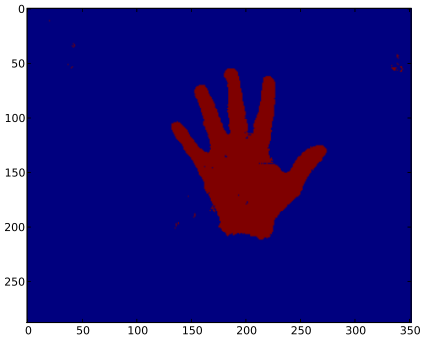
An experiment was carried out to find out about the practical usefulness of the background subtraction features. Basically, the standard experiment was run once, using now the reduced feature vector (without background subtraction features). C was fixed equal to 10^3 and window dimensions equal to 16×16. Both variance covariance features were tested, resulting in the segmentations in Figures 3.17(b) and 3.17(d). Those results can be compared against 3.17(a) and 3.17(c).

Numerically, the removal of the background subtraction features were responsible for an average increase in false positive error of 7.85%. The overall increase in error was of 8.77%.

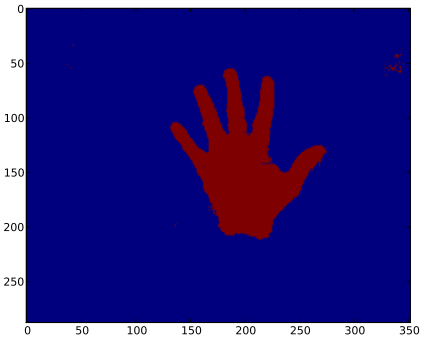
3.9.5 A 3D-grid search

Up to this point, the parameters C and the window size have been investigated separately. Moreover, the parameter γ, which is associated with the RBF kernel function, has not been studied yet. An experiment was carried out to help understand the joint influence of these parameters in the classification performance. The experiment consisted of going through most of the nodes in the 3D grid ($C \times \gamma \times w$), where the w's are window sizes, training and assessing a SVM. C assumed the values in $\{2^3, 2^4, \ldots, 2^9\}$, $\gamma \in \{2^{-15}, 2^{-14}, \ldots, 2^0\}$ and $w \in \{6, 8, 10, 12, 14, 16, 18, 20\}$. This means that there was something close to 800 parameter configurations to train and test a SVM. Even on a powerful machine with 8 Intel Xeons dual core CPUs running at 3.00GHz and with 32GB of RAM memory, the whole experiment took several days to complete. Again, a single image was used for training and a second one for testing. The feature vector utilized included background and covariance features.

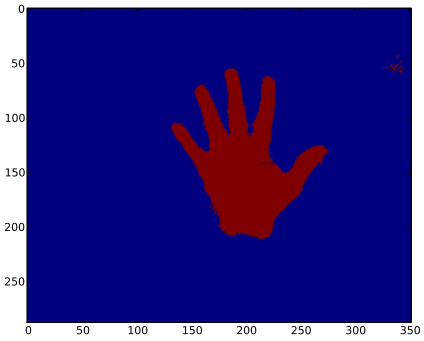
The visualization of the results of this experiment (Figures 3.18 and 3.19) helps to understand the relative influence of each of the parameters in the classification performance. For instance, from the figures it was confirmed that larger values of the penalty parameter are usually preferable. The other important deduction is that the window size did not demonstrated to be a sensitive parameter, in fact, error variation with window size is even difficult to observe in the figures. Nonetheless, the experiment showed that results are extremely sensitive to the choice of γ.



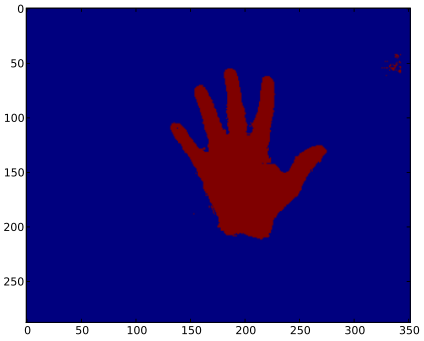
(a) 12×12 window.



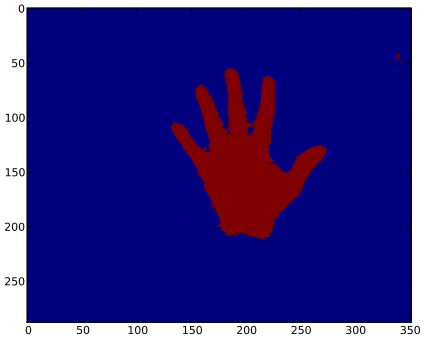
(b) 14×14 window.



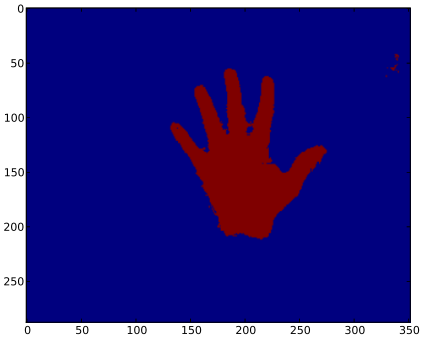
(c) 16×16 window.



(d) 18×18 window.

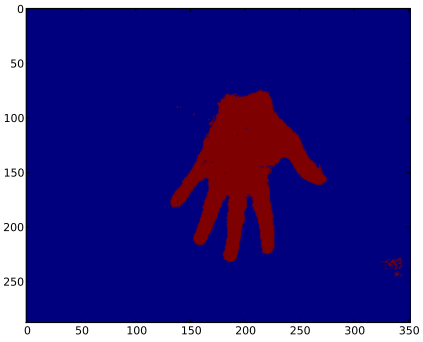


(e) 20×20 window.

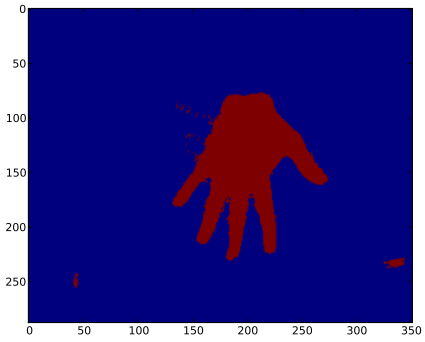


(f) 22×22 window.

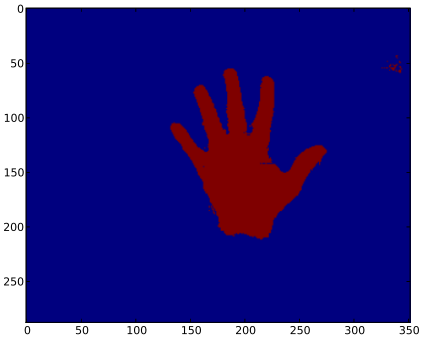
Figure 3.16: Effect of covariance features for various windows sizes. C fixed equal to 10^3.



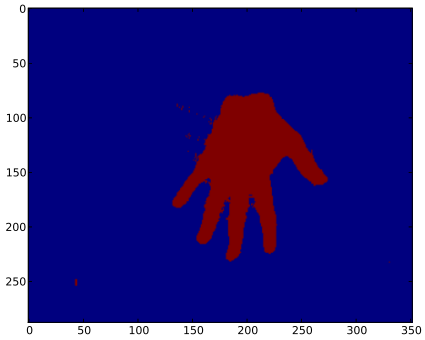
(a) Using background subtraction and variance features.



(b) Using variance features but not using background subtraction features.



(c) Using background subtraction and covariance features.



(d) Using covariance features but not using background subtraction features.

Figure 3.17: Effect of background subtraction features. C fixed equal to 10^3 and window dimensions equal to 16×16.

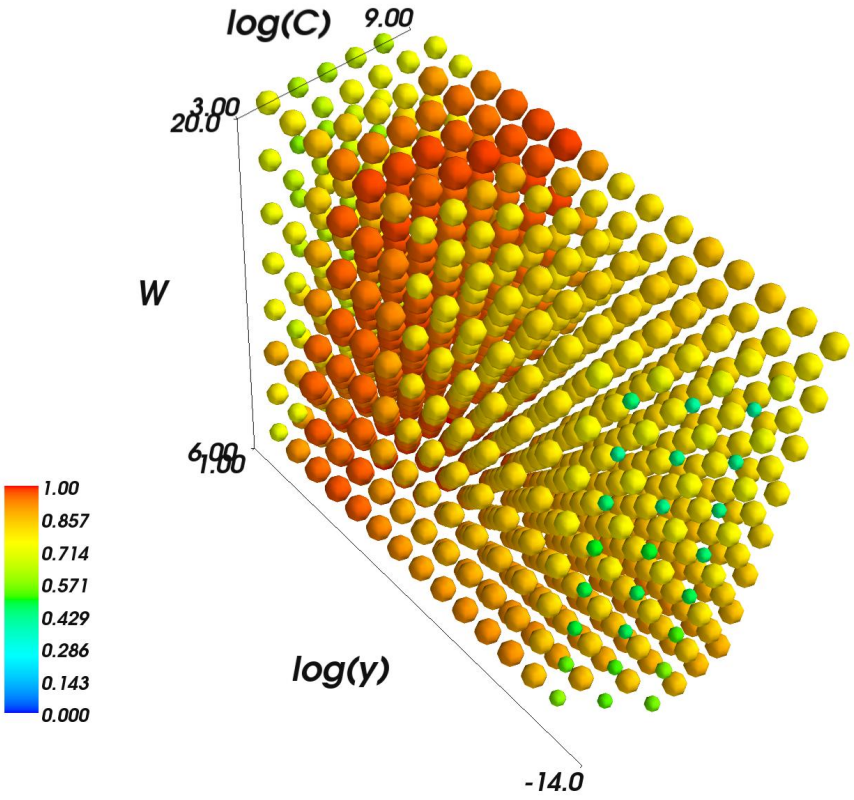


Figure 3.18: Total error (false negatives plus false positives). The error data was rescaled so that the smaller errors are represented by the bigger red spheres and as the error increases the sphere gets smaller and its color shifts towards the blue.

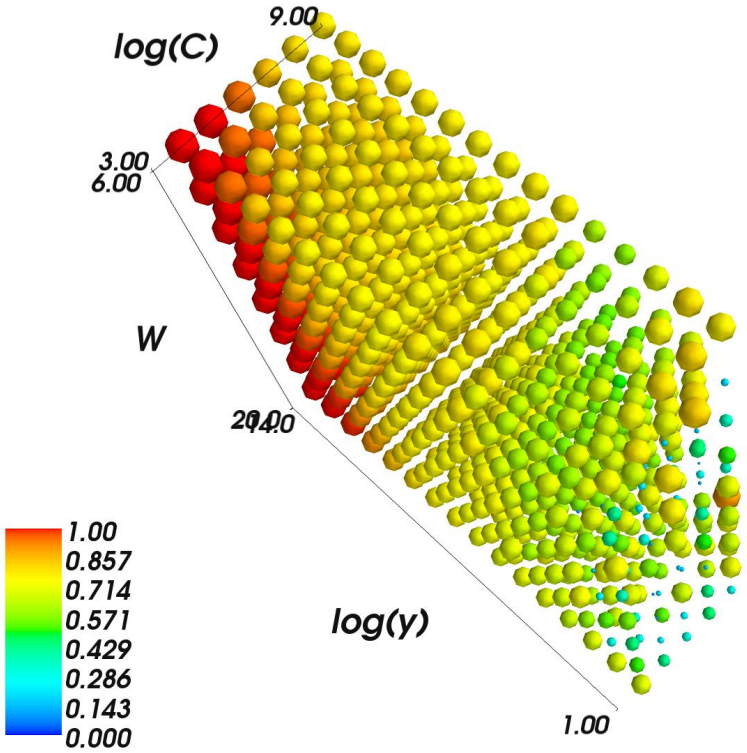


Figure 3.19: False positives. The error data was rescaled so that the smaller errors are represented by the bigger red spheres and as the error increases the sphere gets smaller and its color shifts towards the blue.

Chapter 4

Level Set Method for Contour Tracking

In this chapter, the concepts of implicit interface representation and level set methods for contour tracking are introduced (the interested reader is referred to Cremers et al. [2007], Giraldi et al. [2007], Cremers [2006], Osher and Fedkiw [2003], Osher and Paragios [2003], Paragios and Deriche [2000] and Osher and Sethian [1988a] and references therein for more on the subject of implicit representation and level-set methods). In the sequence, an algorithm that approximates the behavior of the standard level set method approach, however achieving much higher execution speeds is described [Shi and Karl, 2008]. Finally, the pixel-level classification tool developed in the previous chapter is combined with Shi&Karl's algorithm to produce a robust hand-contour tracking. Some experiments and analyses of the resulting method are presented in the final part of the chapter.

4.1 Implicit curve representation

The *Jordan curve theorem* (see `http://e-learn.mine.nu/mizar/mml/jordan.miz`) in topology, using around 6500 lines of reasoning, proves that a non-self intersecting loop (a closed simple curve) divides the plane into two connected components a bounded and an unbounded or, in other words: an "inside" and an "outside"(see Figure 4.1). The curve that bounds object silhouettes, a.k.a. contours, are compositions of closed and non self intersecting curves, moreover, a contour is constrained to the image rectangle $D \subset \Re^2$. Formally, the contour curve C is the interface between the object region, $\Omega \subset D$, and the background of the image, $D \backslash \Omega$.

As an illustration of this consider the set of points $\Omega = \{\vec{x} \text{ s.t. } |\vec{x}| < 1\}$ where $\vec{x} = (x, y)$, that is, the open disk with radius r equal to one, centered at the origin of the $2D$ Cartesian plane. The *explicit* contour representation of Ω can be given in a very compact form,

$$C = \{\vec{x} \text{ s.t. } |\vec{x}| = 1\}.$$

When it comes to real-world objects, however, a straightforward analytical expression like the one above is seldom feasible. A common approach to explicit contour representation is based on approximations of C by a linear combination of polynomial curves of a parameter s, defined in the one-dimensional interval $[0, L]$ (B-splines for example). The most outstanding advantage of such explicit contour representations is the ease to compute points on the contour, which is basically a matter of polynomial evaluations for different values of s in $[0, L]$.

The basic idea of *implicit* contour representations is to define a surface ϕ that embeds C as its zeroth-level isocontour or zeroth-level set. Returning to the disc example, we have that C can also be given by the zeroth-level set (isocontour) of the surface given by

$$\begin{aligned} &\phi : \Re^2 \mapsto \Re \\ &\vec{x} \longmapsto \phi(\vec{x}) = x^2 + y^2 - 1. \end{aligned} \tag{4.1}$$

(a) Simple curve (no self intersections). Defines an inside and an outside clearly.

(b) Self intersecting curve. Where is *in* and *out* in this figure?

Figure 4.1: Examples of a simple and self-intersecting curve.

Being brief: C is given by the points $\vec{x} \in D$, such that $\phi(\vec{x}) = 0$. We say that the contour is *implicitly* represented in ϕ. Again, when it comes to real-world object contours, finding analytical expressions to define the surface ϕ is unviable. Moreover, the discretization and approximation of ϕ has to be done in a 2D grid, as opposed to the 1D interval needed in the explicit representation framework, which makes the task even harder. Then, one might ask, why bother with implicit representations? One important advantage of implicit representations is that it copes with substantial topological changes like merging, splitting, appearance and vanishing of holes in a natural way. These topological changes are impossible to handle using the known explicit representation tool set. Using implicit representation it is trivial to tell if a point is inside, outside or on the interface by evaluating if $\phi(\vec{x}) < 0, \phi(\vec{x}) > 0$ or $\phi(\vec{x}) = 0$, respectively. This establishes a clear contrast with the explicit framework:

- Explicit representation: Easy to obtain points in the contour (they are explicit). However, given a point it is not straightforward to determine what is its relation to the object region point set.
- Implicit representation: Difficult to obtain points in the contour (need to solve the equation $\phi(\vec{x}) = 0$). Notwithstanding, given a point it is trivial to tell its relation to the object.

Added to that, implicit interface representation makes simple Boolean operations, as well as constructive solid geometry (CSG) operations, easy to apply. For example, if ϕ_1 and ϕ_2 are two different implicit functions, then $\phi(\vec{x}) = \min(\phi_1(\vec{x}), \phi_2(\vec{x}))$ is the implicit function representing the union of the interior regions of ϕ_1 and ϕ_2. In the same fashion, $\phi(\vec{x}) = \max(\phi_1(\vec{x}), \phi_2(\vec{x}))$ is the intersection of the interior regions of ϕ_1 and ϕ_2. The complement of $\phi_1(\vec{x})$ can be defined by $\phi(\vec{x}) = -\phi_1(\vec{x})$. Also, $\phi(\vec{x}) = \max(\phi_1(\vec{x}), -\phi_2(\vec{x}))$ represents the region obtained by subtracting the interior of ϕ_2 from the interior of ϕ_1.

Perhaps, the most desirable feature of an implicit representation, compared to the explicit one, is the fact that it does not need a re-parameterization, mechanism or a metric that is invariant to re-parameterization which is a burden associated with explicit representations only.

An important property of ϕ in the level-set formulation is the gradient:

$$\nabla\phi = \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}\right). \tag{4.2}$$

which is perpendicular to the isocontours of ϕ and points in the direction of increasing ϕ. Therefore, the outward normal to the contour can be determined by

$$\vec{N} = \frac{\nabla\phi}{|\nabla\phi|} \tag{4.3}$$

at every contour point. Notice, that the normal is defined in the entire domain D and not only on the contour.

The partial derivatives in (4.2) can be approximated via finite difference schemes on a 2D grid obtained by discretizing D. For example,

$$\frac{\partial\phi}{\partial x} \approx \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x} \tag{4.4}$$

is the well known forward difference scheme which approximates the partial derivative with an error proportional to the grid spacing (resolution) Δx, while,

$$\frac{\partial\phi}{\partial x} \approx \frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x} \tag{4.5}$$

is the backward presenting an error of order $O(\Delta x)$. The central difference scheme

$$\frac{\partial \phi}{\partial x} \approx \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \tag{4.6}$$

on the other hand, presents an error of order $O(\Delta x^2)$.

Implicit contour representation plays a role in interface propagation analogous to what the *kernel trick* does in machine learning, in the sense that it provides a much more expressive formulation at the expense of a reasonable increase in computational cost.

4.2 Level set methods

The numerical methods known as level set methods were originally proposed by Osher and Sethian [Osher and Sethian, 1988a] as a means to simulating propagating fronts (interfaces) based on their implicit representations. The applications of these methods are diverse, going from prediction of fire spreading, through rendering of realistic 3D scenes of water, fire, etc, to object contour tracking in video.

Level set methods deal with the evolution of C (and ϕ, consequently) in time, therefore, the notation $C(t)$ (and $\phi(\vec{x}, t)$), where t represents the time variable, will be used hereafter. Moreover,

$$\phi(C(t), t) = 0, \forall t \in \Re^+$$

results directly from the implicit representation definition. Taking the total time derivative of the equation above, we obtain

$$\frac{d\phi}{dt}(C(t), t) = \nabla\phi \cdot \vec{V} + \frac{\partial \phi}{\partial t} = 0 \tag{4.7}$$

where

$$\vec{V} = \frac{\partial C}{\partial t}(t)$$

is a velocity field defined on the contour points that actually defines their evolution. This partial differential equation (PDE) is universally called a convection or *advection* equation.

Some assumptions are made on $\vec{V}$:

i. In equation (4.7), $\vec{V}$ is considered to be defined not only on the interface points but its definition is extended to the whole domain D, or at least to an ϵ-wide band around C. This is necessary to allow stable numerical implementations.

ii. The component of $\vec{V}$ tangential to C has negligible importance to the determination of the actual placement of the contour. Moreover, only the velocity (and not its variation) provides useful information for the evolution of C at any instant t.

iii. The higher the variation of $\vec{V}$ in D, especially around C, the smaller Δx needs to be so as to assure numerical stability. Thus, it is necessary that the modeling of $\vec{V}$ be done in a way that it minimizes the variation of $\vec{V}$, in order to make the sampling manageable.

A comprehensive discussion on these assumptions can be found in chapter 3 of the book of Osher and Fedkiw [2003]. The first assumption states the need of an extension of the definition of $\vec{V}$ to points outside the contour. Assumptions ii. imply that the definition of $\vec{V}$ in points on the contour be constant and normal to the interface. They also induce that, on points outside the contour, $\vec{V}(\vec{x})$ should be set to $\vec{V}(\vec{x}) = \vec{V}(\vec{x}_C)$ where $\vec{x}_C$ is the point on C closest to $\vec{x}$. Therefore, the best choice for $\vec{V}$ is

$$\vec{V}(\vec{x}) = F\vec{N}(\vec{x}) \tag{4.8}$$

where $\vec{N}(\vec{x})$ is the unit normal to the contour given by (4.3), if $\vec{x} \in C$, or it is calculated using

the extension rule mentioned above. F is a scalar and is determined by the image data and regularization information (internal constraints).

Standard level set methods are obtained through the application of finite difference schemes like the ones in (4.4), (4.5) and (4.6).

Contour tracking with level set methods is a matter of providing a velocity field $\vec{V}_k$ that drives the contour to the object border at the video frame k. For $k = 1$, a standard initialization of C is used, thereafter, the final resulting contour of one frame is used as initialization for the next one.

4.3 A fast level set approximation algorithm

Notice that contour extraction with level sets is based on the iterative solution of a PDE (advection equation) on at least a certain band around C for every video frame. This can be rather slow. Recently, Shi and Karl [2005a] proposed a modified level set algorithm that is extremely fast. Their algorithm basically retains the key advantages of the standard level-set approach and achieves similar results, however, the mechanism for contour evolution is significantly changed. By careful construction they built an algorithm that is mainly based on integer and linked-list operations which are very easy to program and translates into extremely low computational complexity.

Shi&Karl made the clever observation that, for many engineering applications, image segmentation for instance, the information of how the contour evolves is meaningless and the resulting contour C at the final iterative step (the segmentation) is what actually matters. Furthermore, in traditional PDE-based level set, a contour point can be displaced distances smaller than the actual minimal pixel spacing (we say it operates on sub-pixel scale), in most practical cases is a waste of CPU-cycles since, in the end, the contour point is necessarily snatched to a valid pixel location on the image grid, so that it can be displayed.

The algorithm replaces the numerical solution of the advection PDE with an evolution mechanism that involves simple linked-lists operations, in which interface points are always displaced by at least a pixel. The basic setting of the algorithm starts with the definition of two linked-lists, L_{in} and L_{out}, which store the points of the narrow-band around C in the Ω and $D\backslash\Omega$ regions, respectively. The lists are formally defined in the following way:

$$L_{in} = \{\vec{x} | \vec{x} \in \Omega \text{ and } \exists \vec{y} \in N(\vec{x}) \text{ such that } \vec{y} \in D\backslash\Omega\} \tag{4.9}$$

$$L_{out} = \{\vec{x} | \vec{x} \in D\backslash\Omega \text{ and } \exists \vec{y} \in N(\vec{x}) \text{ such that } \vec{y} \in \Omega\} \tag{4.10}$$

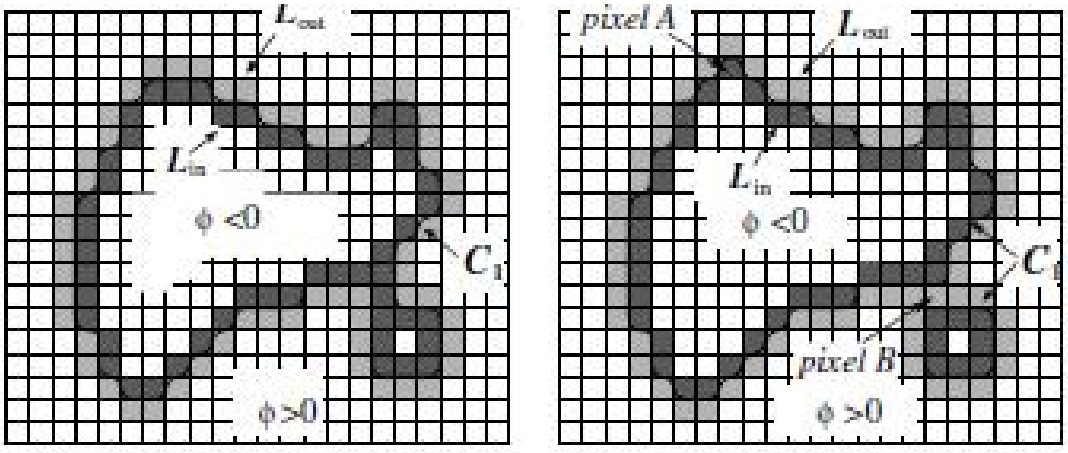
where, given $\vec{x} \in D$, $N(\vec{x}) = \{\vec{y} \in D \text{ s.t. } |\vec{x} - \vec{y}| = 1\}$ is called the neighborhood of $\vec{x}$. Notice that the domain D is considered hereafter as a 2D grid, and every point $\vec{x}$ in the domain now is considered to be a pair of integers (indices).

Based on the definition of the narrow-band lists above, ϕ is then defined on D as follows:

$$\phi(\vec{x}) = \begin{cases} 3, & \text{if } \vec{x} \text{ is an exterior point} \\ 1, & \text{if } \vec{x} \in L_{out} \\ -1, & \text{if } \vec{x} \in L_{in} \\ -3, & \text{if } \vec{x} \text{ is an interior point} \end{cases} \tag{4.11}$$

Notice that, based on this definition, ϕ is represented in the computer as a 2D matrix of integers. Only the sign of F in (4.8) is actually used in the algorithm, therefore, it can be represented in the computer as a 2D matrix of integers as was done with ϕ. The two lists L_{in} and L_{out} are bidirectionally linked, so elements can be added and removed easily (Figure 4.2).

Two basic procedures on the narrow-band lists determine the contour evolution: $switch\_in(\vec{x})$ and $switch\_out(\vec{x})$. Let the point $\vec{x}$ be in the exterior part of the narrow band (L_{out}), the $switch\_in(\vec{x})$ procedure effectively moves the contour outward by a pixel by including $\vec{x}$ in the



(a) The implicit representation of the curve C_1 and the two lists L_{in} and L_{out} in the neighborhood of C_1

(b) Illustration of the motion of the curve C_1 by switching pixels between L_{in} and L_{out}.

Figure 4.2: Level-set-like contour evolution (from Shi and Karl [2005a])

object region (in L_{in}). After removing $\vec{x}$ from L_{out} and inserting it into L_{in}, all neighbors of $\vec{x}$ are tested if they are in the exterior (if $\phi(\vec{y}) = 3$) and if, say, that is the case for $\vec{y}$, then $\vec{y}$ is added to L_{out} becoming a neighboring point ($\phi(\vec{y})$ is set to 1). Similarly, for a point $\vec{x}$ in L_{in}, $switch\_out(\vec{x})$ moves the contour inward at $\vec{x}$ by removing it from L_{in} and adding it to L_{out}, and then testing if any neighbor of $\vec{x}$ is in the interior (if $\phi(\vec{y}) = -3$) and, if that is the case, adding it to L_{in} and setting $\phi(\vec{y})$ to -1.

The algorithm itself is decomposed in two main cycles. In fact, the first cycle (data-driven) is the one that actually segments the frame and, consequently, tracks the contour. The second cycle, as will be seen, improves the result in a smoothing operation.

4.3.1 The data-driven cycle

The data-driven cycle is summarized in Algorithm 1. At every iteration the speed F is computed at every element of L_{in} and L_{out} and its sign is stored in the array F. The next step is to go through all points in L_{out} and, for all points for which $F > 0$, the $switch\_in(\cdot)$ is applied. This scan finds the places in the interface with positive speed and moves them outward by one grid point. After this, L_{in} has to be scanned and some elements may have to be deleted in order to assure that L_{in} is in accordance with its definition.

Similarly, L_{in} is then scanned and, for all points for which $F < 0$, the $switch\_out(\cdot)$ is applied. This scan finds the places in the interface with negative speed and moves them inward by one grid point. After this L_{out} has to be scanned for redundant elements that needs to be deleted so its definition is respected.

The following step is to test the following stopping condition: If either

(a) The speed at all the neighboring grid points satisfies

$$\begin{aligned} F(\vec{x}) &\leq \ \forall \vec{x} \in L_{out} \\ F(\vec{x}) &\geq \ \forall \vec{x} \in L_{in} \end{aligned} \tag{4.12}$$

(b) A pre-specified maximum number of iterations is reached.

is satisfied than the following video frame is loaded and the process re-initiates. The algorithm was reported to be faster than the efficient narrow band sparse-field level-set algorithm [Withtaker, 1998] by an order of magnitude on average while achieving similar results.

4.3.2 The regularization cycle

The algorithm presented in the previous section is capable itself of tracking the hand contour, however, a boundary regularization cycle can be added to it in order to smoothen the final result. Regularization usually consists of implementing a speed function that is the combination of the data-driven (or external) term F_{ext} and an intrinsic speed F_{int} term, chosen proportional to boundary curvature. The curvature-based component of the speed is calculated from the Laplacian of ϕ. Shi and Karl pointed out that such a curvature-based smoothing introduces floating point (non-discrete) operations that may slow computational speed. On the other hand, they also realized that evolving a function according to its Laplacian is equivalent to Gaussian smoothing by studying the heat PDE. Based on this, they proposed a separate cycle that carries out the Gaussian filtering of the discrete-valued $\phi(\vec{x})$ and then updates the lists L_{out} and L_{in} based on the sign of the filtered result.

Let G denote the Gaussian kernel of dimension $N_g \times N_g$ (the Gaussian bell evaluated on the a grid of dimension $N_g \times N_g$), one complete regularization cycle consists of going through all boundary points (L_{in} and L_{out}), and

- For every point $\vec{x} \in L_{out}$, $G \otimes \phi(\vec{x}) < 0, switch\_in(\vec{x})$;

Algorithm 1 Shi & Karl algorithm (data-driven cycle).

```
Initialize φ, F_d, L_in, L_out
for i = 1 to N_a do
  Compute F_d(x) ∀x ∈ L_in, L_out
  for all x ∈ L_out do
    if F_d(x) > 0 then
      switch_in(x)
    end if
  end for
  for all x ∈ L_in do
    if ∀y ∈ N(x), φ(x) < 0 then
      delete x from L_in
      φ(x) = −3
    end if
  end for
  for all x ∈ L_in do
    if F_d(x) > 0 then
      switch_out(x)
    end if
  end for
  for all x ∈ L_out do
    if ∀y ∈ N(x), φ(x) < 0 then
      delete x from L_out
      φ(x) = 3
    end if
  end for
  if stopping condition then
    break loop
  end if
end for
```

- Deal with redundancy created in L_{in} by going through all of its points checking neighborhoods and deleting internal elements;
- For every point $\vec{x} \in L_{in}$, $G \otimes \phi(\vec{x}) > 0, switch\_out(\vec{x})$;
- Deal with redundancy created in L_{out} by going through all of its points checking neighborhoods and deleting external elements.

The algorithm for regularization of the boundary is synthesized in Algorithm 2. Algorithms 1 and 2 can then be combined by alternating between them by first iterating the data-driven cycle N_a times (or until the stop condition), and then apply N_g iterations of the smoothing cycle. The relative weight of these two effects can be controlled by the pair (N_a, N_g). Note that N_g determine the size of the Gaussian kernel as well and it controls the elimination of small "holes" with small radii. In our experiments, we have adopted $N_a \approx 8$ and $N_g \approx 3$.

Algorithm 2 Shi & Karl algorithm (regularization cycle).

```
Compute F_int(x) ∀x ∈ L_in, L_out
for i = 1 to N_g do
  for all x ∈ L_out do
    if F_int(x) > 0 then
      switch_in(x)
    end if
  end for
  for all x ∈ L_in do
    if ∀y ∈ N(x), φ(x) < 0 then
      delete x from L_in
      φ(x) = −3
    end if
  end for
  for all x ∈ L_in do
    if F_int(x) > 0 then
      switch_out(x)
    end if
  end for
  for all x ∈ L_out do
    if ∀y ∈ N(x), φ(x) < 0 then
      delete x from L_out
      φ(x) = 3
    end if
  end for
end for
```

4.4 Contour tracking experiments

The accuracy of the level-set evolution depends on the speed function F that governs the model deformation. In the classical approach, level set speed function depends on the image gradient, ∇I. For example, Geodesic Active Contours [Paragios and Deriche, 2000].

$$F = g(|\nabla I|)(\kappa + \nu). \tag{4.13}$$

where $g(|\nabla I|)$ is commonly defined as

$$g(|\nabla I|) = \frac{1}{1 + |\nabla G_\sigma * I|^p} \tag{4.14}$$

with $p = 1$ or $p = 2$. G_σ denotes a Gaussian convolution filter with standard deviation σ.

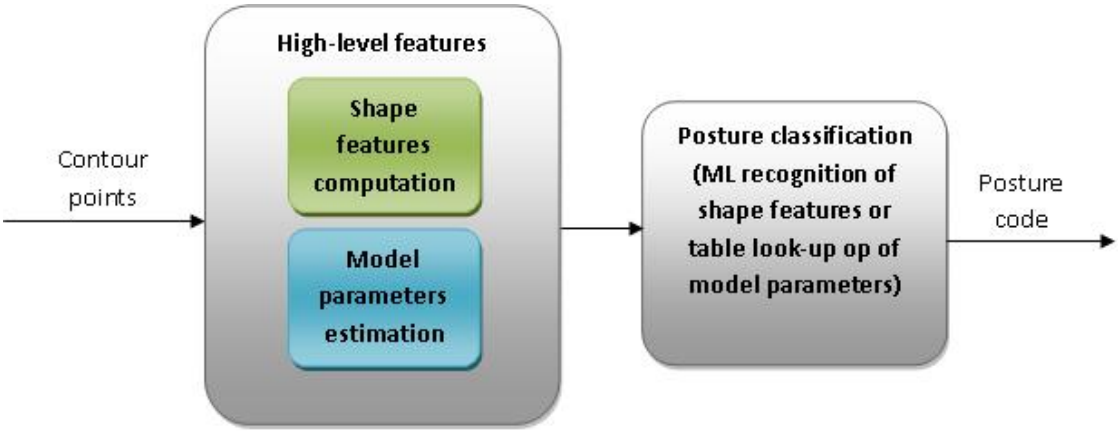


Figure 4.3: Posture Classification Information Flow

The drawback of such approach is that in situations of cluttered background is very difficult to guarantee general properties on the image gradient, or other similar measure alike, which are only valid for the object and not to background.

The approach proposed here is to use the output of the pixel classifier developed in the previous chapter to define the speed function F, that is,

$$F(x) = SVM(x) \tag{4.15}$$

where $SVM(\cdot)$ is the output of a support vector machine (3.22) trained to classify the pixels of the target object from the background (returns 1 if it predicts an object pixel and -1 otherwise), therefore, a more robust behavior under background clutter is expected. The following subsections report some empirical results obtained through this approach.

4.4.1 Initialization

A note is due on the initialization. In practice, level-set methods like the one presented here are very robust to initialization. Basically, a sufficiently good initialization has to have $L_{in} \neq \emptyset$. Figure 4.4 shows the overall initialization setting (Φ, L_{in}, L_{out}) used for the level-set tracker experiments in this chapter.

4.4.2 The "hall monitor" sequence

This experiment is not directly linked with gesture recognition, however, it poses a challenging situation for the contour tracker presented here. The background features strong clutter in colors close to the one of the object. The target object -the walking man- has small details difficult which are hard to model due to the small number of samples (pixels on the feet, hands, etc.) available for training. Another difficulty is that, in the sequence, the person changes appearance significantly by originally facing the wall and turning the back to the camera and moving away from it. Appearance change is an intrinsic weakness of the approach proposed in this thesis, the object model is specified to the appearance presented in the training images.

Thida et al. [2006] proposed a very successful speed function for this type of situation. Their method used an adaptive background model, background subtraction and an adaptive speed function model that can cope with blurred borders. The speed function of that work is dependent on the intensity of motion in general and not on the object specifically (that is why the person's shadow is tracked as well). Their contour evolution scheme uses the same fast level set approximation algorithm used in this thesis, therefore, their work makes a good benchmark reference for our algorithm performance. The method proposed in the thesis present a more noisy

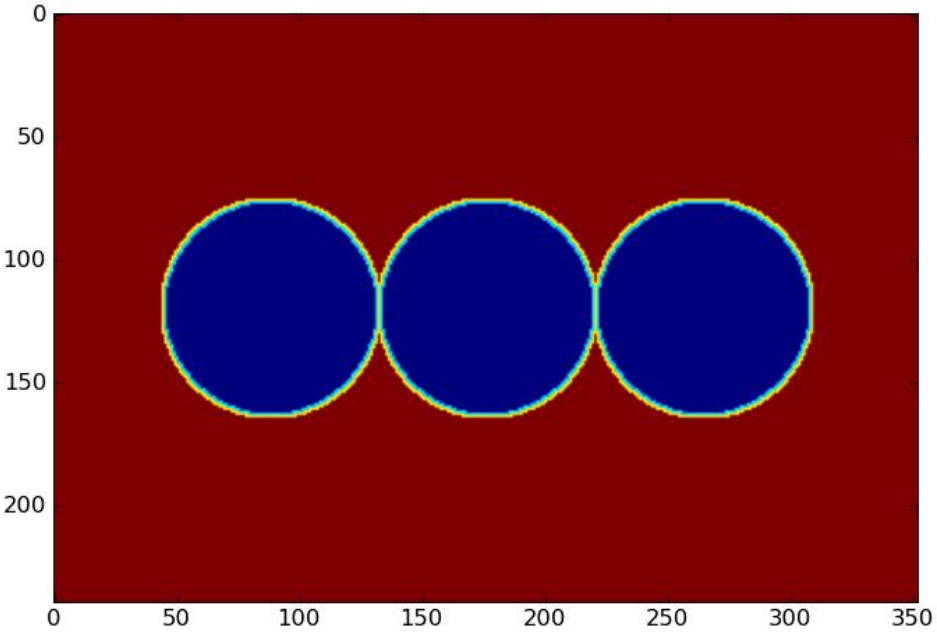


Figure 4.4: Level set initialization. Red is outside, dark blue inside region, yellow L_{out}, baby blue L_{in}.

shape, however, the smooth human outline obtained in Thida et al. [2006] is more drastically distorted by the shadow (try to visualize just the outlines on a blank canvas).

4.4.3 The rock-paper-scissors sequence

In this sequence there is negligible background noise around the object and most importantly, the appearance of the hand throughout the sequence is well represented in the training image set, a single image in fact. The result, as can be verified in Figure 4.6, is very satisfactory.

4.4.4 The 3D Navigation sequence

This sequence contains a more challenging situation in terms of illumination. The monitor, which is turned on and close to the hand, generates a varying illumination condition. Although a fairly good result was achieved (Figure 4.7), several training images were necessary opposed to a single one needed in the previous experiment.

4.5 A discussion on the algorithm's complexity

A detailed derivation for an estimate of the algorithmic complexity of Shi&Karl's algorithm can be found in Shi and Karl [2008]. In essence, the overall computational cost is $O(\mathcal{N}_A P)$, where $\mathcal{N}_A$ is the number of points of the *symmetric difference* between the set of points of the current object region and the set of object points of the initialization (or object region in the previous video frame); and P is the maximum cost of a *switch_in* or *switch_out* operation. The cost of the *switch_in* and *switch_out* in our case is basically the cost of classifying the pixel plus the cost of one remotion and one insertion on a list. The cost of pixel classification clearly dominates the total cost. The cost of pixel classification as estimated from equation (3.22) is $O(N_{SV} \cdot K)$, where N_{SV} is the number of support vectors of the machine that classifies the pixels, and K is the cost of evaluating the kernel, $k(\mathbf{x}, \mathbf{x_i})$, in particular the Gaussian kernel which was used throughout the thesis.

Finally, the computational cost estimative can be rewritten as

$$O(\mathcal{N}_A N_{SV})$$

which reveals the two dominant factors influencing the overall frame processing rate. $\mathcal{N}_A$ is related with the speed the contour changes between frames and N_{SV} is, ultimately, a measure of the object appearance complexity, which is directly proportional to illumination variations and background clutter as was seen in the experiments in the previous session.

The remarkable conclusion that can be derived from the above is that the machine-intelligence-based contour tracking architecture proposed in this thesis automatically handles variations of the problem complexity associated with different problem instances. To a certain extent, increase in problem complexity is directly translated into higher amounts of **computation** instead of the need of changes in the algorithm **structure**.

Figure 4.5: Results of the hall monitor sequence. The right column are the results obtained with the machine learning method proposed in this thesis and the left column the result obtained in Thida et al. [2006].

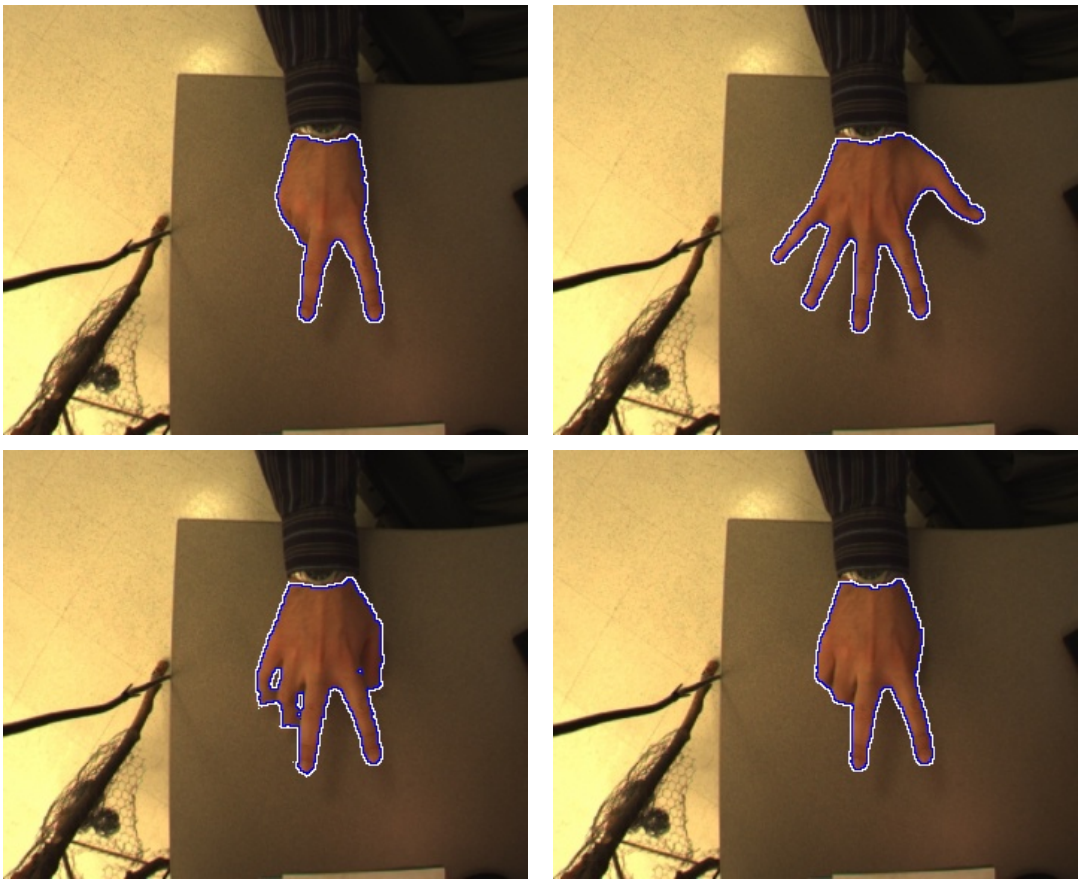


Figure 4.6: Contour tracking with an “easy” background and approximately constant appearance.

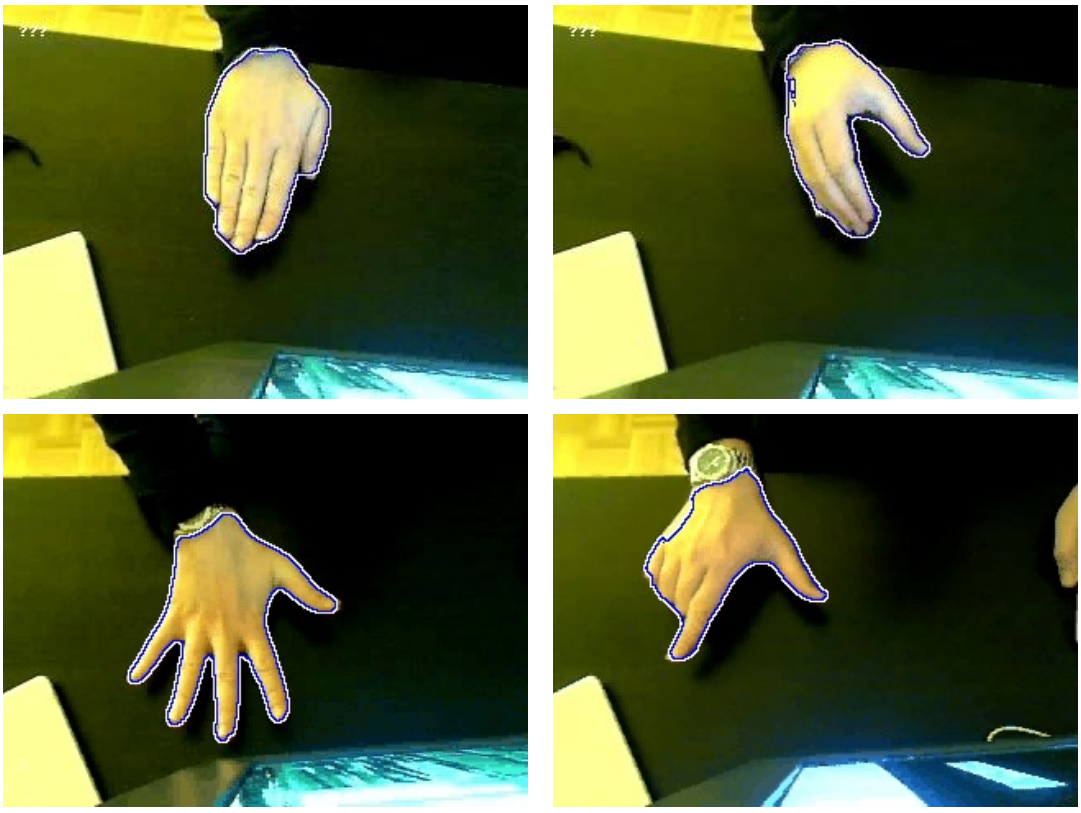


Figure 4.7: 3D Navigation sequence. Varying illumination demands training data that contains representative samples of the various illumination conditions.

Chapter 5

Posture Recognition

The computer representation of contours (as outputted from the level set method) is a finite set of points. The number of points in a contour is variable and has turned out to be between three and seven hundred points on average in preliminary experiments. As was mentioned earlier, contours will be used as intermediate (middle-level) features from which a higher level feature vector will be computed, denominated hereafter as shape feature-vector or simply as shape descriptor. The approach to posture recognition of this thesis is, therefore, training a learning machine to classify shape descriptors.

It is highly desirable that this shape descriptor be as compact (low dimensional) as it can be, and that it present invariability to Euclidian transformations. In other words, different "views" of a posture (different rotation, translation and scale configuration) should yield the same shape vector. In the thesis Hu's moment invariants are adopted precisely because of its invariance properties.

In this chapter, a posture recognition module based on contours and moments invariants measures is presented and experiments revealing its behavior in different situations are described.

5.1 Hu's moments invariants

Recall from the introduction chapter that the general formula for computing contour moments is given by:

$$m_{pq} = \sum_{(x,y)\in C} x^p y^q \tag{5.1}$$

for $p, q = 1, 2, 3 \ldots$. 2D moments computed this way are often called *geometric moments*. The contour *centroid* coordinates, $(\bar{x}, \bar{y})$, can be computed in terms of moments

$$\bar{x} = \frac{m_{10}}{m_{00}}$$

and

$$\bar{y} = \frac{m_{01}}{m_{00}}.$$

Central contour moments, which are translation invariant, are given by the general formula

$$\mu_{pq} = \sum (x, y) \in C(x - \bar{x})^p (y - \bar{y})^q \tag{5.2}$$

Simple formulas for central moments of order up to three in terms of the centroid and the geometrical moments are

$$
\begin{aligned}
\mu_{00} &= m_{00}, \\
\mu_{10} &= 0, \\
\mu_{01} &= 0, \\
\mu_{11} &= m_{11} - \bar{x}m_{01} = m_{11} - \bar{y}m_{10}, \\
\mu_{20} &= m_{20} - \bar{x}m_{10}, \\
\mu_{02} &= m_{02} - \bar{y}m_{01}, \\
\mu_{21} &= m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01}, \\
\mu_{12} &= m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2 m_{10}, \\
\mu_{30} &= m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2 m_{10} \\
\mu_{03} &= m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2 m_{01}
\end{aligned} \tag{5.3}
$$

Scale invariant measures is obtained from the central moments using

$$
\nu_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\left(1+\frac{p+q}{2}\right)}} \tag{5.4}
$$

From these, one can compute seven invariant (to Euclidian transformations) measures using simple formulas as follows:

$$
\begin{aligned}
I_1 &= \nu_{20} + \nu_{02}, \\
I_2 &= (\nu_{20} - \nu_{02})^2 + (2\nu_{11})^2, \\
I_3 &= (\nu_{30} - 3\nu_{12})^2 + (3\nu_{21} - \nu_{03})^2, \\
I_4 &= (\nu_{30} + \nu_{12})^2 + (\nu_{21} + \nu_{03})^2, \\
I_5 &= (\nu_{30} - 3\nu_{12})(\nu_{30} + \nu_{12})[(\nu_{30} + \nu_{12})^2 - 3(\nu_{21} - \nu_{03})^2] + \\
&\quad (3\nu_{21} - \nu_{03})(\nu_{21} + \nu_{03})[3(\nu_{30} + \nu_{12})^2 - (\nu_{21} + \nu_{03})^2] \\
I_6 &= (\nu_{20} - \nu_{02})[(\nu_{30} + \nu_{12})^2 - (\nu_{21} + \nu_{03})^2] + \\
&\quad 4\nu_{11}(\nu_{30} + \nu_{12})(\nu_{21} + \nu_{03}) \\
I_7 &= (3\nu_{21} - \nu_{03})(\nu_{30} + \nu_{12})[(\nu_{30} + \nu_{12})^2 - 3(\nu_{21} + \nu_{03})^2] - \\
&\quad (\nu_{30} - 3\nu_{12})(\nu_{21} + \nu_{03})[3(\nu_{30} + \nu_{12})^2 - (\nu_{21} + \nu_{03})^2].
\end{aligned} \tag{5.5}
$$

Higher order invariant measures would involve more complicated procedures, moreover, results reported in the literature [Chen et al., 2004b] using the first seven Hu's invariants for character recognition are very promising.

5.2 Implementation and Experiments

A posture recognition module was built around Hu's invariants in the following way:

(1) Firstly, a video clip of 20 seconds (at 15 frames per second) in which the hand remains rigid in a single posture is produced for each posture the system is supposed to recognize later, say, P postures total. These videos are meant to produce training data for a posture classifier.

(2) For every video produced in the previous step, the contour tracker is applied and the contour produced for each frame in the video is used to compute a shape descriptor (a 7-dimensional vector in this case). The shape descriptors are piled in a matrix with as

many rows (shape descriptors) as the number of frames in the P video clips combined. Notice that a vector of labels is easily produced since we know that every shape vector computed for a video clip will belong to a single (known) class.

(3) The shape descriptor *training* matrix along with the label vector is then used to train a learning machine, which can be used later as a posture classifier integrated in the GPUI based on hand contours.

5.2.1 The rock-paper-scissors game

The rock-paper-scissors game is based on three very simple and easily discernible hand postures (see Figure 5.1). The posture recognition system depicted here was trained to recognize them with video clips characterized by low noise level, approximately constant illumination and lack of background clutter. The test sequence was produced in the same settings of the training videos.

The experiment used a SVM with Gaussian kernel ($\gamma = 0.15$) and penalty parameter equal to 100. The resulting machine consisted of 52 support vectors and took less than a second training. The average classification accuracy on test data (for 10 executions) was of 98.5%.

5.2.2 A 3D navigation system

The 3D navigation system mentioned earlier consisted of a posture set 66% larger than the previous experiment (five postures). The differences between postures in this set were more subtle (Figure 1.3). Furthermore, both the training and testing sequences here were taken under varying illumination therefore, producing contours with a higher level of noise which was introduced in the shape descriptors. Using a SVM to model the postures, with Gaussian kernel ($\gamma = 2^{-4}$) and penalty parameter 2^14, the resulting machine consisted of around 450 support vectors and took a few seconds (≤ 2) training. The average classification accuracy on test data (for 10 executions) was of 88.7%.

5.3 A discussion on alternatives to Hu's invariants

On the making of this thesis, other alternatives to posture recognition from contours were considered. However, the Hu's invariants approach described in here quickly singled out as the best one for the present.

A note is needed here on Fourier descriptors. Zahn and Roskies [1972] proposed Fourier descriptors for shape matching. One way of deriving Fourier descriptors is by looking at the 2D Cartesian plane where the contour lies as it was the complex plane, that is, the contour $C(s) = (x(s), y(s))$, where s is the arc length, is written as

$$C(t) = x(t) + jy(t)$$

where $j = \sqrt{-1}$ and $t = 2\pi/L$, such that, L is the total length of C.

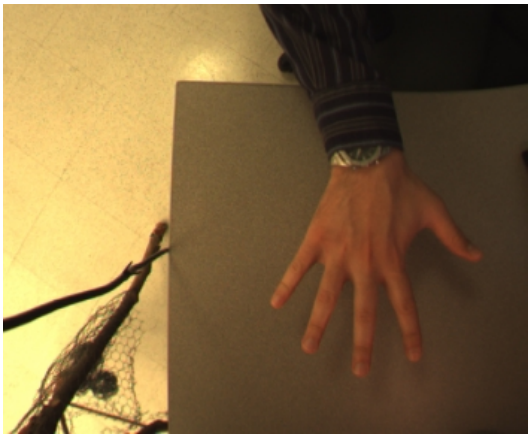
The Fourier descriptors of $C(t)$ are, therefore, the norm of the coefficients c_n of the Fourier series

$$C(t) = \sum_{-\infty}^{+\infty} c_n e^{jnt} \tag{5.6}$$

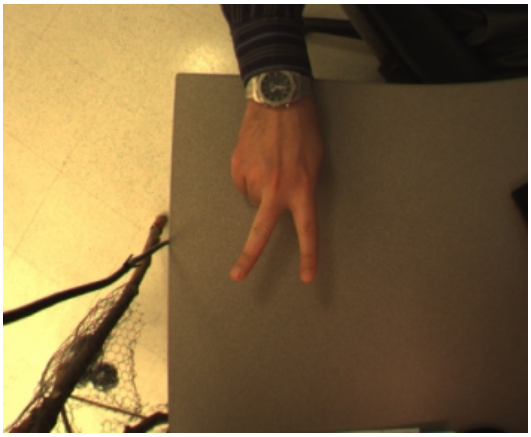
The c_n's can be computed from

$$c_n = \frac{1}{2\pi} \int_0^{2\pi} C(t) e^{-jnt} dt \tag{5.7}$$

(a) **Rock** beats scissors



(b) **Paper** beats rock -wraps it!



(c) **Scissors** beats paper. Obviously.

Figure 5.1: Rock-paper-scissors game hand-postures

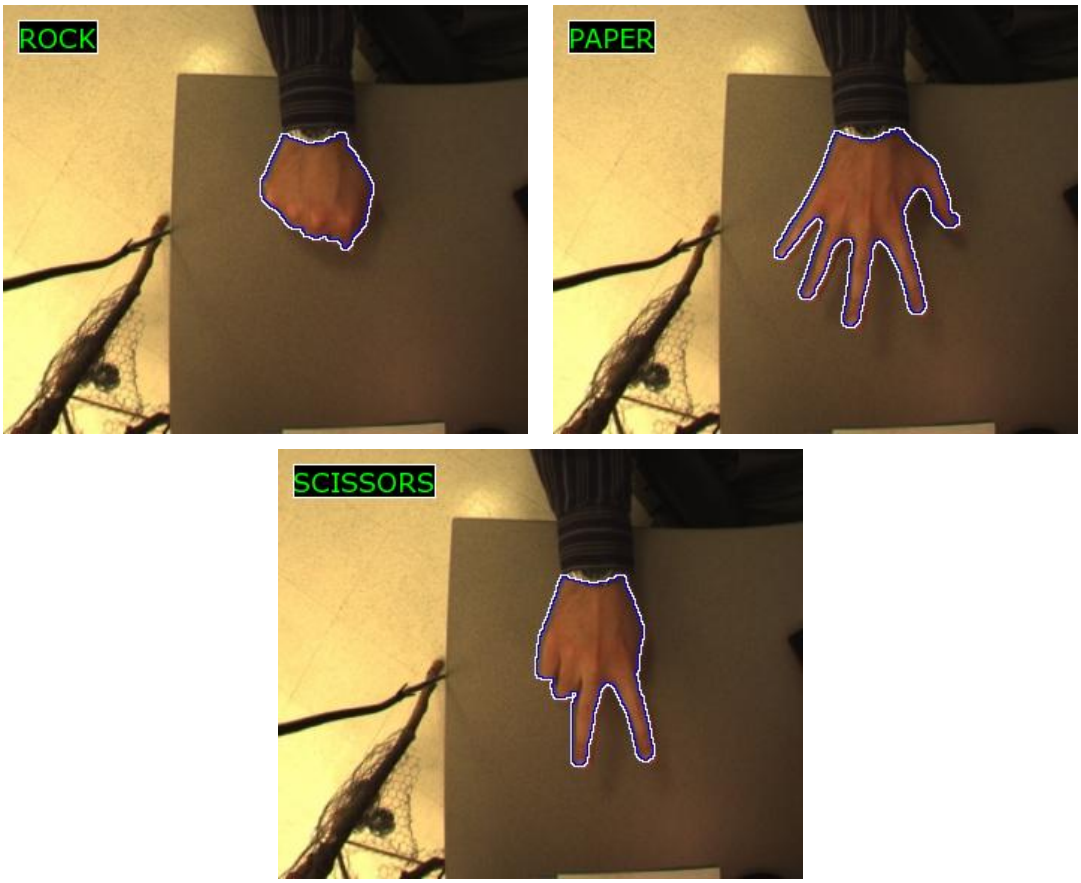


Figure 5.2: The classification rate in this sequence was around 98.5%.

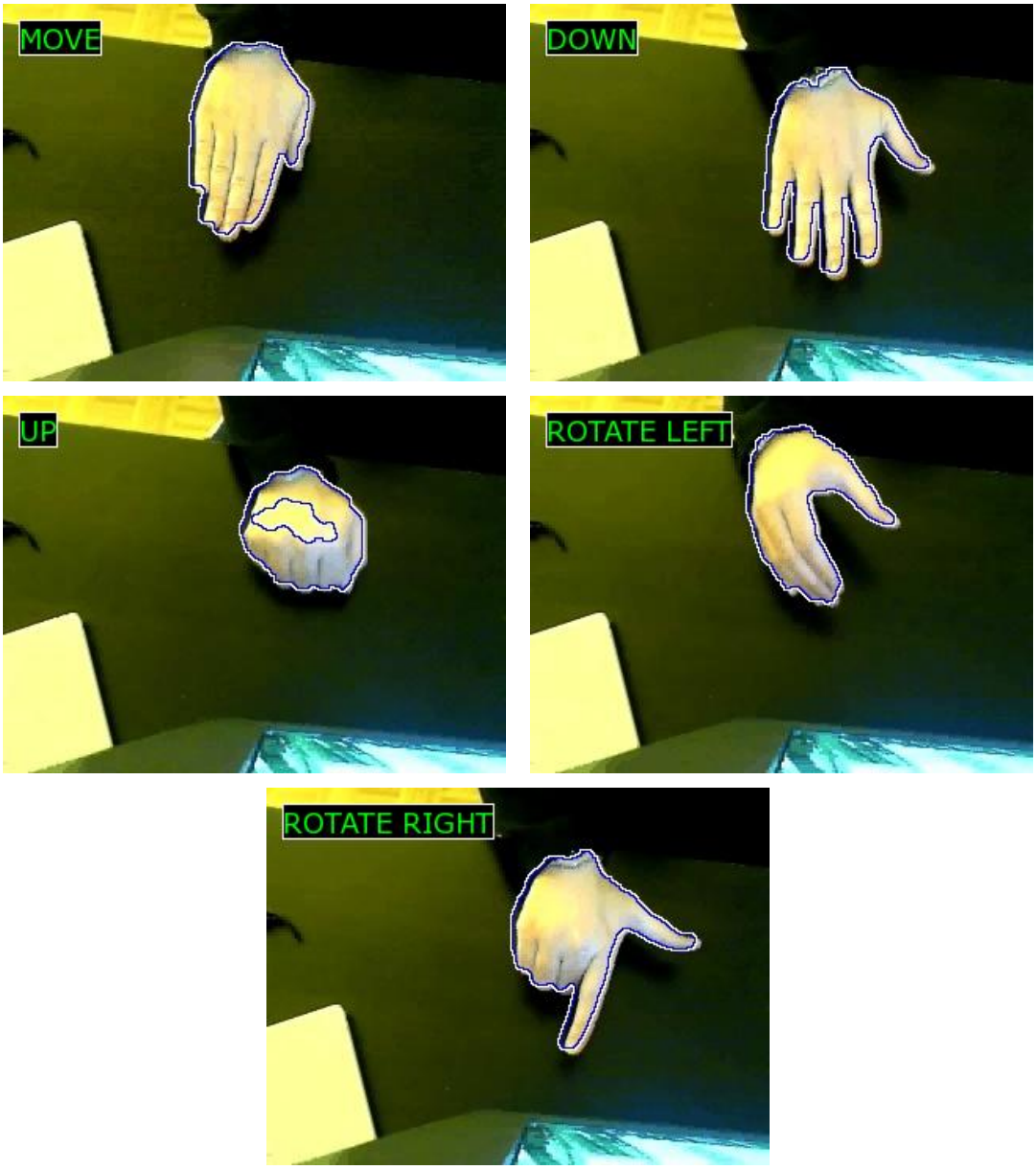


Figure 5.3: More classes and noisy contour extraction still yields a fairly good accuracy (88.7%)

The Fourier series is truncated in practice. The coefficients can be normalized in order to become invariant to translation, scaling and rotation.

Fourier descriptors were the first candidates I considered for contour description, however, this technique proved to be incompatible with the level set framework. These type of shape descriptors need the contour points to be stored in a spatially ordered list, that is, the contour has to be explicitly represented (parameterized) which is not the case for the level-set framework that uses implicit representation.

Another alternative considered was to build a feature vector for shape recognition based on *shape contexts*. Shape contexts were proposed recently [Belongie and Malik, 2000] in a framework for shape matching. These shape descriptors present the desirable property of being independent of parameterization and robust to noise. Essentially, shape contexts are 2D histograms, produced in the following way:

- Pick n points on the contour;
- For each point p_i on the shape, consider the $n-1$ vectors obtained by connecting p_i to all other points. The set of all these vectors is a rich description of the shape localized at that point but is far too detailed;
- For every point p_i, the coarse histogram of the relative *polar* coordinates of the remaining $n-1$ points,

  $$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\}$$

  is defined to be the shape context of p_i (Figure 5.4). The bins are normally taken to be uniform in log-polar space.

Shape contexts are intrinsically invariant to position and can be easily made invariant to scaling and rotation. However, it soon became clear in preliminary experiments that there was no trivial way of making a low dimensional shape context based shape descriptor. A single shape context is a 2D matrix with usually more than forty degrees of freedom in practical cases. A feature vector with such high dimensions (multiple of forty, for example) was probably responsible for extremely poor classification performance in preliminary experiments, thus, this approach was eliminated as a candidate.

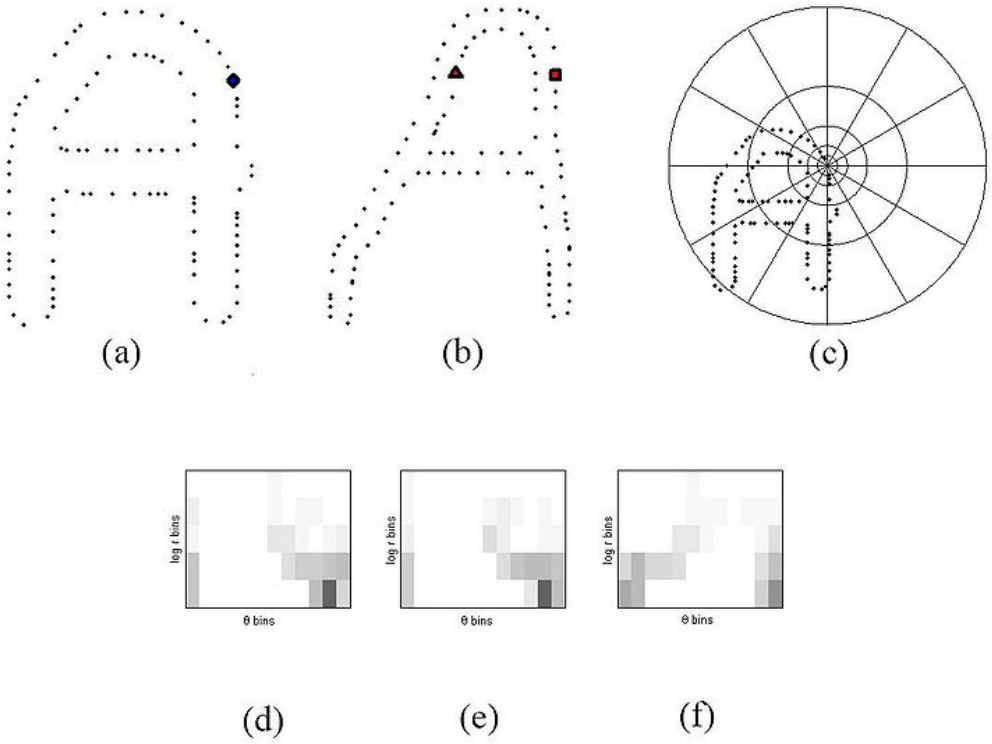


Figure 5.4: Bin distribution for shape context computation (from Belongie et al. [2002]). (a) and (b) are the sampled edge points of the two shapes. (c) is the diagram of the log-polar bins used to compute the shape context. (d) is the shape context for the circle, (e) is that for the diamond, and (f) is that for the triangle. As can be seen, since (d) and (e) are the shape contexts for two closely related points, they are quite similar, while the shape context in (f) is very different.

Chapter 6

Hand Motion Estimation

The shape feature-vectors studied in the previous chapter seek to eliminate trajectory information from the posture recognition process -making it independent of trajectory, thus manageable. Recall that we defined trajectory as the sequence of $4-$dimensional vectors that determine position, rotation and scale of an arbitrary posture or, in other words, the Euclidian transformation of a referential posture template. In this chapter, it is addressed the problem of recursive motion estimation (tracking) given a fixed posture. Note that the ideas presented here can be extended to the track of hand articulations as proposed in the works of Tosas and Bai [2007] and MacCormick and Isard [2000], for instance.

In the early 1990's, Andrew Blake and collaborators developed a very successful rigid-object contour tracking framework for objects undergoing motion restricted to certain spaces (Euclidian or affine spaces, for example). They based their work on B-splines for contour representation (see Blake et al. [1993] and references therein) which will be used throughout this chapter (introduced in the next section).

As was discussed in the introduction, we argue that natural gestures with the purpose of communicating a specific command will, in many cases, present a characteristic dynamic behavior, in a way that the dynamic signature of the gesture is inseparable from its meaning, for example, waving good-bye is defined by the characteristic motion as it is by the palm posture. We expect every gesture to be more or less associated with a certain dynamic mode (motion type) and that, in regular use, the hand is likely to be changing abruptly from one dynamic mode to the other such that the switching can be modeled by a Markov Chain. We assume that all dynamic modes can be modeled by a linear state-space system of the same dimension; therefore, the overall dynamics can be modeled with Markovian Jumps Linear Systems (MJLS). Markov jump linear systems (MJLS) are suitable to model physical systems that behave linearly but suffer abrupt changes in the dynamics from time to time (according to a Markov Chain process) with applications ranging from aircraft control systems to macroeconomics simulation.

We will present here a novel tracking algorithm based on a generalization of the Kalman filter for MJLS (the Costa filter); and a measurement module based on the SVM object appearance modeling method described in Chapter 3. Moreover, in this chapter, we provide algorithms for learning linear-second-order-dynamic models and Monte Carlo simulations of MJLS. Root Mean Square (RMS) error performance comparisons between the Costa filter, the Kalman filter and the quasi-optimal MJLS filter known as the Interacting Multiple Models filter (IMM) [Blom and Bar-Shalom, 1988] are presented in the final section.

6.1 Contour representation and the tracking problem formulation

6.1.1 A B-splines contour representation framework

B-splines curves are parametric curves with many interesting properties, for instance, there are available well tested algorithms for point set interpolation -or approximation- with B-splines,

furthermore, since B-splines curves are polynomial by parts, computer representation, manipulation and storage is, thus, greatly simplified, since all those operations reduce to manipulations with small vectors (polynomial coefficients).

In this framework, contours are described as linear combinations of elements of a set of spline functions previously defined, namely the *spline basis*. The contour curve $\mathbf{r}_k$ in time step k can be described as the matrix product:

$$\mathbf{r}_k(s) = U(s)^T \mathbf{Q}_k \ , 0 \leq s \leq L \tag{6.1}$$

where each $s \in [0, L]$ determines a point in $\mathbf{r}_k$, L is said to be the number of curve *spans*[1] , the column vector

$$\mathbf{Q}_k = \begin{pmatrix} \mathbf{Q}_k^x \\ \mathbf{Q}_k^y \end{pmatrix} \tag{6.2}$$

with $\mathbf{Q}_k^x, \mathbf{Q}_k^y \in \Re^{N_Q}$ is called the *control vector* and actually defines the curve shape at time step k, and

$$U(s) = \begin{pmatrix} \mathbf{B}^T(s) & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^T(s) \end{pmatrix}$$

with $\mathbf{0} = (0, 0, ..., 0)^T \in \Re^{N_B}$ is a $2 \times 2N_B$ matrix. $\mathbf{B}(s)$ is called the spline basis vector and

$$\mathbf{B}^T(s) = (\beta_0(s), \beta_1(s), ..., \beta_{N_B-1}(s)) \tag{6.3}$$

where $\beta_j(s)$, $j = 0, ..., N_B - 1$, are the spline basis functions.

The matrix $U(s)$ may be computed off-line for a fine discretization of the parameter space $[0, L]$, so that, retrieving points on the curve becomes the trivial task of calculating (6.1) given a control vector $\mathbf{Q}_k$. In practice N_Q is fixed (≈ 200).

We assume, henceforth, rigid-body motion and that the distance camera-target is such that perspective effects may be ignored. Furthermore, given the contour description for $k = 0$, $\mathbf{Q}_0$, every other subsequent contour disposition, $\mathbf{Q}_k$, is a result of an Euclidian transformation of $\mathbf{Q}_0$, that is

$$\mathbf{Q}_k = W_0 \mathbf{X}_k + \mathbf{Q}_0 \tag{6.4}$$

where $\mathbf{X}_k \in \Re^{N_X}$ is the configuration vector for time-step k and

$$W_0 = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{Q}_0^x & -\mathbf{Q}_0^y \\ \mathbf{0} & \mathbf{1} & \mathbf{Q}_0^y & \mathbf{Q}_0^x \end{pmatrix} \tag{6.5}$$

with $\mathbf{1} = (1, 1, ..., 1)^T \in \Re^{N_B}$, defines all acceptable transformations of $\mathbf{Q}_0$, in fact, its column vectors form a basis for the space of Euclidian transformations of $\mathbf{Q}_0$. The subset of contours $\mathbf{Q}_k$ for which (6.4) holds is said to be the *configuration* (or *shape*) *space* of $\mathbf{Q}_0$. Moreover, each element of the configuration vector has a well defined meaning:

- $\mathbf{X}^0$ is the vertical coordinate
- $\mathbf{X}^1$ is the horizontal coordinate
- $\mathbf{X}^2$ and $\mathbf{X}^3$ combined define rotation α and scaling ρ in the following way:

$$\mathbf{X}^2 = \rho(\cos(\alpha) - 1) \tag{6.6}$$

$$\mathbf{X}^3 = \rho \sin(\alpha) \tag{6.7}$$

[1] In our implementation L was set to 50.

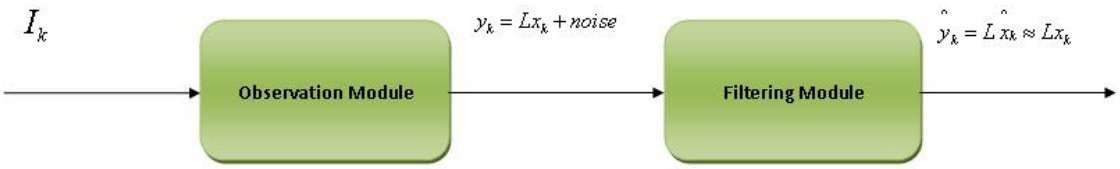


Figure 6.1: Basic elements of the tracking system. I_k is the k^{th} input video frame, $\mathbf{y}_k$ is the measured state and $\hat{\mathbf{y}}_k$ is the filtered measurement which will hopefully have a lower level of noise on average.

6.1.2 State space filtering formulation

We assume that we have some *a priori* knowledge of the dynamics in the form of a second-order autoregressive dynamic equation:

$$\mathbf{X}_k - \bar{\mathbf{X}} = \mathfrak{a}_2(\mathbf{X}_{k-2} - \bar{\mathbf{X}}) + \mathfrak{a}_1(\mathbf{X}_{k-1} - \bar{\mathbf{X}}) + \mathfrak{b}\omega_k \tag{6.8}$$

where $\mathfrak{a}_1$, $\mathfrak{a}_2$ and $\mathfrak{b}$ are $N_X \times N_X$ matrices, and ω_k is a vector of Gaussian random variables with zero mean and variance equal to 1, and $\bar{\mathbf{X}} = E[\mathbf{X}_k]$. As will be seen in detail in Section 6.4.1, all input data is preprocessed for normalization to $[-1, 1]$, therefore, $\bar{\mathbf{X}} = 0$.

Defining the state vector

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{X}_{k-1} \\ \mathbf{X}_k \end{pmatrix} \tag{6.9}$$

we are able to rewrite (6.8) as

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\omega_k \tag{6.10}$$

where

$$A = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ \mathfrak{a}_2 & \mathfrak{a}_1 \end{pmatrix} \text{ and } B = \begin{pmatrix} \mathbf{0} \\ \mathfrak{b} \end{pmatrix} \tag{6.11}$$

The state vector $\mathbf{x}_k$ is measured under noise, for which an observation model is assigned

$$\mathbf{y}_k = L\mathbf{x}_k + H\eta_k \tag{6.12}$$

where $\mathbf{y}_k$ is the observation or measurement, L the *observation coefficient* matrix, H the measurement-noise covariance matrix, and η_k is defined in the same way as ω_k.

In general, the tracking problem is the problem of *recursive estimation* of the state vector $\mathbf{x}_k$, to be carried out over a $2N_X$-dimensional *state space*, given the dynamic model (6.10), the observation model (6.12), and the measurement signal $\{\mathbf{y}_k\}$.

As will be seen in the following sections, our measurement module extracts (from the video frames) a noisy measurement of the current configuration vector, that is,

$$L = \begin{pmatrix} \mathbf{0}_{N_X \times N_X} & \mathbf{I}_{N_X} \end{pmatrix}. \tag{6.13}$$

In this context, a tracker can be seen as the composition of two basic modules: the measurement or observation module, which extracts noisy measurements of the configuration from the video; and the filtering module, which fuses motion-dynamic information, previously acquired and encoded in linear models, with measurement information, in order to produce a (tentatively) de-noised version of the measurement itself. The block diagram of Figure 6.1 illustrates the basic organization of the tracker.

The optimal solution to the filtering problem posed by equations (6.10) and (6.12) is the well known Kalman filter; with the condition that the distribution of the initial state and the

measurements are Gaussians. For non-Gaussian measurements, the Particle filter (CONDENSATION [Isard and Blake, 1998a]) has been applied with success, however, in this case real-time performance is usually difficult due to the large number of particles needed.

6.2 Modeling dynamics with Markov jump linear systems and the Costa filter

Usually a single linear dynamic model (a pair (A, B)) is used by the filter to estimate motion for the entire video, although, throughout the video, the actual target dynamic (or operation mode) often varies significantly. The ideal solution would consider this dynamic variation during motion, such that, for each time step, the exact dynamic model would be applied by the filter, however, this approach turns out to be unrealistic or unpractical.

In this work we consider the case in which multiple linear dynamic models are available, each roughly describing a certain kind of motion that might dominate motion in different parts of the video, for example, a model for rotation, another for in-plane translation, one more for scale variation, and possibly models for combinations of these types of motion. In practice, these models are *learned* from training data. We assume also that the transitions between motion modes are random and occur in such small time intervals that we may consider them as *abrupt* changes in the dynamics.

This particular case, nonetheless, covers important real-life applications, in particular gesture recognition (see Tomasi et al. [2003] and Athitsos and Sclaroff [2002]). A MJLS is characterized by a finite set of discrete-time linear systems with modal transition given by a Markov chain, that is, we consider systems of the form:

$$\mathcal{G} = \begin{cases} \mathbf{x}_{k+1} = A_{\theta_k}\mathbf{x}_k + B_{\theta_k}\omega_k \\ \mathbf{y}_k = L_{\theta_k}\mathbf{x}_k + H_{\theta_k}\eta_k \\ \mathbf{x}_0, \theta_0 \text{ given.} \end{cases} \tag{6.14}$$

where the noise processes ω_k and η_k are both vectors of Gaussian random variables with zero mean and variances equal to one, θ_k is called the jump parameter and determines the current mode, and $\mathbf{x}_0$ and θ_0 are the given initial conditions of the contour state and the jump parameter. The jump parameter is assumed to evolve stochastically as a Markov chain process with transition matrix T and finite state-space $\mathcal{S} = \{1, \cdots, N\}$.

It is well known that the optimal filter for MJLS with unknown θ_k (our case) is a non-linear, varying size, bank of Kalman filters, which is unfeasible in practice since it requires quantities of memory and CPU cycles that increase exponentially with k . Many important *nonlinear suboptimal* filters were proposed, among them the Interactive Multiple Model (IMM) algorithm in [Blom and Bar-Shalom, 1988] and the particle filter in [Doucet et al., 2000b].

In this work we advocate the Linear Minimum Mean Square Error Filter for Discrete-time Markov jump linear systems first proposed by Costa [1994] (see Fragoso et al. [2005] and Fragoso and Rocha [2006] for continuous-time versions) which will be hereafter referred to as the Costa filter (CF). The Costa filter is, in contrast with the IMM and others, a light-weight *linear* filter with sound stability properties.

The Costa filter works with the following assumptions:

(1) $B_i B_i' > 0$ ($'$ = transpose) for all $i \in \mathcal{S}$.

(2) $\{\omega_k\}$ and $\{\eta_k\}$ are null mean second-order, independent wide sense stationary sequences mutually independent with covariance matrices equal to the identity.

(3) $\mathbf{x}_0\delta_{\{\theta_0=i\}}, i \in \mathcal{S}$ are second order random vectors with the expected value $E(\mathbf{x}_0\delta_{\{\theta_0=i\}}) = \mu_i$ and $E(\mathbf{x}_0\mathbf{x}_0'\delta_{\{\theta_0=i\}}) = V_i, i \in \mathcal{S}$.

(4) $\mathbf{x}_0$ and $\{\theta_k\}$ are independent of $\{\omega_k\}$ and $\{\eta_k\}$.

Filtering happens in an augmented state space, that is, estimations are produced for an augmented state-vector $\mathbf{z}_k$ defined as follows:

$$\mathbf{z}_k^j \triangleq \mathbf{x}_k \delta_{\{\theta_k = j\}} \in \Re^n \tag{6.15}$$

$$\mathbf{z}_k \triangleq \begin{pmatrix} \mathbf{z}_k^1 \\ \vdots \\ \mathbf{z}_k^n \end{pmatrix} \in \Re^{Nn}. \tag{6.16}$$

where, in our case, $n = 2N_X$. We also define $\mathbf{q}_k = E(\mathbf{z}_k)$, the projection $\hat{\mathbf{z}}_{k|k-1}$ of $\mathbf{z}_k$ onto the linear subspace spanned by $\mathbf{y}^{k-1} = (\mathbf{y}_{k-1}^T \cdots \mathbf{y}_0^T)^T$ and

$$\tilde{\mathbf{z}}_{k|k-1} \triangleq \hat{\mathbf{z}}_k - \mathbf{z}_{k|k-1}.$$

The second-moment matrices associated to the above variables are

$$\begin{aligned} Q_k^i &\triangleq E(\mathbf{z}^{\mathbf{i}}{}_k \mathbf{z}^{\mathbf{iT}}{}_k) \in \mathbb{B}(\Re^n), i \in \mathcal{S}, \\ Z_k &\triangleq E(\mathbf{z}_k \mathbf{z}_k^T) = diag[Q_k^i] \in \mathbb{B}(\Re^{Nn}), \\ \hat{Z}_{k|l} &\triangleq E(\hat{\mathbf{z}}_{k|l} \hat{\mathbf{z}}_{k|l}^T) \in \mathbb{B}(\Re^{Nn}), 0 \leq l \leq k, \\ \tilde{Z}_{k|l} &\triangleq E(\tilde{\mathbf{z}}_{k|l} \tilde{\mathbf{z}}_{k|l}^T) \in \mathbb{B}(\Re^{Nn}), 0 \leq l \leq k. \end{aligned} \tag{6.17}$$

where $diag[Q_k^i]$ is the block diagonal matrices with diagonal elements $Q_k^i, \forall i \in \mathcal{S}$ and $\mathbb{B}$ stands for **Banach** space, therefore, $\mathbb{B}(\Re^l, \Re^p)$ is, in our case, the space of real valued matrices of dimension $p \times l$ ($\mathbb{B}(\Re^l) = \mathbb{B}(\Re^l, \Re^l)$. The Costa filter is, in fact, a linear filter for the linear system built from the following augmented matrices

$$\mathsf{A} \triangleq \begin{bmatrix} T_{11}A_1 & \cdots & T_{N1}A_N \\ \vdots & \ddots & \vdots \\ T_{1N}A_1 & \cdots & T_{NN}A_N \end{bmatrix} \in \mathbb{B}(\Re^{Nn}) \tag{6.18}$$

$$\mathsf{H} \triangleq \begin{bmatrix} H_1\pi_1^{1/2} & \cdots & H_N\pi_N^{1/2} \end{bmatrix} \in \mathbb{B}(\Re^{Nn}, \Re^n) \tag{6.19}$$

$$\mathsf{L} \triangleq \begin{bmatrix} L_1 & \cdots & L_N \end{bmatrix} \in \mathbb{B}(\Re^{Pn}, \Re^n) \tag{6.20}$$

$$\mathsf{B} \triangleq diag\left[\begin{bmatrix} T_{1j}\pi_1^{1/2}B_1 & \cdots & T_{Nj}\pi_N^{1/2}B_N \end{bmatrix}\right] \in \mathbb{B}(\Re^{N^2n}, \Re^{Nn}). \tag{6.21}$$

where $\pi_i = Prob(\theta_k = i), i \in \mathcal{S}$.

With that in hand, it can be shown that, according to Costa [1994], the linear minimal mean squared error state estimate $\hat{\mathbf{x}}_k$, is given by

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N} \hat{\mathbf{z}}_{k|k}^i \tag{6.22}$$

The filtered measurements, $\hat{\mathbf{y}}_k$, which are the final output of the tracker which, in our case, is the spline configuration vector $\mathbf{X}_k$ defined in (6.4), is given directly by the product

$$\hat{\mathbf{y}}_k = \mathsf{L}\hat{\mathbf{z}}_{k|k} \tag{6.23}$$

Where the augmented state estimates $\hat{\mathbf{z}}_{k|k}$ are produced by the recursive equation

$$\hat{\mathbf{z}}_{k|k} = \hat{\mathbf{z}}_{k|k-1} + \tilde{Z}_{k|k-1}^i \mathsf{L}^T (\mathsf{L}\tilde{Z}_{k|k-1}\mathsf{L}^T + \mathsf{H}\mathsf{H}^T)^{-1} (\mathbf{y}_k - \mathsf{L}\hat{\mathbf{z}}_{k|k-1}) \tag{6.24}$$

$$\hat{\mathbf{z}}_{k|k-1} = \mathsf{A}\hat{\mathbf{z}}_{k-1|k-1}, k \geq 1 \tag{6.25}$$

$$\hat{\mathbf{z}}_{0|-1} = \mathbf{q}(0) \tag{6.26}$$

One of the main features of this filter is that the error covariance matrices $\tilde{Z}_{k|k-1}$ needed in the equation above can be computed offline, in fact, the Costa filter **gain** term $\mathsf{C}_k = \tilde{Z}^i_{k|k-1}\mathsf{L}^T(\mathsf{L}\tilde{Z}_{k|k-1}\mathsf{L}^T + \mathsf{H}\mathsf{H}^T)^{-1}$ (an equivalent of the Kalman gain in the augmented space) in (6.24) can be computed *a priori*, for a long range of k's, and stored. Therefore, during filtering time the only computation necessary is of the form:

$$\hat{\mathbf{z}}_{k|k} = \hat{\mathbf{z}}_{k|k-1} + \mathsf{C}_k(\mathbf{y}_k - \mathsf{L}\hat{\mathbf{z}}_{k|k-1}). \tag{6.27}$$

A side-by-side comparison of our implementations of the Costa filter and the IMM have shown that the former is orders of magnitude faster than the latter which, because of its mixing or interactive step, requires the online computation of N Kalman filters (including covariances).

The sequence of covariances $\tilde{Z}_{k|k-1}$ for $k = 1, 2, 3, \cdots$ is obtained through

$$\tilde{Z}_{k|k-1} = Z_k - \hat{Z}_{k|k-1} \tag{6.28}$$

recall that Z_k is defined in terms of the set $\{Q^j_k\}_{j=1,\cdots,N}$ (6.17), which are computed recursively through

$$\begin{aligned} Q^j_{k+1} &= \sum_{i=1}^{N} T_{ij}(A_i Q^i_k A^T_i + \pi_i B_i B^T_i) \\ Q^j_0 &= \mathbb{Q}_0 \pi_j, j = 1, 2, \cdots, N \end{aligned} \tag{6.29}$$

$\mathbb{Q}_0$ being set to $\mathbf{x}_0 \cdot \mathbf{x}^T_0$ for simplicity. In general, $\mathbb{Q}_0$ should be set to the expectation $E[\mathbf{x}_0\mathbf{x}^T_0]$.

$\hat{Z}_{k|k-1}$ is computed recursively as follows

$$\hat{Z}_{k|k} = \hat{Z}_{k|k-1} + \hat{Z}_{k|k-1}\mathsf{L}^T(\mathsf{L}\tilde{Z}_{k|k-1}\mathsf{L}^T + \mathsf{H}\mathsf{H}^T)^{-1}\mathsf{L}\hat{Z}_{k|k-1}, \tag{6.30}$$

$$\hat{Z}_{k|k-1} = \mathsf{A}\hat{Z}_{k-1|k-1}\mathsf{A}^T, \tag{6.31}$$

$$\hat{Z}_{0|-1} = \mathbf{z}_0\mathbf{z}^T_0 \tag{6.32}$$

In Costa et al. [2005] it was proved that the covariance matrix can be directly obtained through from the following recursive Riccati difference equation of dimension Nn

$$\begin{aligned} \tilde{Z}_{k+1|k} &= \mathsf{A}\tilde{Z}_{k|k-1}\mathsf{A}^T + \mathfrak{B}(Q_k, k) + \mathsf{B}\mathsf{B}^T - \mathsf{A}\tilde{Z}_{k|k-1}\mathsf{L}^T(\mathsf{L}\tilde{Z}_{k|k-1}\mathsf{L}^T + \mathsf{H}\mathsf{H}^T)^{-1} \\ &\times \mathsf{L}\tilde{Z}_{k|k-1}\mathsf{A}^T. \end{aligned} \tag{6.33}$$

where the functional $\mathfrak{B}(\cdot, k) : \mathbb{H}^n \mapsto \mathbb{B}(\Re^{Nn})$ is defined for $\Upsilon = (\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_N) \in \mathbb{H}^n$ by

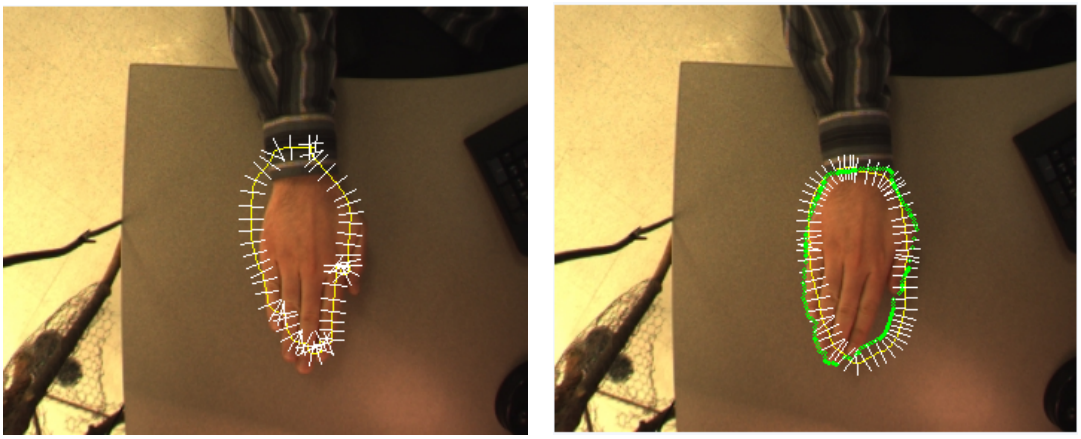
$$\mathfrak{B}(\Upsilon, k) \triangleq diag\left[\sum_{i=1}^{N} T_{ij} A_i \Upsilon_i A^T_i\right] - \mathsf{A}(diag[\Upsilon_i])\mathsf{A}^T. \tag{6.34}$$

note that the extra term $\mathfrak{B}(\cdot, k)$ would be zero for the case with no jumps ($N = 1$) reducing equation (6.33) to the standard Kalman filter Riccati equation.

6.3 The measurement module

In general, our measurement module is based on the measurement framework described in chapter six of Blake and Isard [1998]. That is, a measurement vector is obtained in two distinct steps:

(1) On the contour $(\mathbf{r}_{k-1}(s))$ associated with the previous estimative $(\mathbf{X}_{k-1})$ N_s evenly spaced points are identified. For each of these points, a search is done in a line segment perpendicular to the contour at that point to find the point in the line most likely to be an object contour point in the current time step (see illustration of the method in Figure 6.2).



(a) Previous contour estimative in yellow. Normal segments in white.

(b) Measured contour points drawn in green.

Figure 6.2: Visualization of the contour points measurement scheme in action.

(2) With the set of contour point measurements obtained with the method above, the Algorithm 3 is applied to solve the fitting problem (a minimization problem as described next). The solution of the fitting problem is a cofiguration vector whose associated contour best fits to the set of measured contour points obtained in the previous step.

6.3.1 Contour fitting

Given the previous contour estimate $\mathbf{r}_{k-1}(s)$ (or $\mathbf{X}_{k-1}$ in the configuration space) with normal lines $\mathbf{n}(s)$, and a regularization weight matrix $\bar{S} = W_0^T \mathcal{U} W_0$, where $\mathcal{U} = \frac{1}{L}\int_0^L U^T(s)U(s)ds$, the contour fitting problem is formulated as:

$$\begin{aligned} \min_{\mathbf{Y}_k} T \qquad & \text{where} \\ T \;&=\; (\mathbf{Y}_k - \mathbf{X}_{k-1})^T \bar{S} (\mathbf{Y}_k - \mathbf{X}_{k-1}) \\ &+\; \sum_{i=1}^{N_s} \frac{1}{\sigma_i^2} (\nu_i - \bar{\mathbf{n}}(s_i)^T U(s_i) W_0(s_i) [\mathbf{Y}_k - \mathbf{X}_{k-1}])^2 \end{aligned} \tag{6.35}$$

N_s is the number of normal lines traced in the measurement process, and also assumed to be the number of measured contour points available. We consider therefore the partition $s_i, i \in \mathcal{S}$, s.t. $s_1 = 0, s_{i+1} = s_i + h, s_N = L$ of the curve parameter space $[0, L]$ and let $\mathbf{r}_f(s_i)$ be the measured point obtained from the line segment passing through the curve point $\mathbf{r}_{k-1}(s_i)$ and perpendicular to the contour using the measurement method mentioned in the beginning of this section. $\bar{\mathbf{n}}$ is an outward vector perpendicular to the contour at s_i, such that, $|\bar{\mathbf{n}}(s_i)| = 1 \quad \forall i = 1, 2, 3, \ldots$.

The iterative Algorithm 3 approximates the solution for (6.35), which is equivalent to finding the contour-configuration $\mathbf{y}_k$ that fits better to the feature data $\mathbf{r}_f(s_i)$. The algorithm uses the normal projection of the displacement $\nu_i = (\mathbf{r}_f(s_i) - \bar{\mathbf{r}}(s_i))\bar{\mathbf{n}}(s_i)$ instead of the simple displacement $(\mathbf{r}_f(s_i) - \bar{\mathbf{r}}(s_i))$ in order to avoid the burden of variable *parameterization* which makes the standard norm $|\mathbf{r}_f(s) - \mathbf{r}(s)|$ useless for comparing shapes.

6.4 Important implementation issues

6.4.1 Data normalization

Data normalization has become a standard procedure in the machine learning community and all experiments involving SVMs in this thesis made use of it. The need of data normalization

Algorithm 3 CURVE FITTING

$$\mathbf{Z}_0 = \mathbf{0} \text{ and } S_0 = \mathbf{0}$$

Iterate, for $i = 1, ..., I$

$$\begin{array}{rcll} \nu_i & = & (\mathbf{r}_f(s_i) - \bar{\mathbf{r}}(s_i))\bar{\mathbf{n}}(s_i); & \\ \mathbf{h}(s_i)^T & = & \bar{\mathbf{n}}(s_i)^T U(s_i) W_0; & \\ \rho_i & = & \sqrt{\mathbf{h}(s_i)^T \mathbf{h}(s_i)}; & \\ \textbf{if} & (|\nu_i| < 2\rho) & \textbf{then} & \\ S_i & = & S_{i-1} + \dfrac{1}{\sigma_i^2}\mathbf{h}(s_i)\mathbf{h}(s_i)^T; & \\ \mathbf{Z}_i & = & \mathbf{Z}_{i-1} + \dfrac{1}{\sigma_i^2}\mathbf{h}(s_i)\nu_i; & \\ \textbf{Else} & & & \\ S_i & = & S_{i-1} \text{ , } \mathbf{Z}_i = \mathbf{Z}_{i-1}; & (6.36) \end{array}$$

The aggregated observation factor is $\mathbf{Z} = \mathbf{Z}_I$ with associated statistical information $S = S_I$. Finally, the best fitting curve is given in shape-space by:

$$\mathbf{y}_k = \mathbf{X}_{k-1} + (\bar{S} + S)^{-1}\mathbf{Z}.$$

was early noticed in machine learning, when neural networks started to become popular and to be applied to a broader spectrum of problems. It was noticed that feature vectors in many real problems contained features of different *nature* and, often, very different ranges and that, for these problems, neural networks presented a strong performance drop caused by the fact that the feature vector elements with greater values (and greater errors) were given a greater importance and elements of small values were neglected.

Through experimentation, we noticed that a similar phenomenon was taking place in our original tracker implementation. It was observed that position (true state ranging from 0 to 500 for instance) tracking was satisfactory, however, rotation and scaling (true state ranging between -3.0 and 3.0) were unacceptable. We suspected that the position errors (between 0 and 5 considering a 1% noise to signal ratio) were somehow composing with the noise for rotation and scaling (in the interval $[0, 0.03]$ considering again 1% noise to signal ratio).

In the context of this thesis, data normalization consists of, given a feature vector (measurement) $\mathbf{y}$, apply Algorithm 4, which basically centralizes the elements of $\mathbf{y}$ around 0 and then normalize all elements to a common scale ($[-1, 1]$).

Algorithm 4 Data Normalization

Initialization- Normalization uses three parameter vectors:

a) $\bar{\mathbf{y}}$ the mean of $\mathbf{y}$. This vector is approximated by averaging a given training set or obtained, by precise knowledge of the problem, by picking the half way value between the minimum and maximum physically acceptable $\mathbf{y}$.

b) κ the upper bound for $\mathbf{y}$. Every element κ^i of this vector is an upper boundary for its counter part $\mathbf{y}^i$ which is estimated by the maximum value for $\mathbf{y}^i$ in a given training set or determined by physical knowledge of the problem.

c) ι the lower bound for $\mathbf{y}$. Again, estimated from training data or determined by knowledge of the problem.

Normalization- Is done in two steps, for every element $\mathbf{y}^i$ of $\mathbf{y}$ do:

$$\mathbf{y}^i \longleftarrow \mathbf{y}^i - \bar{\mathbf{y}}^i$$

$$\mathbf{y}^i \longleftarrow 2\frac{\mathbf{y}^i - \iota^i}{\kappa^i - \iota^i} - 1$$

To be useful, the final output of the system $\hat{\mathbf{y}}$ needs to be de-normalized (Algorithm 5).

6.4.2 System identification

The learning of systems parameters from empirical data is usually called system identification in the control and filtering literature. For learning the parameters, we provide N short videos with clutter-free background, low noise, and deliberately slow motion, such that, the optimality conditions of the Kalman Filter are approximated. The video should be short and representative of a specific kind of motion (rotation, translation, etc.) that is expected to be present in the test videos. We then use the following procedure to acquire the parameters:

(1) For each short video run the single model (Kalman) filter assuming constant velocity default dynamics, that is, using:

$$A_{default} = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ \mathfrak{a}_2 & \mathfrak{a}_1 \end{pmatrix} \tag{6.38}$$

$$B_{default} = \begin{pmatrix} \mathbf{0} \\ \mathfrak{b} \end{pmatrix} \tag{6.39}$$

Algorithm 5 DATA DE-NORMALIZATION

Using the three parameter vectors acquired in the initialization of the previous algorithm $(\bar{\mathbf{y}}, \kappa, \iota)$:

$$\begin{aligned} \mathbf{y}^i &\longleftarrow \mathbf{y}^i + 1 \\ \mathbf{y}^i &\longleftarrow \mathbf{y}^i \frac{\kappa^i - \iota^i}{2} \\ \mathbf{y}^i &\longleftarrow \mathbf{y}^i + \iota^i \\ \mathbf{y}^i &\longleftarrow \mathbf{y}^i + \bar{\mathbf{y}}^i \end{aligned} \tag{6.37}$$

with $\mathfrak{a}_1 = -I_{N_X}$, $\mathfrak{a}_2 = 2I_{N_X}$ and

$$\mathfrak{b} = \gamma_0 t^{3/2} \left(W_0^T \mathcal{U} W_0 \right)^{-\frac{1}{2}}$$

where I_{N_X} is the $N_X \times N_X$ identity matrix, t is the time in seconds and γ_0 is the rate of growth of the radii of spherical search area ($\approx 35 pixels/s$). The estimated *configuration* sequence $\{\mathbf{X}_k\}$ for the interval in which the tracker estimates correctly is recorded, let's say the first J video frames.

(2) The training data obtained in the above step is then employed in the following equations (from Blake and Isard [1998]) to identify the dynamic parameters A_i and B_i that would *emulate* this training set.

First calculate the auxiliary vectors R_i and matrices R_{ij} and R'_{ij} for $i, j = 0, 1, 2$:

$$\begin{aligned} R_i = \sum_{k=3}^{J} \mathbf{X}_{k-i}, \qquad & R_{ij} = \sum_{k=3}^{J} \mathbf{X}_{k-i} \mathbf{X}_{k-j}^T, \\ R'_{ij} \quad = \quad & R_{ij} - \frac{1}{J-2} R_i R_j^T. \end{aligned} \tag{6.40}$$

Then estimate the subparts of A_i and B_i as in (6.11):

$$\begin{aligned} \mathfrak{a}_2 &= \left(R'_{02} - R'_{01} R'^{-1}_{11} R'_{12} \right) \left(R'_{22} - R'_{21} R'^{-1}_{11} R'_{12} \right)^{-1} \\ \mathfrak{a}_1 &= \left(R'_{01} - \mathfrak{a}_2 R'_{21} \right) R'^{-1}_{11} \\ \mathfrak{b} &= \left(\frac{1}{J-2} \left(R_{00} - A_2 R_{20} - A_1 R_{10} - \mathbf{D} R_0^T \right) \right)^{\frac{1}{2}} \end{aligned}$$

where

$$\mathbf{D} = \frac{1}{J-2} \left(R_0 - \mathfrak{a}_2 R_2 - \mathfrak{a}_1 R_1 \right).$$

(3) Repeat the process above for the N available exemplar videos.

6.4.3 Mode transition probability modeling and Markov chain generation

The mode transition probability matrix (MTPM), T, determines the frequency of jumps between modes and has to more or less model the behavior of the underlying physical problem. In our implementation, T is defined as

$$T \triangleq \left(\varrho I_{N \times N} + \mathbf{1}_{N \times N} \right) \frac{1}{N + \varrho} \tag{6.41}$$

where ϱ is an integer which determines on average how many time steps the systems stays on the current mode before jumping with equal probability to any of the other possible modes. As it was defined, T depends on a single parameter ϱ which controls an important property of the Markov

chain, namely, the average jump frequency. By varying ϱ different filters can be compared in different scenarios of jump frequency. This choice of a symmetric MTPM is very generic and we predict that overall performance could improve substantially if MTPMs tuned for the specific task at hand are used, for example, by learning the MTPM from empirical data.

A simple method to generate Markov chain paths, given the MTPM and the length of the chain l, was implemented (Algorithm 6) in order to carry out the Monte Carlo simulations described in the simulations section of this chapter as well as to help evaluate choices of ϱ.

Algorithm 6 MARKOV CHAIN GENERATOR

Input: T, l, the mode transition probability matrix and the length of the chain

Output: θ , the Markov chain which is a l-dimensional vector of elements from the set $\{1, \ldots, N\}$

Initialization: Compute the matrix F, such that,

$$F_{ij} = \sum_{k=1,\ldots,j} T_{ik} \tag{6.42}$$

Generation loop:

$\theta_1 \longleftarrow 1$
for $k = 1$ to $l - 1$ **do**
  Using the programing language built-in uniform random number generator, draw r from $U[0, 1]$
  $j \longleftarrow 1$
  while $F_{\theta_k j} < r$ **do**
    $j \longleftarrow j + 1$
  end while
  $\theta_{k+1} \longleftarrow j$
end for

6.4.4 Observation noise modeling

The observation noise covariance matrices H_i, $i = 1, \ldots N$ are all defined in the same way:

$$H_i = \sigma_0 I_{N_X \times N_X} \tag{6.43}$$

where σ_0 was determined by trial and error through visual inspection. We concluded that with $\sigma_0 \in [0.02, 0.06]$ a sequence of simulated observations can be generated which resembles actual measurements made on video.

6.5 Monte Carlo simulations

In order to assess the properties of the filter, we implemented a platform for Monte Carlo simulations where the true states are known (generated) and, therefore, exact errors can be computed and the effect of process and observation noise can be controled.

Four artificial modes (dynamic behaviors) were designed to produce different dynamic relationships among the four configuration variables (see description in Table 6.1).

Furthermore, to the deterministic motion presented in Table 6.1 a random value drawn from the normal distribution $N(0; \sigma_p)$ was added to play the part of the process noise. The algorithm for system identification provided in section 6.4.2 was then used to produce a set of four pairs of linear dynamic models $\{(A_i, B_i)\}_{i=1,2,3,4}$.

With the help of the Markov chain generator algorithm provided earlier (Algorithm 6) and the state update equations (6.14), we can easily generate MJLS random state sequences accompanied by noisy measurements (see illustration of Figure 6.3).

Mode	$\mathbf{X}^0$	$\mathbf{X}^1$	$\mathbf{X}^2$	$\mathbf{X}^3$
1	$c + vt - at^2/2$	$c + vt - at^2$	$\rho t/\tau$	0
2	$c + vt - at^2/2$	$c + vt - at^2$	$\rho(\tau - t)/\tau$	0
3	$c + vt - at^2$	$c + vt - at^2/2$	$\cos(\pi t/\tau) - 1$	$\sin(\pi t/\tau)$
4	$c + vt - at^2$	$c + vt - at^2/2$	$\cos(\pi(\tau - t)/\tau) - 1$	$\sin(\pi(\tau - t)/\tau)$

Table 6.1: Hand crafted dynamic modes. We used $c = 100$, $v = 10$, $a = 0.1$ and $t \in [0, 2\tau]$ (with $\tau = 100$) which assures that the motion begins and finish at the same point and the visual representation remains inside a 500×500 window. ρ was fixed equal to 2, which gives a maximum of twice the original scale.

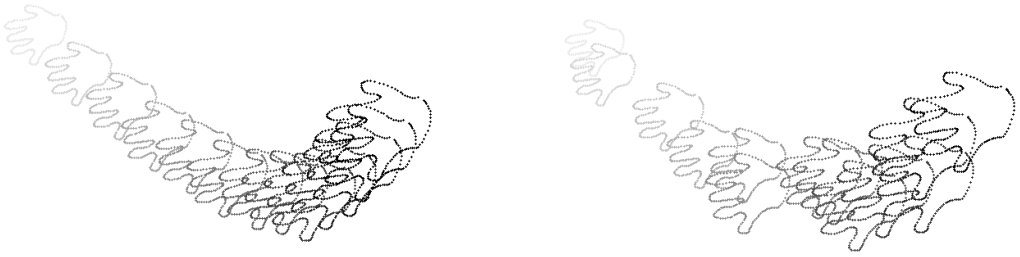
Based on this simulation platform we can perform RMS error comparisons between filters, for example, the plot in Figure 6.4 shows the RMS error of $|\hat{\mathbf{y}}_k - \mathbf{X}_k|_2$ for the filters KF, CF and IMM, which averaged the error over 2^{10} runs. In this particular case, the mode sequence was set by hand and fixed for all runs, as follows:

$$k = 1 \xrightarrow{\theta_k=1} k = 100 \xrightarrow{\theta_k=2} k = 200 \xrightarrow{\theta_k=3} k = 300 \xrightarrow{\theta_k=4} k = 400 \tag{6.44}$$

For all experiments REF is the plot of the RMS error for the un-filtered measurement $\mathbf{y}_k$, which works as a reference. Figure 6.5 shows again the RMS of the Euclidian-norm error, however, this time, the mode sequence θ was generated using Algorithm 6.

We carried out an experiment to investigate the influence of the rate of mode switching ($\propto \frac{1}{\varrho}$) on the filter performance. For that purpose, the parameters relative to process and observation noise were fixed, and the RMS error was computed for $\varrho \in [1, 300]$, that is, for jump probability values in $[0.003, 0.20]$. Results for this experiments are illustrated in Figure 6.6.

Finally, we kept the observation noise and ϱ fixed and varied σ_p (the standard deviation of the process noise) obtaining the results in Figure 6.7. Notice that, the IMM consistently achieved better estimatives for all experiments, on the other hand, the Costa filter presented the concrete advantage of being approximately $20\times$ faster on average than the IMM.



(a) Low process noise level.

(b) High process-noise random path.

Figure 6.3: Our simulation platform can generate an arbitrary number of random paths for the hand template with stochastic properties conveniently defined by the user.

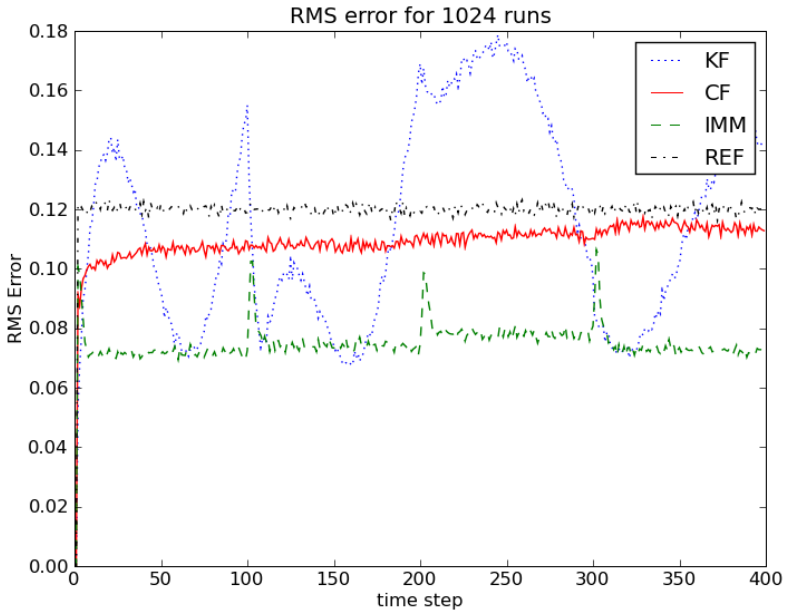


Figure 6.4: Root mean square error comparison between filters with fixed path. REF, KJ, CF, IMM are respectively the unfiltered measurement, the Kalman filter, Costa filter and IMM average errors.

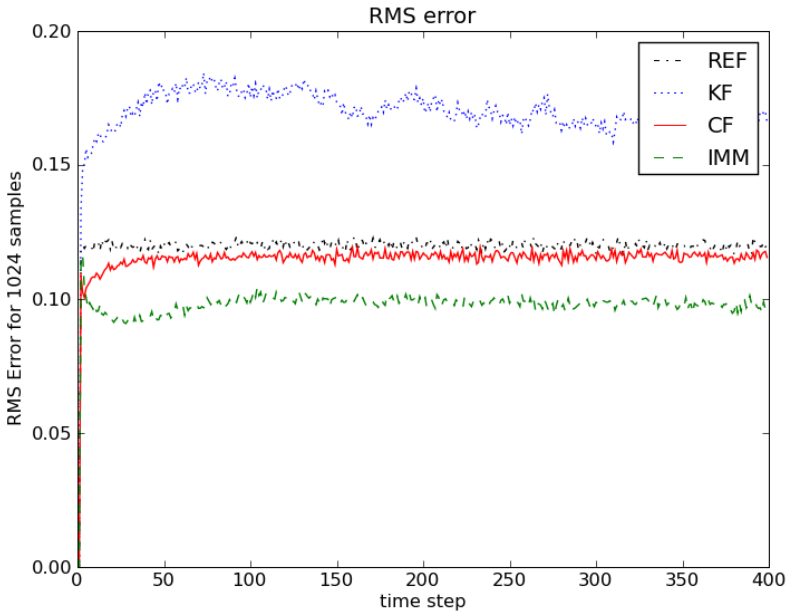


Figure 6.5: Root mean square error comparison with random paths (θ generated with Algorithm 6).

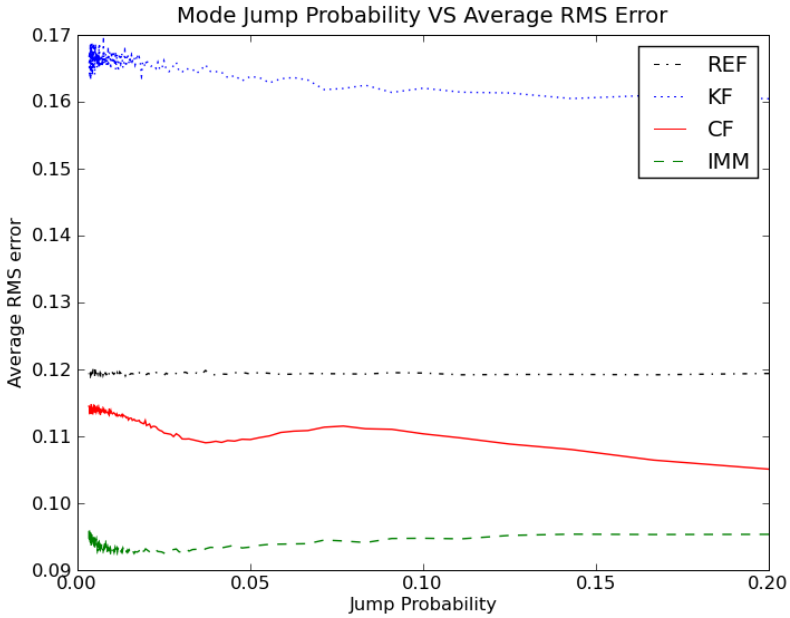


Figure 6.6: The average RMS errors for 400 time steps long MJLS random walks for different values of the mode jump probability.

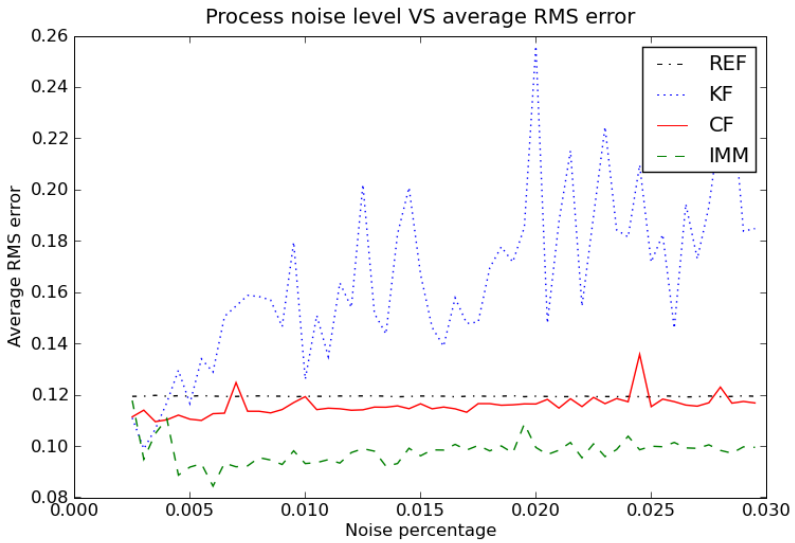


Figure 6.7: Shows the influence of process noise on the filters performances.

Chapter 7

Conclusion

7.1 Summing up

The thesis presented a toolset for the design of gesture recognition systems based on the extraction and recognition of hand contours. The main motivation was the application in intelligent human-computer interfaces. However, contour extraction in video is far from being a trivial task presenting challenging obstacles, such as: strong deterioration in the presence of background clutter and illumination fluctuation. It was argued that object contour extraction involves, even if implicitly, the classification of pixels in the image as belonging to the object or the background. It turns out that some prior information (knowledge) of the problem has to be used to build a classification rule for the pixels. The common approach to this problem involves modeling pixel values with standard parametric probability distributions, or the determination of thresholds values. A novel approach to this problem was proposed to tackle the problem of building robust pixel classification rules in which general (non-parametric) statistical learning theory –SVM, in particular– is used to acquire the model from a set of examples, that is, samples for which the correct classification is known. This approach will be able to deal efficiently with background clutter and illumination fluctuation as long as these distortions are properly represented in the set of training examples.

To improve robustness against noise in pixel classification, the pixel value is not used by itself, but a feature vector is built using information about a small vicinity around the pixel. This way, small image blocks of the image matrix are actually analyzed at a time, which is in fact a strategy closer to how the human vision-system works.

A series of experiments in the above framework were carried out to understand the effect of certain problem parameters (type of features used, size of the pixel vicinity), as well as SVM training parameters (error penalty, and RBF kernel parameter γ, the relation positives/negatives in the training samples) in the final pixel classification performance. A good amount of quantitative results could be derived from these experiments and analyzed, however, straightforward visual inspection can be easily done since the results could be put in the form of segmented hand-images.

Classification of every pixel in the video frame (segmentation) using these techniques proves to be too slow for any reasonable application. A very fast level-set approximation method is proposed to minimize the number of pixel classification operations per video frame to a small fraction of the total number of pixels in the image while retaining the major desirable qualities of the level-set contour tracking technology.

The output of the contour tracking algorithm (the hand contour for every frame) is represented in the computer as a list of 2D points. In order to make the problem of posture recognition manageable, it is necessary to come up with a fixed-dimension feature vector that uniquely describes the posture independently of the pose (orientation, position, scale), in other words, a shape descriptor invariant to Euclidian transformations is needed. In this thesis, very

low posture classification error rates are achieved in posture recognition with the use of seven Hu's moments invariants as shape descriptors.

Rigid-object contour tracking technology based on a B-splines contour representation framework was proposed for pose tracking. A Markov Jump Linear System dynamic modeling of the hand motion was proposed and a Costa filter was applied resulting in a stable and accurate pose tracking system. The measurement module of the implementation of this filter is based on a relatively small number of line searches for points in the hand contour, therefore, once again it was possible to use the SVM pixel modeling mentioned earlier, however, in an indirect way. A Monte Carlo simulation platform was used for RMS comparison of the Costa filter with the Interacting Multiple Models filter in terms of speed and estimation quality.

7.2 Future work

Different pixel features were not tested, for example, Fourier coefficients could turn out to be a more principled way of storing information about a pixel neighborhood. Experiments are needed to find out how many Fourier coefficients are sufficient for achieving low classification error rates.

Python, the programming language used for the implementation of the experiments, is an excellent programming language for quick prototyping of ideas and experimenting, however, loops in Python can be extremely slow. A more reliable investigation of speed performance of the algorithms presented in the thesis would need a complete reimplementation of everything in C/C++.

There is nothing preventing one from using neighborhood features calculated from multiple windows of different scale and positions (see Malric et al. [2008]). This is a natural way of augmenting the expressiveness of the pixel-level feature vector which was left as future work.

The hand annotation and initialization method is still very primitive and wearisome. There are various practical ways of improving it which can be tried, for instance, using hand templates.

Skin regions associated with the forearm and face and other exposed skin body parts of the user may constitute a severe distraction for the object feature extraction algorithm and consequently for the tracking procedure.

Moments based on Zernike's polynomials have interesting invariance properties and may be more expressive than the Hu's moments invariants, however, their implementation needs care for correct scaling and other details and are left as future work.

Bibliography

G. D. Abowd and E. D. Mynatt. Charting past, present, and future research in ubiquitous computing. **ACM Trans. Comput.-Hum. Interact.**, 7(1):29–58, 2000. ISSN 1073-0516. doi: http://doi.acm.org/10.1145/344949.344988.

M. Al-Amin Bhuiyan. On gesture recognition for human-robot symbiosis. In **Proc. 15th IEEE International Symposium on Robot and Human Interactive Communication RO-MAN 2006**, pages 541–545, 2006. doi: 10.1109/ROMAN.2006.314445.

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. **Neural Computation**, 9:1545–1588, 1997.

J. A. Anderson. **An Introduction to Neural Networks**. The MIT Press, 1995.

C. Andrieu, M. Davy, and A. Doucet. A particle filtering technique for jump markov systems. In **Proceedings of the XI European Signal Processing Conference**, 2002.

C. Andrieu, M. Davy, and A. Doucet. Efficient particle filtering for jump markov systems. application to time-varying autoregressions. **Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]**, 51(7): 1762–1770, 2003.

M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, D. Sci, T. Organ, and S. A. Adelaide. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. **Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]**, 50(2):174–188, 2002.

E. M. Arvacheh and H. R. Tizhoosh. Pattern analysis using zernike moments. In **Proc. IEEE Instrumentation and Measurement Technology Conference IMTC 2005**, volume 2, pages 1574–1578, 2005.

V. Athitsos. **Learning Embeddings for Indexing, Retrieval, and Classification,with Applications to Object and Shape Recognition in Image Databases**. PhD thesis, Boston University, 2006.

V. Athitsos and S. Sclaroff. An appearance-based framework for 3d hand shape classification and camera viewpoint estimation. **Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on**, pages 40–45, 2002.

V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. **Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on**, 2, 2003.

V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Learning euclidean embeddings for indexing and classification. Technical report, Computer Science Department, Boston University, 2004.

S. Avidan. Support vector tracking. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 26(8):1064–1072, 2004.

H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In **Proceedings of the ninth European Conference on Computer Vision**, 2006.

S. S. Beauchemin and J. L. Barrón. The computation of optical flow. **ACM Computing Surveys**, 27:433–466, 1995.

R. Bellman. **Adaptive Control Processes: A Guided Tour**. Princeton University Press., 1961.

S. Belongie and J. Malik. Matching with shape context. In **IEEE Workshop on Content-based Access of Image and Video Libraries**, 2000.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 24: 509–521, 2002.

D. P. Bertsekas. **Nonlinear Programming**. Athena Scientific, 1995.

M. A. A. Bhuiyan, M. E. Islam, N. Begum, M. Hasanuzzaman, Chang Hong Liu, and H. Ueno. Vision based gesture recognition for human-robot symbiosis. In **Proc. 10th international conference on Computer and information technology ICCIT 2007**, pages 1–6, 2007. doi: 10.1109/ICCITECHN.2007.4579430.

A. Blake and M. Isard. **Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion**. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1998.

A. Blake, R. Curwen, and A. Zisserman. A framework for spatio-temporal control in the tracking of visual contours. **International Journal of Computer Vision**, 11(2):127–145, 1993.

A. Blake, M. Isard, and D. Reynard. Learning to track curves in motion. **Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on**, 4, 1994.

A. Blake, M. Isard, and D. Reynard. Learning to track the visual motion of contours. **Artificial Intelligence**, 78(1-2):179–212, 1995.

H. A. P. Blom and E. A. Bloem. Particle filtering for stochastic hybrid systems. **Decision and Control, 2004. CDC. 43rd IEEE Conference on**, 3, 2004.

H.A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. **Automatic Control, IEEE Transactions on**, 33(8):780–783, 1988.

A. F. Bobick and A. D. Wilson. A state-based approach to the representation and recognition of gesture. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 19(12): 1325–1337, 1997.

L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In **Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on**, pages 405–410, 2002.

A. G. Brooks and C. Breazeal. Working with robots and objects: revisiting deictic reference for achieving spatial common ground. In **HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction**, pages 297–304, New York, NY, USA, 2006. ACM. ISBN 1-59593-294-1. doi: http://doi.acm.org/10.1145/1121241.1121292.

V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn. Fingartips: gesture based direct manipulation in augmented reality. In **GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia**, pages 212–221, New York, NY, USA, 2004. ACM. ISBN 1-58113-883-0. doi: http://doi.acm.org/10.1145/988834.988871.

A. Bulzacki, Lian Zhao, Ling Guan, and K. Raahemifar. Computerized recognition of human gestures. In **Proc. Canadian Conference on Electrical and Computer Engineering CCECE 2008**, pages 001003–001006, 2008. doi: 10.1109/CCECE.2008.4564687.

C .J. C. Burges. A tutorial on support vector machines for pattern recognition. **Data Mining and Knowledge Discovery**, 2(2):121–167, 1998.

C. J. C. Burges and B. Schölkopf. **Advances in Neural Information Processing Systems**, chapter Improving accuracy and speed of support vector learning machines, pages 375–381. MIT Press, 1997.

M. C. Cabral, C. H. Morimoto, and M. K. Zuffo. On the usability of gesture interfaces in virtual reality environments. In **CLIHC '05: Proceedings of the 2005 Latin American conference on Human-computer interaction**, pages 100–108, New York, NY, USA, 2005. ACM. ISBN 1-59593-224-0. doi: http://doi.acm.org/10.1145/1111360.1111370.

J. F. Canny. A computational approach to edge detection. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 8:679–714, 1986.

G. Caridakis, O. Diamanti, K. Karpouzis, and P. Maragos. Automatic sign language recognition: vision based feature extraction and probabilistic recognition scheme from multiple cues. In **PETRA '08: Proceedings of the 1st ACM international conference on PErvasive Technologies Related to Assistive Environments**, 2008.

C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: image segmentation using expectation-maximization and its application to image querying. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 24(8):1026–1038, 2002.

Li-Wei Chan, Yi-Wei Chia, Yi-Fan Chuang, and Yi-Ping Hung. i-m-top: An interactive multi-resolution tabletop display system. In **Proc. IEEE International Conference on Multimedia and Expo**, pages 11–11, 2007. doi: 10.1109/ICME.2007.4284573.

D. Chen, R. Malkin, and J. Yang. Multimodal detection of human interaction events in a nursing home environment. In **ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces**, pages 82–89, New York, NY, USA, 2004a. ACM. ISBN 1-58113-995-0. doi: http://doi.acm.org/10.1145/1027933.1027949.

D. Chen, J. Yang, R. Malkin, and H. D. Wactlar. Detecting social interactions of the elderly in a nursing home environment. **ACM Trans. Multimedia Comput. Commun. Appl.**, 3(1): 6, 2007a. ISSN 1551-6857. doi: http://doi.acm.org/10.1145/1198302.1198308.

F. Chen, E. Choi, J. Epps, S. Lichman, N. Ruiz, Y. Shi, R. Taib, and M. Wu. A study of manual gesture-based selection for the pemmi multimodal transport management interface. In **ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces**, pages 274–281, New York, NY, USA, 2005. ACM. ISBN 1-59593-028-0. doi: http://doi.acm.org/10.1145/1088463.1088510.

F.S. Chen, C.M. Fu, and C.L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. **Image and Vision Computing**, 21(8):745–758, 2003.

Q. Chen. **Real-Time Vision-Based Hand Tracking and Gesture Recognition**. PhD thesis, University of Ottawa, 2008.

Q. Chen, E. Petriu, and X. Yang. A comparative study of fourier descriptors and hu's seven moment invariants for image recognition. In **Canadian Conference on Electrical and Computer Engineering**, 2004b.

Q. Chen, M. Rahman, X. Shen, A. El Saddik, and N. D. Georganas. Navigating 3d virtual environments of learning object by hand gestures. **International Journal of Advanced Media and Communication**, 1:351–368, 2007b.

C. Choi, J. H. Ahn, and H. Byun. Visual recognition of aircraft marshalling signals using gesture phase analysis. In **Proc. IEEE Intelligent Vehicles Symposium**, pages 853–858, 2008. doi: 10.1109/IVS.2008.4621186.

C. K. Chui. **An Introduction to Wavelets**. Academic Press, San Diego, 1992.

A. A. A. Cordeiro Junior, M. D. Fragoso, N. D. Georganas, and J. C. de Oliveira. The markovian jump contour tracker. In **Proceedings of the 17th International Federation of Automatic Control World Congress**, 2008.

O.L.V. Costa. Linear minimum mean square error estimation for discrete-time markovian jump linear systems. **Automatic Control, IEEE Transactions on**, 39(8):1685–1689, 1994.

O.L.V. Costa, M.D. Fragoso, and R.P. Marques. **Discrete-time Markov Jump Linear Systems**. Springer, 2005.

C. Costanzo, G. Iannizzotto, and F. La Rosa. Virtualboard: real-time visual gesture recognition for natural human-computer interaction. In **Proc. International Parallel and Distributed Processing Symposium**, pages 8 pp.–, 2003. doi: 10.1109/IPDPS.2003.1213222.

D. Cremers. Dynamical statistical shape priors for level set based tracking. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 28(8):1262–1273, August 2006.

D. Cremers, N. Sochen, and C. Schnorr. Towards recognition-based variational segmentation using shape priors and dynamic labeling. **Int. Conf. on Scale Space Theories in Computer Vision**, 2695:388–400, 2003.

D. Cremers, M. Rousson, and R. Deriche. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. **International Journal of Computer Vision**, 72(2):195–215, April 2007.

R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. In **Third IEEE International Conference on Automatic Face and Gesture Recognition (FG'98)**, 1998.

J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. **Proceedings of hte IEEE Computer Societey Conference on Computer Vision and Pattern Recognition**, 2:126–133, 2000.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. **Statistics and Computing**, 10:0960–3174, 2000a.

A. Doucet, A. Logothetis, and V. Krishnamurthy. Stochastic sampling algorithms for state estimation of jump markov linear systems. **Automatic Control, IEEE Transactions on**, 45(2):188–202, 2000b.

A. Doucet, N. de Freitas, and N. Gordon, editors. **Sequential Monte Carlo Methods in Practice**. Springer, 2001a.

A. Doucet, N. J. Gordon, and V. Kroshnamurthy. Particle filters for state estimation of jump markov linear systems. **Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]**, 49(3):613–624, 2001b.

H. Du and E. Charbon. A virtual keyboard system based on multi-level feature matching. In **Proc. Conference on Human System Interactions**, pages 176–181, 2008. doi: 10.1109/HSI.2008.4581429.

R. O. Duda, P. E. Hart, et al. **Pattern Classification and Scene Analysis**. Wiley New York, 1973.

M. Ehreumann, T. Lutticke, and R. Dillmann. Dynamic gestures as an input device for directing a mobile platform. In **Proc. ICRA Robotics and Automation IEEE International Conference on**, volume 3, pages 2596–2601, 2001.

S. Eickeler, A. Kosmala, and G. Rigoll. Hidden markov model based continuous online gesture recognition. In **In Int. Conference on Pattern Recognition (ICPR)**, 1998.

A. El-Sawah, C. Joslin, N. D. Georganas, and E. M. Petriu. A framework for 3d hand tracking and gesture recognition using elements of genetic programming. In **Proc. Fourth Canadian Conference on Computer and Robot Vision CRV '07**, pages 495–502, 2007. doi: 10.1109/CRV.2007.3.

Ayman El-Sawah. **Towards Context-Aware Gesture Enabled User Interfaces**. PhD thesis, Ottawa-Carleton Institute of Computer Science School of Information Technology and Engineering University of Ottawa, 2008.

M. Elbouz, A. Alfalou, and H. Hamam. Fast transmission by mobile phone of recognition sign language using a segmented multidecision filter adapted to the parallel virtual machine (pvm). In **Proc. Future generation communication and networking (fgcn 2007)**, volume 2, pages 586–589, 2007. doi: 10.1109/FGCN.2007.125.

A. Erol, G. Bebis, M. Nicolescu, R.D. Boyle, and X. Twombly. A review on vision-based full dof hand motion estimation. **Workshop on Vision for HCI, in conjunction with CVPR**, 2005.

H. Fei. A hybrid hmm/particle filter framework for non-rigid hand motion recognition. **Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on**, 5, 2004.

R. A. Fisher. **Contributions to Mathematical Statistics**. Wiley New York, 1952.

M. D. Fragoso, O. L. V. Costa, J. Baczynski, and N. C. S. Rocha. Optimal linear mean square filter for continuous-time jump linear systems. **IEEE Transactions on Automatic Control**, 50(9):1364–1369, September 2005.

M.D. Fragoso and N.C.S. Rocha. Stationary filter for continuous-time markovian jump linear systems. **SIAM Journal on Control and Optimization**, 44(6):801–815, 2006.

T. Fukuda, Y. Nakauchi, K. Noguchi, and T. Matsubara. Human behavior recognition for cooking support robot. In **Proc. 13th IEEE International Workshop on Robot and Human Interactive Communication ROMAN 2004**, pages 359–364, 2004. doi: 10.1109/ROMAN.2004.1374787.

G. A. Giraldi, P. S. S. Rodrigues, R. L. S. Silva, A. L. Apolinário Jr., and J. S. Suri. **Deformable Models**, chapter Level Set Formulation For Dual Snake Models, pages 195–233. Springer New York, 2007.

G. A. Giraldi, P. S. Rodrigues, E. C. Kitani, and C. E. Thomaz. Dimensionality reduction, classification and reconstruction problems in statistical learning approaches. **Revista de Informática Teórica e Aplicada**, 15, 2008.

H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In **Proceedings of the British Machine Vision Conference (BMVC'06)**, 2006.

S. Grange, T. Fong, and C. Baur. Moris: a medical/operating room interaction system. In **ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces**, pages 159–166, New York, NY, USA, 2004. ACM. ISBN 1-58113-995-0.

S. Gutta, H. Huang, F. Imam, and H. Wechsler. Face and hand gesture recognition using hybrid classifiers. In **Proc. Second International Conference on Automatic Face and Gesture Recognition**, pages 164–169, 1996. doi: 10.1109/AFGR.1996.557259.

H. A. Habib and M. Mufti. Real time mono vision gesture based virtual keyboard system. **Consumer Electronics, IEEE Transactions on**, 52(4):1261–1266, 2006. ISSN 0098-3063. doi: 10.1109/TCE.2006.273143.

Z. Hang and R. Qiuqi. Visual gesture recognition with color segmentation and support vector machines. In **Proc. 7th International Conference on Signal Processing ICSP '04**, volume 2, pages 1443–1446 vol.2, 2004. doi: 10.1109/ICOSP.2004.1441598.

V. Henderson, S. Lee, H. Brashear, H. Hamilton, T. Starner, and S. Hamilton. Development of an american sign language game for deaf children. In **IDC '05: Proceedings of the 2005 conference on Interaction design and children**, pages 70–79, New York, NY, USA, 2005. ACM. ISBN 1-59593-096-5. doi: http://doi.acm.org/10.1145/1109540.1109550.

W. Hinterberger, O. Scherzer, C. Schnrr, and J. Weickert. Analysis of optical flow models in the framework of the calculus of variations. **Numerical Functional Analysis and Optimization**, 23:69–89, 2002.

S. Hong, N. A. Setiawan, and C. W. Lee. Human-robot interaction using context awareness and active plane model. In **Proc. International Joint Conference SICE-ICASE**, pages 4770–4775, 2006. doi: 10.1109/SICE.2006.314787.

B. K. P. Horn and B. G. Schunck. Determining optical flow. **Artificial Intelligence**, 17:185–203, 1981.

T. Hosaka, T Kobayashi, and N. Otsu. Image segmentation using map-mrf estimation and support vector machine. **Interdisciplinary Information Sciences**, 13:33–42, 2007.

C. Hu, M. Q. Meng, P. X. Liu, and X. Wang. Visual gesture recognition for human-machine interface of robot teleoperation. In **Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)**, volume 2, pages 1560–1565 vol.2, 2003. doi: 10.1109/IROS.2003.1248866.

M. K. Hu. Visual pattern recognition by moment invariants. **IRE Transactions on Information Theory**, 8(2):179–187, 1962. ISSN 0096-1000. doi: 10.1109/TIT.1962.1057692.

G. Hua and Y.Yes Wu. Multi-scale visual tracking by sequential belief propagation. In **Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, 2004.

G. Iannizzotto, C. Costanzo, F. La Rosa, and P. Lanzafame. A multimodal perceptual user interface for video-surveillance environments. In **ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces**, pages 45–52, New York, NY, USA, 2005. ACM. ISBN 1-59593-028-0. doi: http://doi.acm.org/10.1145/1088463.1088475.

M. Isard and A. Blake. Condensation—conditional density propagation for visual tracking. **International Journal of Computer Vision**, 29(1):5–28, 1998a.

M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. **Computer Vision, 1998. Sixth International Conference on**, pages 107–112, 1998b.

Y. Iwai, H. Shimizu, and M. Yachida. Real-time context-based gesture recognition using hmm and automaton. In **Proc. International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems**, pages 127–134, 1999. doi: 10.1109/RATFG.1999.799235.

H. Je, J. Kim, and D. Kim. Vision-based hand gesture recognition for understanding musical time pattern and tempo. In **Proc. 33rd Annual Conference of the IEEE Industrial Electronics Society IECON 2007**, pages 2371–2376, 2007a. doi: 10.1109/IECON.2007.4460080.

H. Je, J. Kim, and D. Kim. Hand gesture recognition to understand musical conducting action. In **Proc. 16th IEEE International Symposium on Robot and Human interactive Communication RO-MAN 2007**, pages 163–168, 2007b. doi: 10.1109/ROMAN.2007.4415073.

M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. **International Journal of Computer Vision**, 46(1):81–96, 2002. URL `citeseer.ist.psu.edu/jones99statistical.html`.

C. Joslin, A. El-Sawah, Q. Chen, and N. D. Georganas. Dynamic gesture recognition. In **IMTC-Instrumentation and Measurement Technology Conference**, 2005.

B. H. Juang and L. R. Rabiner. Hidden markov models for speech recognition. **Technometrics**, 33:251–272, 1991.

S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In **Proceedings of the Int. Symp. Aerospace/Defense Sensing, Simul. and Controls**, 1997.

E. Kaiser, A. Olwal, D. McGee, H. Benko, A. Corradini, X. Li, P. Cohen, and S. Feiner. Mutual disambiguation of 3d multimodal interaction in augmented and virtual reality. In **ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces**, pages 12–19, New York, NY, USA, 2003. ACM. ISBN 1-58113-621-8. doi: http://doi.acm.org/10.1145/958432.958438.

R. E. Kalman. A new approach to linear filtering and prediction problems. **Journal of Basic Engineering**, 82:35–45, 1960.

S. P. Kang, G. Rodnay, M. Tordon, and J. Katupitiya. A hand gesture based virtual interface for wheelchair control. In **Proc. IEEE/ASME International Conference on Advanced Intelligent Mechatronics AIM 2003**, volume 2, pages 778–783 vol.2, 2003. doi: 10.1109/AIM.2003.1225441.

M. Karam. **A framework for research and design of gesture-based human-computer interactions**. PhD thesis, University of Southampton, 2006.

Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. **Proc. CVPR**, 2:506–513, 2004.

S. Kettebekov, M. Yeasin, and R. Sharma. Prosody based audiovisual coanalysis for coverbal gesture recognition. **Multimedia, IEEE Transactions on**, 7(2):234–242, 2005. ISSN 1520-9210. doi: 10.1109/TMM.2004.840590.

J. S. Kim, W. Jang, and Z. Bien. A dynamic gesture recognition system for the korean sign language (ksl). **Systems, Man, and Cybernetics, Part B, IEEE Transactions on**, 26: 354–359, 1996.

M. Kölsch. **Vision Based Hand Gesture Interfaces for Wearable Computing and Virtual Environments.** PhD thesis, University of California at Santa Barbara, August 2004.

M. Kölsch and M. Turk. Hand tracking with flocks of features. In **Computer Vision and Pattern Recognition. CVPR 2005. IEEE Computer Society Conference on**, 2005a.

M. Kölsch and M. Turk. Flocks of features for tracking articulated objects. **Real-Time Vision for Human-Computer Interaction**, 2005b.

C. Kotropoulos and I. Pitas. Segmentation of ultrasonic images using support vector machines. **Pattern Recognition Letters**, 24:715–727, 2002.

U. Köthe. **Visualization and Processing of Tensor Fields**, chapter Low-level Feature Detection Using the Boundary Tensor, pages 63–79. Springer Berlin Heidelberg, 2006.

T. Kurita and S. Hayamizu. Gesture recognition using hlac features of parcor images and hmm based recognizer. In **Proc. Third IEEE International Conference on Automatic Face and Gesture Recognition**, pages 422–427, 1998. doi: 10.1109/AFGR.1998.670985.

P. Lee, D. Cheok, S. James, L. Debra, W. Jie, W. Chuang, and F. Farbiz. A mobile pet wearable computer and mixed reality system for human-poultry interaction through the internet. **Personal Ubiquitous Comput.**, 10(5):301–317, 2006. ISSN 1617-4909. doi: http://dx.doi.org/10.1007/s00779-005-0051-6.

S. W. Lee. Automatic gesture recognition for intelligent human-robot interaction. In **Proc. 7th International Conference on Automatic Face and Gesture Recognition FGR 2006**, pages 645–650, 2006. doi: 10.1109/FGR.2006.25.

T. Lee and T. Hollerer. Hybrid feature tracking and user interaction for markerless augmented reality. In **Proc. IEEE Virtual Reality Conference VR '08**, pages 145–152, 2008. doi: 10.1109/VR.2008.4480766.

J. C. Lementec and P. Bajcsy. Recognition of arm gestures using multiple orientation sensors: gesture classification. In **Proc. 7th International IEEE Conference on Intelligent Transportation Systems**, pages 965–970, 2004. doi: 10.1109/ITSC.2004.1399037.

R. H. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In **Proceedings. Third IEEE International Conference on Automatic Face and Gesture Recognition.**, 1998a.

R. H. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In **Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on**, 1998b.

R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In **ICIP02**, 2002.

D. G. Lowe. Object recognition from local scale-invariant features. In **Proceedings of the International Conference on Computer Vision**, 1999.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In **Proceedings of Imaging Understanding Workshop**, 1981.

J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. **Proc. ECCV**, 2:3–19, 2000.

E. Maggio and A. Cavallaro. Hybrid particle filter and mean shift tracker with adaptive transition model. **Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on**, 2, 2005.

D. Majoe, S. Schubiger, A. Clay, and S. M. Arisona. Sqeak: A mobile multi platform phone and networks gesture sensor. In **Proc. 2nd International Conference on Pervasive Computing and Applications ICPCA 2007**, pages 699–704, 2007. doi: 10.1109/ICPCA.2007.4365533.

F. Malric. Artificial neural network based optical hand posture recognition using a color-coded glove. Master's thesis, University Of Ottawa, 2008.

F. Malric, A. El Saddik, and N.D. N. D. Georganas. Artificial neural networks for real-time optical hand posture recognition using a color-coded glove. In **Computational Intelligence for Measurement Systems and Applications, 2008. CIMSA 2008. 2008 IEEE International Conference on**, 2008.

S. Manay, D. Cremers, B.-W. Hong, A. Yezzi, and S. Soatto. Integral invariants for shape matching. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 28(10): 1602–1618, October 2006.

B. S. Manjunath, J. R. Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. **Circuits and Systems for Video Technology, IEEE Transactions on**, 11(6):703–715, 2001.

P. Menezes, F. Lerasle, and J. Dias. Visual tracking modalities for a companion robot. In **Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems**, pages 5363–5368, 2006. doi: 10.1109/IROS.2006.282099.

C.H. Messom and A.L.C. Barczak. Fast and efficient rotated haar-like features using rotated integral images. In **Australian Conference on Robotics and Automation (ACRA2006)**, 2006.

S. Mitra and T. Acharya. Gesture recognition: A survey. **Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on**, 37(3):311–324, May 2007. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.893280.

K. R. Müller, G. Rätsch S. Mika, . Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. **IEEE Transaction on Neural Networks**, 12:181–201, 2001.

J. A. Montero and L. E. Sucar. A decision-theoretic video conference system based on gesture recognition. In **Proc. 7th International Conference on Automatic Face and Gesture Recognition FGR 2006**, pages 387–392, 2006. doi: 10.1109/FGR.2006.7.

K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In **CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems**, pages 237–242, New York, NY, USA, 1991. ACM. ISBN 0-89791-383-3. doi: http://doi.acm.org/10.1145/108844.108900.

J.C. Nascimento and J.S. Marques. Robust shape tracking in the presence of cluttered background. **Multimedia, IEEE Transactions on**, 6(6):852–861, 2004.

C. L. Nehaniv. Classifying types of gesture and inferring intent. In **AISB'05 Symposium on Robot Companions: Hard Problems and Open Challenges in Robot-Human Interaction, in Proc.**, 2005.

C. W. Ng and S. Ranganath. Gesture recognition via pose classification. **icpr**, 03:3703, 2000. doi: http://doi.ieeecomputersociety.org/10.1109/ICPR.2000.903641.

J. Nocedal and S. J. Wright. **Numerical Optimization (2nd ed)**. Springer-Verlag Berlin-New York, 2006.

E.J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. **Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on**, pages 889–894, 2004.

S. Osher and R. Fedkiw. **Level Set Methods and Dynamic Implicit Surfaces**. Springer, 2003.

S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton–jacobi formulations. **Journal of Computational Physics**, 79:12–49, 1988a.

S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. **Journal of Computational Physics**, 79:12–49, 1988b. URL `citeseer.ist.psu.edu/osher88fronts.html`.

S.J. Osher and N. Paragios. **Geometric Level Set Methods in Imaging, Vision, and Graphics**. Springer, 2003.

N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 22(3):266–280, 2000.

V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 19:677–695, 1997.

M. Piccardi. Background subtraction techniques: a review. In **Systems, Man and Cybernetics, 2004 IEEE International Conference on**, 2004.

R. Plankers and P. Fua. Articulated soft objects for multiview shape and motion capture. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 25(9):1182–1187, 2003.

L. R. Rabiner. A tutorial on hidden markov models and selected applications inspeech recognition. In **Proceedings of the IEEE**, 1989.

L. R. Rabiner and B. Juang. An introduction to hidden markov models. **ASSP Magazine, IEEE**, 3:4–16, 1986.

S. Rajko, Gang Qian, T. Ingalls, and J. James. Real-time gesture recognition with minimal training requirements and on-line learning. In **Proc. IEEE Conference on Computer Vision and Pattern Recognition CVPR '07**, pages 1–8, 2007. doi: 10.1109/CVPR.2007.383330.

V. S. Rao and C. Mahanta. Gesture based robot control. In **Proc. Fourth International Conference on Intelligent Sensing and Information Processing ICISIP 2006**, pages 145–148, 2006. doi: 10.1109/ICISIP.2006.4286082.

J. M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. **Proc. European Conference on Computer Vision**, 2:35–46, 1994.

T. H. Reiss. The revised fundamental theorem of moment invariants. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 13:830–834, 1991.

Y. Sato, K. Oka, H. Koike, and Y. Nakanishi. Video-based tracking of user's motion for augmented desk interface. In **Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition**, pages 805–809, 2004. doi: 10.1109/AFGR.2004.1301633.

B. Schölkopf and A. J. Semola. **Learning with Kernels**. Cambridge MA: MIT Press, 2001.

J. A. Sethian. Level set methods: An act of violence. **American Scientist**, 85(3), 1997.

M. Shah. Motion-based recognition: A survey. **Image and Vision Computing**, 13:129–155, 1995.

R. Sharma, J. Cai, S. Chakravarthy, I. Poddar, and Y. Sethi. Exploiting speech/gesture co-occurrence for improving continuous gesture recognition in weather narration. In **Proc. Fourth IEEE International Conference on Automatic Face and Gesture Recognition**, pages 422–427, 2000. doi: 10.1109/AFGR.2000.840669.

C. Shen, H. Li, and M.J. Brooks. Classification-based likelihood functions for bayesian tracking. **Proceedings of the IEEE International Conference on Video and Signal Based Surveillance**, 2006.

J. Shi and C. Tomasi. Good features to track. **Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on**, pages 593–600, 1994.

Y. Shi. **Object-based Dynamic Imaging with Level Set Methods**. PhD thesis, Boston University, 2005.

Y. Shi and W. Karl. A real-time algorithm for the approximation of level-set-based curve evolution. **IEEE Transactions on Image Processing**, 17:645–655, 2008.

Y. Shi and W. C. Karl. Real-time tracking using level sets. In **Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**, 2005a.

Y. Shi and W. C. Karl. A fast level set method without solving pdes. In **Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on**, 2005b.

L. Sigal, S. Sclaroff, and V. Athitsos. Skin color-based video segmentation under time-varying illumination. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 26 (7):862–877, July 2004.

M. Singh, M. Mandal, and A. Basu. Visual gesture recognition for ground air traffic control using the radon transform. In **Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)**, pages 2586–2591, 2005a. doi: 10.1109/IROS.2005.1545408.

R. Singh, B. Seth, and U. B. Desai. Vision based gui for interactive mobile robots. In **IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces**, pages 254–256, New York, NY, USA, 2005b. ACM. ISBN 1-58113-894-6. doi: http://doi.acm.org/10.1145/1040830.1040887.

Y. Sriboonruang, P. Kumhom, and K. Chamnongthai. Visual hand gesture interface for computer board game control. In **Proc. IEEE Tenth International Symposium on Consumer Electronics ISCE '06**, pages 1–5, 2006. doi: 10.1109/ISCE.2006.1689527.

B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. Buhmann. Topology free hidden markov models: Application to background modeling. **IEEE International Conference on Computer Vision**, 1:294–301, 2001.

M. C. Su. A fuzzy rule-based approach to spatio-temporal hand gesture recognition. **Systems, Man, and Cybernetics Part C, IEEE Transactions on**, 30(2):276–281, 2000. ISSN 1094-6977. doi: 10.1109/5326.868448.

M. Thida, K. L. Chan, and H. L. Eng. An improved real-time contour tracking algorithm using fast level set method. **Lecture Notes on Computer Science**, 4319:702–711, 2006.

C. Tomasi, S. Petrov, and A. Sastry. 3d tracking = classification + interpolation. In **Ninth IEEE International Conference on Computer Vision (ICCV'03)**, 2003.

T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto. Making good features track better. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR'98**, pages 178–183, Santa Barbara, CA, June 1998. IEEE Computer Society Press.

M. Tosas and L. Bai. Tracking tree-structured articulated objects using particle interpolation. In **Computer Graphics and Imaging**, 2007.

M. Tosas and B. Li. **Computer Vision in Human-Computer Interaction**, chapter Virtual Touch Screen for Mixed Reality, pages 48–59. Springer Berlin/Heidelberg, 2004.

C. M. Travieso, J. B. Alonso, and M. A. Ferrer. Sign language to text by svm. In **Proc. Seventh International Symposium on Signal Processing and Its Applications**, volume 2, pages 435–438 vol.2, 2003. doi: 10.1109/ISSPA.2003.1224907.

E. Tse, C. Shen, S. Greenberg, and C. Forlines. Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. In **AVI '06: Proceedings of the working conference on Advanced visual interfaces**, pages 336–343, New York, NY, USA, 2006. ACM. ISBN 1-59593-353-0. doi: http://doi.acm.org/10.1145/1133265.1133336.

V. N. Vapnik. **Statistical Learning Theory**. Wiley-Interscience, 1998.

V. N. Vapnik. An overview of statistical learning theory. **IEEE Transactions on Neural Networks**, 10:988–999, 1999.

V. N. Vapnik. **The Nature of Statistical Learning Theory**. Springer, 2000.

R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (gpca). **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 27(12):1945–1959, 2005.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In **Computer Vision and Pattern Recognition, IEEE Computer Society Conference on**, 2001a.

P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. **International Journal of Computer Vision**, 63(2):153–161, 2005.

Paul Viola and Michael Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. pages 1311–1318, 2001b.

O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In **Proceedings of the Ninth IEEE International Conference on Computer Vision**, 2003.

O. Williams, A. Blake, and R. Cipolla. Sparse bayesian learning for efficient visual tracking. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 27:1292, 2005.

A. D. Wilson and A. F. Bobick. Parametric hidden markov models for gesture recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 21(9):884–900, Sept. 1999. doi: 10.1109/34.790429.

R. Withtaker. A level-set approach to 3d reconstruction from range data. **International Journal of Computer Vision**, 29:203–231, 1998.

P. Wu, B. S. Manjunath, S. Newsam, and HD Shin. A texture descriptor for browsing and similarity retrieval. **Signal Processing: Image Communication**, 16(1-2):33–43, 2000.

Y. Wu, G. Hua, and T. Yu. Tracking articulated body by dynamic markov network. **Computer Vision, 2003. Proceedings. Nineth IEEE International Conference on**, pages 1094–1101, 2003.

Y. Wu, J. Lin, and T. S. Huang. Analyzing and capturing articulated hand motion in image sequences. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 27 (12):1910–1922, 2005.

Y. Yao and M. L. Zhu. Hand tracking in time-varying illumination. In **Proc. International Conference on Machine Learning and Cybernetics**, volume 7, pages 4071–4075 vol.7, 2004.

Y. Yao, M. Zhu, Y. Jiang, and G. Lu. A bare hand controlled ar map navigation system. In **Proc. IEEE International Conference on Systems, Man and Cybernetics**, volume 3, pages 2635–2639 vol.3, 2004. doi: 10.1109/ICSMC.2004.1400728.

A. Yilmaz, X. Li, and M. Shah. Object contour tracking using level sets. **Proc. of 6th Asian Conference on Computer Vision (ACCV)**, 2004.

Y. Yuan, Y. Ying, and K. L. Barner. Tactile gesture recognition for people with disabilities. In **Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)**, volume 5, pages v/461–v/464 Vol. 5, 2005. doi: 10.1109/ICASSP.2005.1416340.

C.T. Zahn and R.Z. Roskies. Fourier descriptors for plane close curves. **IEEE Transactions on Computers**, C-21:269–281, 1972.

D. Zhou, Y. Wang, and X. Chen. **Technologies for E-Learning and Digital Entertainment**, chapter Hand Contour Tracking Using Condensation and Partitioned Sampling, pages 343–352. Springer Berlin / Heidelberg, 2008.

H. Zhou, L. Xie, and X. Fang. Visual mouse: Sift detection and pca recognition. In **Proc. International Conference on Computational Intelligence and Security Workshops CISW 2007**, pages 263–266, 2007. doi: 10.1109/CISW.2007.4425487.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo