

DYNAMIC HAND GESTURES RECOGNITION FOR HUMAN COMPUTER INTERFACES

Teză destinată obținerii
titlului științific de doctor inginer
la
Universitatea Politehnica Timișoara
în domeniul INGINERIE ELECTRONICĂ ȘI
TELECOMUNICAȚII
de către

Ing. Gheorghe Daniel Popa

Conducător științific: prof.univ.dr.ing. Marius Oteșteanu
Referenți științifici: prof.univ.dr.ing. Franz Quint
prof.univ.dr.ing. Corneliu Rusu
prof.univ.dr.ing. Vasile Gui

Ziua susținerii tezei: 16.01.2015

Seriile Teze de doctorat ale UPT sunt:

- | | |
|---|--|
| 1. Automatică | 9. Inginerie Mecanică |
| 2. Chimie | 10. Știința Calculatoarelor |
| 3. Energetică | 11. Știința și Ingineria Materialelor |
| 4. Ingineria Chimică | 12. Ingineria sistemelor |
| 5. Inginerie Civilă | 13. Inginerie energetică |
| 6. Inginerie Electrică | 14. Calculatoare și tehnologia informației |
| 7. Inginerie Electronică și Telecomunicații | 15. Ingineria materialelor |
| 8. Inginerie Industrială | 16. Inginerie și Management |

Universitatea Politehnica Timișoara a inițiat seriile de mai sus în scopul diseminării expertizei, cunoștințelor și rezultatelor cercetărilor întreprinse în cadrul școlii doctorale a universității. Seriile conțin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susținute în universitate începând cu 1 octombrie 2006.

Copyright © Editura Politehnica – Timișoara, 2015

Această publicație este supusă prevederilor legii dreptului de autor. Multiplicarea acestei publicații, în mod integral sau în parte, traducerea, tipărirea, reutilizarea ilustrațiilor, expunerea, radiodifuzarea, reproducerea pe microfilme sau în orice altă formă este permisă numai cu respectarea prevederilor Legii române a dreptului de autor în vigoare și permisiunea pentru utilizare obținută în scris din partea Universității Politehnica Timișoara. Toate încălcările acestor drepturi vor fi penalizate potrivit Legii române a drepturilor de autor.

România, 300159 Timișoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Foreword

The present PhD thesis is the result of my PhD research conducted in the Department of Communications at the Politehnica University of Timisoara and – during the Erasmus mobility in the summer semester of 2013 – at the Faculty of Electrical Engineering and Information Technology of the University of Applied Sciences in Karlsruhe, Germany.

First of all, I would like to address special thanks to my PhD advisor, Prof. Dr. Ing. Marius Ottesteanu for the competent and careful guidance of my entire activity as a PhD student and for all the support offered in overcoming the obstacles encountered in my research activity.

I want to thank Prof. Dr. Ing. Vasile Gui for all the constructive discussions about my work, for pointing directions of interest for my research and for giving me the opportunity to work in the team of the research project “Statistic and Semantic Modeling in Image Sequences Analysis”.

Distinguished thanks go to Prof. Dr.-Ing. Franz Quint for all the support offered during my PhD stage at the University of Applied Sciences in Karlsruhe in the summer semester of 2013, and for all the advice and encouragements he offered me since for finalizing my PhD thesis.

I want to thank the members of the advisory committee, Prof. Dr. Ing. Florin Alexa, Prof. Dr. Ing. Aldo De Sabata and Prof. Dr. Ing. Catalin Căleanu for all the constructive discussions and for the suggestions they made during our meetings.

I also want to address thanks to the members of the evaluation committee for accepting to evaluate my thesis and for all the useful suggestions they made.

Last but not least, I want to thank my wife Ingrid and my family for offering me understanding and constant support.

Timișoara, January 2015

Gheorghe Daniel Popa

To my family.

Popa, Gheorghe Daniel

Dynamic Hand Gestures Recognition for Human Computer Interfaces

Teze de doctorat ale UPT, Seria 7, Nr. 78, Editura Politehnica, 2015, 124 pagini, 42 figuri, 5 tabele.

ISSN: 1842-7014

ISBN: 978-606-554-899-2

Cuvinte cheie:

Human Computer Interaction, dynamic gestures recognition, robust methods, video tracking, tracking initialization, mean shift

Rezumat,

This thesis focuses on a subject of major interest for the computer vision research community: the human computer interaction. The main problem to solve is to allow human subjects to interact with computers in a natural way, without requiring important computational and storage resources or expensive equipment. The thesis presents new solutions related to the tracking of fingers/hands in video sequences and the recognition of dynamic hand gestures based on trajectory.

On the tracking related side, new solutions are proposed for the tracking initialization and for the tracking process itself. Multiple features like color, foreground, shape and size are used within a finite state machine to provide a safe tracking initialization with few spatial and temporal constraints. The same basic features are integrated to generate a sparse representation using line strips, followed by multiple cascaded clustering and filtering stages of the new features leading to finger/hand identification and localization. The proposed tracking solution proved robustness in many challenging situations including motion blur, scale changes, fast motion and occlusions.

The gesture recognition contributions include the thoughtful choice of the gesture alphabet – the gestures are successions of line strokes – and solutions for the processing of trajectory and gesture recognition. The median filter and mean shift clustering are used for trajectory segmentation, leading to a symbolic representation of the trajectory, which reduces the task of gesture recognition to a comparison of the symbolic representations of the trajectory and gesture prototypes.

The results evaluations indicate that the newly proposed solutions achieve the pursued goal – real-time running capability, using inexpensive hardware systems – without important sacrifices in terms of naturalness, precision and robustness.

CONTENTS

Abbreviations	7
List of Tables	8
List of Figures	9
1. Introduction	10
1.1. Preliminary information	10
1.2. Motivation	11
1.3. Thesis overview	12
1.3.1. Introduction	12
1.3.2. Non Parametric Density Estimation	12
1.3.3. Video Tracking	12
1.3.4. Robust Integration of Multiple Features for Hand/Finger Tracking	13
1.3.5. Real-time Dynamic Hand Gesture Recognition	13
1.3.6. Conclusions and Future Developments	13
1.4. List of published papers related to the present thesis	13
2. Non Parametric Density Estimation	15
2.1. Fundamentals of statistical analysis	16
2.1.1. Characterization of univariate and multivariate data	16
2.1.2. Multivariate data	18
2.2. Non-parametric estimation and basic non-parametric estimators	20
2.2.1. Histograms	21
2.2.2. Frequency polygons	23
2.2.3. Averaged Shifted Histograms (ASH)	23
2.3. Kernel density estimation	25
2.3.1. Parameters that affect the quality of kernel based estimation	27
2.3.2. Criteria for assessing the quality of estimation	28
2.3.3. The optimal kernel	32
2.3.4. The optimal bandwidth	33
2.3.5. Bandwidth selection methods	33
2.3.6. Kernel density estimation for multivariate data	39
2.3.7. Adaptive smoothing	40
2.4. Conclusions	41
3. Video Tracking	43
3.1. Target representation and localization	44
3.1.1. Probability density estimation using kernel based operators	45
3.1.2. Estimating the gradient of the probability density function	46
3.1.3. The mean shift algorithm	46
3.1.4. The mean shift algorithm in video tracking	47
3.2. Filtering and data association	52
3.2.1. Kalman filters	53
3.2.2. Particle filters	54
3.3. Hand tracking solutions	56
3.4. Conclusions	57
4. Robust Integration of Multiple Features for Hand/Finger Tracking	58
4.1. Features used in hand/finger tracking	59
4.1.1. Color	60
4.1.2. Background subtraction	63
4.2. Tracking initialization	64
4.2.1. Conditions for hand/finger detection	65
4.2.2. The hand detection algorithm	66

6 Contents

4.2.3. Tracker initialization	68
4.2.4. Implementation details	69
4.3. Finger detection	72
4.4. Target identification	79
4.5. Tracker guiding and trajectory recording	80
4.6. Performance evaluation	82
4.6.1. Tracking initialization	82
4.6.2. Tracking	85
5. Real-Time Dynamic Hand Gesture Recognition	92
5.1. Trajectory based gesture recognition	94
5.2. Hand trajectory processing	95
5.2.1. Trajectory recording and smoothing	96
5.2.2. Trajectory segmentation	96
5.2.3. Feature extraction	99
5.3. Gesture recognition	101
5.4. Results and discussions	103
6. Conclusions and Future Developments	110
6.1. Review of the original contributions	111
6.2. Future research and development directions	112
Bibliography	114

ABBREVIATIONS

AMISE – Asymptotic Mean Integrated Squared Error
ASH – Averaged Shifted Histograms
ASIR – Auxiliary Sample Importance Resampling
BCV – Biased Cross-Validation
CAMSHIFT – Continuously Adaptive Mean Shift
CIE – fr. *Commission Internationale de l'Eclairage*
CMY – Cyan, Magenta, Yellow
CRT – Cathode Ray Tube
EKF – Extended Kalman Filter
fps – frames per second
GMM – Gaussian Mixture Model
HCI – Human Computer Interaction
HMM – Hidden Markov Model
HSV – Hue, Saturation, Value
ISB – Integrated Squared Bias
ISE – Integrated Squared Error
IV – Integrated Variance
LSCV – Least Squares Cross-Validation
MISE – Mean Integrated Squared Error
MSE – Mean Squared Error
OpenCV – Open Computer Vision library
pdf – probability density function
RGB – Red, Green, Blue
RPF – Regularized Particle Filter
SCV – Smoothed Cross-Validation
SIS – Sequential Importance Sampling
SIR – Sampling Importance Resampling
SVM – Support Vector Machine
UCV – Unbiased Cross-Validation
UKF – Unscented Kalman Filter

LIST OF TABLES

Table 2.1. Widely used second order kernels.....	26
Table 4.1. Summary of the initialization results.....	83
Table 4.2. Tracking Results Comparison.....	90
Table 5.1. Codes associated to angle directions.....	99
Table 5.2. Gestures examples with corresponding symbolic representation.....	102

LIST OF FIGURES

Fig. 2.1 Graphical representation of a histogram.....	21
Fig. 2.2. Generating a frequency polygon.	23
Fig. 2.3. The averaging of shifted histograms.....	24
Fig. 2.4. The influence of the kernel bandwidth over the quality of the estimate. ...	27
Fig. 2.5. Estimates obtained with triangle (a) and normal kernels (b, c).	28
Fig. 3.1. Block diagram of a video tracking system.....	43
Fig. 3.2. Target localization using the mean shift algorithm.	51
Fig. 4.1. The main processing stages of a vision-based gesture recognition	58
Fig. 4.2. Conic representation of the HSV color space.	61
Fig. 4.3. Screen capture – user tries to initialize the tracking process.	66
Fig. 4.4. State machine diagram.	68
Fig. 4.5. Flowchart of frame processing for hand detection.	69
Fig. 4.6. Circular representation of the 8-bit hue.	70
Fig. 4.7. Binary images after color space analysis in the hand mask region.	70
Fig. 4.8. Matching pixels in the rectangular hand mask region.	71
Fig. 4.9. Line strip features used for finger detection.....	73
Fig. 4.10. The layers of the tracking algorithm.	73
Fig. 4.11. The ρ , θ parameters of the line segment	74
Fig. 4.12. The pair line strips used to determine the θ value.....	75
Fig. 4.13. The search for attached hand procedure.....	77
Fig. 4.14. Histograms of global matching percentages	82
Fig. 4.15. Histogram of the number of frames in the CONFIRM state	84
Fig. 4.16. Tracking under severe motion blur	85
Fig. 4.17. Tracking results when the finger is partially occluded	86
Fig. 4.18. Finger moving away from the camera	87
Fig. 4.19. Tracking the finger at fast motion speed:	88
Fig. 4.20. Tracking the finger in the neighborhood of other skin colored objects. ...	89
Fig. 4.21. Tracking results when the finger is partially occluded:	89
Fig. 5.1. Gesture segmentation state machine.....	96
Fig. 5.2. Dealing with the circular definition of the angle	99
Fig. 5.3. Angle classes.	99
Fig. 5.4. Effect of global rotation correction.....	100
Fig. 5.5. Sequential gesture discrimination.	103
Fig. 5.6. The CamShift tracker.	104
Fig. 5.7. Trajectory of a 3-strokes gesture.	105
Fig. 5.8. Angle histogram for 2-stroke gesture.	105
Fig. 5.9. Angle histogram for 3-stroke gesture.	106
Fig. 5.10. Angle histogram for 4-stroke gesture.....	106
Fig. 5.11. Histogram of the standard deviations over a set of 70 strokes.....	107
Fig. 5.12. Original and filtered angle sequences plot.....	107
Fig. 5.13. Segmented trajectory and extracted features.....	108
Fig. 6.1. Hierarchical overview of the main original contributions.	111

1. INTRODUCTION

1.1. Preliminary information

The continuous development of computers and in general of the systems using microprocessors/microcontrollers or digital signal processors during the last decades conducted to the expansion of these systems in almost all aspects of modern life. In this context, human subjects need to interact very often with processor-based systems and the need has arisen for developing new interfaces, in order to facilitate the Human-Computer Interaction (HCI). Traditional input devices like mice, keyboards, touchpads or touchscreens do not provide natural interfaces. Gestures represent an important communication means between people and can provide more natural communication solutions between humans and computers than traditional interfaces. Gestures are expressive (meaningful) motions of the human body, which contain spatial and/or temporal variation. Recently, more and more research in the field of Human Computer Interaction (HCI) focuses on developing gesture-based interfaces.

A very popular approach for gesture-based HCI relies on devices that visually track the movements of the user. Gesture recognition systems based on tracking the user's movements usually involve 3 main processing stages: image preprocessing, tracking of the human body parts of interest (e.g. hands, face) from frame to frame and recognition of meaningful gestures. Each of the 3 stages may involve complex operations, whose implementations for running in real time represent real challenges – especially on systems with limited computational resources.

Image preprocessing is required for removing noise from the images acquired with a camera and extracting meaningful data from the images for the tracking algorithm. Besides filtering, color space conversions and low-level feature extraction are common operations implemented at this processing stage. Color space conversions may be necessary, in order to facilitate the extraction of some features, while feature extraction itself scans the image for meaningful features that may help in identifying the target (the human body part to track) and represents them in a symbolic format.

The goal of vision based tracking is to identify the position of the target in each frame – based on the features provided by the image preprocessing stage – and to extract information about the movement of the object of interest from frame to frame.

The gesture recognition stage processes the data provided by the tracking stage, decides whether the user is performing a meaningful gesture and – if this is the case – it identifies the gesture.

The present thesis presents contributions in each of the above described processing stages. The contribution in the field of image preprocessing is represented by the thoughtful choice of the features which provide the basis for a robust tracking algorithm. In the field of vision based tracking, the contributions include a secure semi-automatic target initialization method and a multilayer

tracking algorithm with reduced computational complexity. For the topmost processing stage – the gesture recognition – the contributions are represented by the method to define the gesture set and the trajectory processing algorithm which, together, ease very much the gesture classification process.

1.2. Motivation

The goal of this thesis is to offer new solutions for use in the implementation of gesture based human-computer interfaces. The target gesture based interfaces need to be designed in such a way as to allow implementations on computers with medium or low computational and storage resources available and without the need for expensive auxiliary devices (e.g. data gloves, Kinect, time of flight, stereo vision cameras).

The wide spreading of webcams during the last years, based on their decreasing prices, concurrently with the constant improvement of the quality of images they can provide, has led to the ubiquitous availability of these devices with modern processor based systems (e.g. laptops, desktop computers, tablets, smartphones etc.). To date, the most popular application of webcams is to provide video communications over the Internet, but their wide availability, together with the significant improvement of image quality in the last years, make them an appealing choice for usage as input devices in HCI.

The subject of gesture recognition based on images acquired using a single camera has attracted researchers since the 90's. However, most available solutions are either not robust enough, or require important computational and/or storage resources, which are not available on common processor-based systems, or require high quality images. Despite the fact that the quality of images provided by webcams increased significantly in the last years, the images and video sequences acquired with such devices are still relatively noisy and the frame rate is low (12 – 15 fps). Motion blur often occurs in video sequences acquired with insufficient lighting and/or when fast motion is present. These drawbacks of webcam acquired images and video sequences increase the difficulty of detecting and tracking targets, making the implementation of tracking (which is essential for gesture-based interfaces) a challenging task.

There are a few key factors, like the choice of the set of features used for detection of the target human body parts and the design of the gesture set, which have a severe impact on the performance of a HCI and its ability to run in real time on largely available computer-based systems. A good choice of the features can allow the implementation of robust tracking algorithms with reduced computational complexity. The definition of gesture sets can contribute to the simplification of the gesture recognition process, allowing efficient implementations with reduced complexity.

The first processing step in a gesture recognition system is represented by the image preprocessing. In this step, the images provided by the webcam are processed and relevant features are extracted, in order to be used for the detection of the target (i.e. finger/hand) in the next processing step. An important objective at this level is to identify the appropriate features that can offer reliable information to allow the tracking algorithm to identify the target. My choice for the feature set includes color (requires human skin color segmentation), foreground (requires background subtraction) and geometrical shape and proportionality (assessed through features derived from the results of color segmentation and background subtraction).

The tracking of the hand/finger requires to identify the target's position in each frame, based both on the data (features) obtained after the image preprocessing step and on the positions of the target in the previous frames (including data obtained from their processing). The main goal at this level is to develop a robust tracking algorithm, which should be able to run in real-time. A secondary objective is to define a tracking initialization method that can provide safe and accurate initial target detection, in order to prevent accidental triggering of the tracking and to allow building an accurate initial model of the target.

In the last processing step (i.e. gesture recognition), the trajectory provided by the tracker is processed and a decision is made whether the user has performed a meaningful gesture (including the identification of the gesture) or not. A first objective at this level is the definition of a gesture set that facilitates the recognition process and at the same time allows the natural execution of gestures. Other goals are the development of a trajectory filtering method, followed by a gesture recognition algorithm. The trajectory filtering is necessary because, generally, the trajectories provided by the tracker tend to be noisy due to a multitude of factors (e.g. camera noise, lighting changes, occlusions etc.), that influence the accuracy of target position identification.

1.3. Thesis overview

1.3.1. Introduction

After presenting short preliminary information on the subject of the thesis, the motivation that led to the writing of the present thesis is presented. Then a short overview of each chapter of the thesis is provided in the current section. Next, the list of publications containing personal contributions is presented.

1.3.2. Non Parametric Density Estimation

This chapter presents basic information about non parametric density estimation techniques. First, some fundamentals of statistical analysis are presented for univariate and multivariate data. Then, the basic non parametric density estimators (like histograms, frequency polygons and ASH) are described. Next, the Kernel density estimators are discussed, together with the parameters that affect the quality of the estimation and bandwidth selection methods. Finally, a review of the most important ideas resulting from this chapter is presented.

1.3.3. Video Tracking

After a short introduction to video tracking, the main approaches used in video tracking are detailed. The target representation and localization process is explained, together with the most representative tracking algorithms relying on it – mean-shift based algorithms. The filtering and data association process is then presented, together with tracking algorithms relying on it – algorithms using Kalman filters and algorithms using particle filters. Next, the most popular approaches used for tracking hands and fingers are shortly presented. Finally, a short review outlining the advantages and disadvantages of the tracking algorithms presented in this chapter is given.

1.3.4. Robust Integration of Multiple Features for Hand/Finger Tracking

This chapter presents the original contributions with respect to feature extraction and tracking processing stages. The chapter is mainly based on research results previously published in [1], [2], [3] and [4]. In the beginning of the chapter, the place of tracking within a gesture recognition system is outlined. Next, the low-level features used for tracking and tracking initialization are discussed, together with the methods for extracting them from image sequences. The tracking initialization method proposed in [1] is then presented, followed by descriptions of the finger detection, target identification and tracker guiding/trajectory recording processing layers used for tracking, proposed in [3] and [4]. The last section of the chapter presents the results of the performance evaluations, for both the tracking initialization solution and the tracking algorithm.

1.3.5. Real-time Dynamic Hand Gesture Recognition

This chapter presents the contributions corresponding to the gesture recognition processing stage and relies mostly on the results of research previously published in [5] and [6]. A short introduction to gesture recognition is given in the beginning of the chapter, outlining the main types of gesture recognition systems. Then, special attention is given to the known solutions for trajectory based dynamic gesture recognition and especially for dynamic hand gesture recognition. The next two sections focus on the description of the solutions proposed in [5] and [6] for processing the hand trajectory (aiming at the extraction of relevant features to allow gesture recognition) and for gesture recognition, respectively. The last section of the chapter presents the experimental results, outlining the performances of the proposed trajectory processing method and of the gesture recognition system, which can easily operate in real-time on systems with low computational resources.

1.3.6. Conclusions and Future Developments

The introductory part of this chapter presents some conclusions regarding the new tracking and gesture recognition solutions proposed in the thesis. Then, a summary of the original contributions is presented, outlining the two major categories of own contributions: tracking related (safe and precise tracking initialization; robust finger tracking algorithm) and gesture recognition related (gesture definition method; solutions for the segmentation and symbolic representation of hand trajectories; fast gesture recognition method). The last section of this chapter is dedicated to the proposals for future research and development directions.

1.4. List of published papers related to the present thesis

1. D. Popa, G. Simion, V. Gui, and M. Ottesteanu, "Trajectory based hand gesture recognition," presented at the Proceedings of the 6th WSEAS international conference on Computational intelligence, man-machine systems and cybernetics, Tenerife, Canary Islands, Spain, 2007.
2. D. Popa, G. Simion, V. Gui, and M. Ottesteanu, "Real time trajectory based hand gesture recognition," *WSEAS Transactions on Information Science and Applications*, vol. 5, pp. 532-546, 2008.
3. G. Simion, V. Gui, M. Ottesteanu, D. Popa, and C. David, "Hand Edge Detection for Gesture Analysis in a Sparse Framework," *Scientific Bulletin of the*

- "Politehnica" University of Timișoara, Romania, *Transactions on Electronics and Communications*, vol. 53(67), pp. 155-160, 2008.
4. V. Gui, G. Popa, P. Nisula, and V. Korhonen, "Finger Detection in Video Sequences Using a New Sparse Representation," *Acta Techn. Napoc.*, vol. 52, pp. 1-6, 2011.
 5. D. Popa, V. Gui, and M. Ottesteanu, "Semi-Automatic Hand/Finger Tracker Initialization for Gesture-Based Human Computer Interaction," in *Digital Information and Communication Technology and Its Applications*, vol. 166, H. Cherifi, J. Zain, and E. El-Qawasmeh, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 417-430.
 6. D. Popa, V. Gui, and M. Ottesteanu, "Real-Time Finger Tracking with Improved Performance in Video Sequences with Motion Blur," in *International Conference on Telecommunications and Signal Processing - TSP 2014*, Berlin, 2014, pp. 715-720.
 7. D. Popa, V. Gui, and M. Ottesteanu, "Real-Time Multi-Cue Finger Tracking for Human Computer Interaction," in *International Conference on Telecommunications and Signal Processing - TSP 2014*, Berlin, 2014, pp. 721-727.

2. NON PARAMETRIC DENSITY ESTIMATION

Univariate data sets are collections of measurements that evaluate the objects in a sample based on a single parameter. The analysis of such data sets is performed in a single dimension and is well developed in literature. Multivariate data sets are collections of measurements performed on multiple parameters of each object within a sample set. The measurements performed on the parameters of the object can be treated as random variables. Data sets of this type are used in many fields even if they are not always treated as multivariate data sets. The early approaches on processing multivariate data sets relied exclusively on univariate analysis techniques, each variable being analyzed apart from the others. Later, plenty of multivariate analysis methods were developed. The important advances in the development of computers in the last decades significantly reduced the costs of the computational resources and facilitated the practical implementation of the multivariate analysis techniques in more and more application fields.

While some of the multivariate analysis techniques require the use of the same measuring scale for all the variables, most of them do not set such a restriction. Usually all parameters are measured simultaneously for each object, resulting in a correlation between the variables. In most situations each variable depends on the other variables in the set, therefore separate processing of variables may not provide enough information about the analyzed objects or systems. Multivariate analysis extracts the main characteristics of the process that generates the variables through simultaneous analysis of all measured variables. Many of the multivariate analysis methods aim at "untangling" the information provided by the variables in the analyzed set by reducing the number of dimensions necessary for expressing the characteristics of the analyzed objects [7].

A complete analysis of a multivariate data set involves applying various types of statistical methods: parametric, non-parametric and graphic [8]. The parametric methods are considered to be the most powerful, but they are very sensitive to the correctness of the initial model definition. A parametric estimation starts from a known model, which depends on a parameter set, attempting to find the best estimations for the parameters. Small deviations of the analyzed data from the initial model may cause important errors in the interpretation of results in this case.

Non-parametric methods have the advantage of flexibility, as they do not rely on the assumption that the analyzed data fit a known model, trying instead to build the model using the structure of the data. The term "non-parametric" only means that the number and type of parameters are not established *a priori*, not that these methods do not have parameters. Though initially the statisticians were rather reluctant to the non-parametric methods, later the conclusion was reached that it is better to sacrifice a small part of the optimality of the parametric methods, in order to achieve the robustness of the non-parametric methods with respect to the errors in the model's specifications. The slight disadvantage of the non-parametric methods with respect to the effectiveness is outweighed by the risk reduction with respect to the misinterpreting of the data, caused by inaccurate initial model specification.

The advantages of parametric and non-parametric methods can be combined by using together a parametric and a non-parametric term, resulting in a so-called semi-parametric method.

Graphic methods allow for the visual discovery of some structures in the analyzed data (in some situations other methods of analysis are not able to identify these structures).

The next paragraph of this section presents some basics of statistical analysis of uni- and multivariate data. Then non-parametric estimation techniques are introduced, with the early estimators belonging to this category (histograms, frequency polygons and average shifted histograms, which at limit can also be regarded as kernel estimators). Later, kernel estimators are presented, with the most widely used kernels, criteria for assessing the quality of estimation, the bandwidth selection problem and considerations on the use of kernel estimation in multidimensional analysis.

2.1. Fundamentals of statistical analysis

2.1.1. Characterization of univariate and multivariate data

A random variable is defined as a mathematical function that associates real numbers to the outputs of a random process. The probability density function (pdf) indicates the relative likelihood of the random variable to take a given value. Such a function may not have negative values and $\int_{-\infty}^{+\infty} f(x)dx = 1$. The probability for the random variable to take a value in the limited interval of values $[a, b]$ can be calculated by integrating the probability density function over the interval $[a, b]$.

The mean of a random variable is defined as the average of all the values taken by the random variable and is noted with μ . The mean is also known as the expected value of the random variable, noted as $E(x)$. The mean of a sample of n observations over the values of the random variable is calculated as the arithmetical mean of the n values.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (2.1)$$

If the sample contains a large number of observations, it is very likely for its mean to be very close to the mean of the random variable, μ , but not identical to it. Though, the mean of a sample is considered a good estimator of the mean of the random variable. $E(\bar{x}) = \mu$, therefore \bar{x} is an unbiased estimator and moreover the variance of the mean of samples is smaller than in the case of using a single observation [7]. The estimator \hat{p} of a parameter p is assumed to be unbiased if $E(\hat{p}) = p$. The bias of the estimator of a parameter/function is defined as the difference between the expected value of the estimator and the real value of the parameter/function, or the expected value of the difference between the estimator and the real value:

$$bias = E(\hat{p}) - p = E(\hat{p} - p). \quad (2.2)$$

The spreading of the possible values of the random variable around the mean value can be evaluated using the variance. The variance, σ^2 , is defined as the mean of the squares of the deviations of the random variable's values from its mean, μ :

$$\sigma^2 = E(x - \mu)^2 = E(x^2) - \mu^2. \quad (2.3)$$

As opposite to the variance, the standard deviation, defined in equation (2.4) is not a quadratic value and measures the mean value of the deviations of the values of the random variable from the mean:

$$\sigma = \sqrt{E(x - \mu)^2} = \sqrt{E(x^2) - \mu^2}. \quad (2.4)$$

For a sample consisting of n observations the variance is:

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} = \frac{\sum_{i=1}^n x_i^2 - n\bar{x}^2}{n - 1} \quad (2.5)$$

The intuitive value of the denominator of the fractions in equation (2.5) should be n , but with this value the variance of the sample would no longer be an unbiased estimator of the variance σ^2 . It can be proven that $E(s^2) = \sigma^2$, which means that the variance of the sample, as defined by equation (2.5), is an unbiased estimator of the variance of the random variable.

Two random variables, which reflect the results of the measurement of two parameters which characterize the studied objects, produce an array of bivariate data. The two random variables, measuring parameters of the same objects, will tend to vary simultaneously (or *co-vary*), either in the same manner or in opposite manners. The covariance is defined by equation (2.6) as the mean of the product of the differences between the current values and the mean values of the two variables [7]:

$$\text{cov}(x, y) = \sigma_{xy} = E[(x - \mu_x)(y - \mu_y)]. \quad (2.6)$$

Considering the definition of covariance in equation (2.6), it can be noticed that a positive value of the covariance indicates a tendency of the two variables to co-vary in the same manner (both variables are on the same side of their means), while a negative value of the covariance indicates a tendency of co-varying in opposite manners (when one variable is on one side of its mean the other variable is on the opposite side of its mean).

Two random variables, x and y , are considered independent of each other if the behavior of one variable is not affected in any way by the behavior of the other variable. The covariance is an indicator of linear dependency between two variables, but it does not provide any information about non-linear dependencies. Therefore, any two independent random variables will have the covariance 0, but the opposite is not always true. If two variables have the covariance 0, they are not necessarily also independent, because non-linear dependencies may exist between the two variables without being revealed by the covariance.

Equation (2.7) defines the covariance for a sample of n measurements over the two parameters. For the same reason as in the case of the variance in the

univariate domain, the covariance of the sample defined in the following manner is an unbiased estimator of the covariance of the two random variables:

$$s_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n-1} = \frac{\sum_{i=1}^n x_i y_i - n\bar{x}\bar{y}}{n-1} \quad (2.7)$$

The covariance is a quantity which depends a lot both on the measurement units used for the parameters and the range of the parameters' values. Therefore, a comparison between the covariances of two different sets of bivariate data is actually useless. In order to enable the usefulness of such a comparison, a normalization of the covariance is necessary. The normalization is achieved by division with the standard deviations of the two variables [7]. The quantity obtained after the normalization of the covariance is also known as the correlation coefficient between the two variables and this quantity always takes values in the range $[-1, +1]$, indicating the strength of the linear dependency between the two variables. The correlation coefficient between two random variables, x and y , is defined by equation (2.8), while the correlation coefficient for a sample of n values of the bivariate data set is defined by equation (2.9):

$$\text{corr}(x, y) = \rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \quad (2.8)$$

$$r_{xy} = \frac{s_{xy}}{s_x s_y} \quad (2.9)$$

2.1.2. Multivariate data

The analysis of data sets consisting of more than two random variables is far more complex than in the uni- or bi-dimensional case. For multivariate data the term of mean value is replaced by the mean vector. The mean vector of a sample of n vectors of a p -dimensional variable x is defined in equation (2.10) as the mean of the n observed vectors, or the vector containing the mean values of the p one-dimensional variables that make up the multidimensional variable:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \cdot \\ \cdot \\ \bar{x}_p \end{pmatrix}, \quad x_i = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_p \end{pmatrix} \quad (2.10)$$

The n vectors in the multidimensional variable sample can be written as a data matrix, as in equation (2.11). The matrix contains in each column the n samples of one of the p random variables (the lines of the matrix contain the transposed vectors). This arrangement of the matrix is preferred, as generally the vectors of the sample outnumber the variables.

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{np} \end{pmatrix} \quad (2.11)$$

The mean vector of the sample can be calculated based on the data matrix using the equation:

$$\bar{x} = \frac{1}{n} X^T J, \quad J = \underbrace{(1 \ 1 \ \dots \ 1)}_{n \text{ elements}}^T \quad (2.12)$$

Similar to the one-dimensional case, the mean vector of the sample, \bar{x} , is an unbiased estimator of μ – the mean vector of the variable x [7].

The equivalents in the multidimensional domain of the covariance and correlation coefficient are the covariance matrix and the matrix of the correlation coefficients. Equation (2.13) defines the covariance matrix of a multidimensional variable, x . Considering that the expected value of a matrix is equal to the matrix of the expected values of elements, the covariance matrix contains on the main diagonal the variances of the p one-dimensional variables and in the remaining positions the covariances between all possible pairs of one-dimensional variables. The covariance between two real random variables is the same, regardless of which of the variables is considered to be the first one and which one is considered to be the second. Therefore, the covariance matrix is symmetric with respect to the main diagonal.

$$\text{cov}(x) = \Sigma = E \left[(x - \mu)(x - \mu)^T \right] = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \dots & \sigma_{pp} \end{pmatrix}, \quad \sigma_{ij} = \sigma_{ji} \quad (2.13)$$

The covariance matrix of a sample of vectors of the multidimensional variable x has the same structure as the covariance matrix of the variable, but all the values of the variances and covariances in the matrix correspond to the n values sample [7]. Equations (2.14) and (2.15) present two alternative methods for calculating the covariance matrix of the sample using the n observation vectors and the data matrix respectively:

$$S = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^T = \frac{1}{n-1} \left(\sum_{i=1}^n x_i x_i^T - n \bar{x} \bar{x}^T \right) \quad (2.14)$$

$$S = \frac{1}{n-1} \left[X^T X - X^T \left(\frac{1}{n} J \right) X \right] = \frac{1}{n-1} X^T \left(I - \frac{1}{n} J \right) X, \quad J = jj^T \quad (2.15)$$

The matrix of the correlation coefficients is defined in equation (2.16) similar to the covariance matrix, but in this case the main diagonal contains all 1s, as the correlation coefficient of a one-dimensional variable with itself is always 1:

$$P_p = \begin{pmatrix} \rho_{11} & \rho_{12} & \dots & \rho_{1p} \\ \rho_{21} & \rho_{22} & \dots & \rho_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p1} & \rho_{p2} & \dots & \rho_{pp} \end{pmatrix} = \begin{pmatrix} 1 & \rho_{12} & \dots & \rho_{1p} \\ \rho_{21} & 1 & \dots & \rho_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p1} & \rho_{p2} & \dots & 1 \end{pmatrix} \quad (2.16)$$

The matrix of the correlation coefficients for a sample has a similar structure, the correlation coefficients between variables being replaced with correlation coefficients between their samples. The correlation coefficients' matrix can be obtained from the covariance matrix and vice versa using equations (2.17) and (2.18). The matrix D_s used in these equations is a diagonal matrix containing the standard deviations of the samples of the p variables [7].

$$R = D_s^{-1} S D_s^{-1}, \quad D_s = \text{diag}(s_1, s_2, \dots, s_p) \quad (2.17)$$

$$S = D_s R D_s \quad (2.18)$$

Though the covariance and correlation coefficients' matrices provide a complete view over the manner of variation of the data, there are situations where it is useful to have a unique numeral indicator of the data spreading. Such numeric indicators are the generalized variance of the sample and total variance of the sample.

The generalized variance is the determinant of the covariance matrix and is useful for revealing possible linear dependencies between variables. If the generalized variance is 0, the covariance matrix has a null eigenvalue, indicating a linear dependency between variables. The eigenvector associated to the null eigenvalue may reveal the form of the linear dependency. Such an analysis may simplify the processing of the data, reducing the variable set to a set of linear independent variables after removal of redundant variables. A small value of the generalized variance (or – for normalized evaluation – of the determinant of the correlation coefficients' matrix) may indicate either a reduced dispersion or a so-called multi-collinearity. The term multi-collinearity indicates the existence of a quasi-linear relation within the variables set and is caused either by high values of the correlation coefficients of multiple variable pairs, or by high correlation coefficients between one variable and multiple other variables in the set.

The total variance of the sample is the sum of all the variances of the p variables. Though the total variance does not take into account the covariances between variables, it is still useful for comparisons in certain analysis techniques.

2.2. Non-parametric estimation and basic non-parametric estimators

The parametric estimation starts with a family of probability density functions, depending on a parameter set. The main goal in this case is to find the best estimator of the parameter set. One of the most widely used families of probability density functions is the family of normal densities, which are defined using the two parameters set (μ, σ^2) . The parametric estimation consists of two main steps: model specification and parameters' estimation. The parametric estimation methods focus on the second step, the first one relying mainly on the statistician's experience for choosing the most appropriate model (i.e. family of

probability density functions). Parametric methods are very effective when the family of probability density functions is well chosen, but are very sensitive to model specification errors. Minor deviations from the initially assumed conditions may turn the optimal parametric estimator into an inefficient one.

For the one-dimensional case, a multitude of already defined models are available, allowing the choice of the most appropriate one, based on the available data. In the multivariate domain, in many situations the *a priori* available data do not provide enough information for choosing a proper model. In such situations non-parametric methods can be taken into account, as they require no initial specification of a model, and focus directly on finding an estimator for the probability density function.

The most widely known and used non-parametric estimator is the histogram, which estimates the probability density. The histogram is also at the origins of two other classes of non-parametric estimators of the probability density: frequency polygons and averaged shifted histograms. Kernel estimators represent an important class of non-parametric estimators, leading to better estimations of the probability density than the histograms or histogram-based estimators.

2.2.1. Histograms

The histogram is a representation of the data in an observations sample based on splitting the sample into multiple disjoint subsets (bins) and counting the observations that fit into each subset. Multiple choices are available for the graphical representation of the histograms, depending on the application. Some authors distinguish between the histogram as a probability density estimator on one side and the histogram as an instrument for data visualization on the other side. Nevertheless, regardless of the mode of representation, a histogram provides information both on the frequency of the observations and on the relative frequency of the observations [7]. Fig. 2.1 presents the histogram of a sample consisting of 50 values, drawn from a normal distribution with the mean 100 and the standard deviation 1, and 40 values, drawn from a normal distribution with the mean 105 and

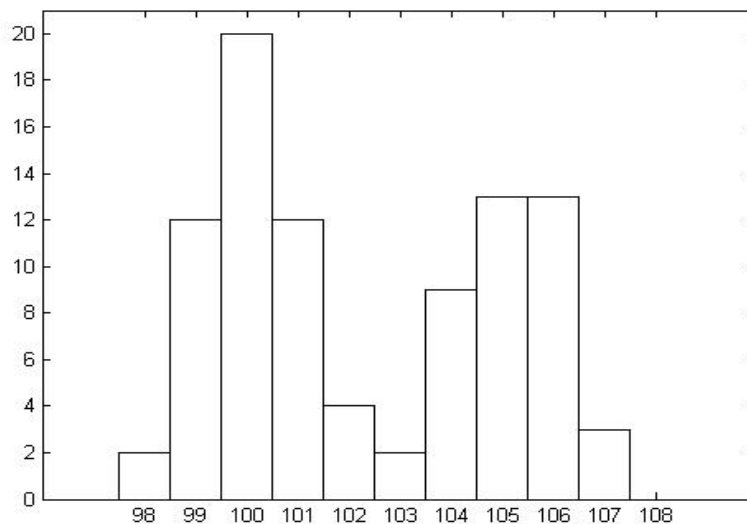


Fig. 2.1 Graphical representation of a histogram.

standard deviation 1.

Let n be the total number of observations, k – the total number of bins and m_i the number of observations in the i -th bin. The histogram must fulfill the following condition:

$$n = \sum_{i=1}^k m_i \quad (2.19)$$

Usually it is considered that the bins which compose the histogram are equally spaced non-overlapping intervals. A histogram is completely defined by two parameters: the bin width – usually noted with h – and the origin, which can be conveniently chosen amongst the bin ends and is usually noted with t_0 . The way a histogram reflects the structure of the data is influenced by both parameters, but a greater importance is given to the bin width, h . If the chosen bin width is too small, the resulting histogram has plenty of bins containing no observations and bins with only a few observations. The visual aspect of such a histogram is noisy, revealing false structures in the sample, due to the reduced number of observations. On the other hand, the choice of a too large bin width leads to an excessively smoothed histogram which hides certain structures in the observations sample (e.g. multimodality). Over time, a wide variety of rules were proposed for calculating the optimal width of the bins in a histogram. The Sturges rule, proposed in 1926, was the first rule of this kind, being in fact a rule for calculating the number of bins:

$$k = \lceil 1 + \log_2 n \rceil \quad (2.20)$$

The main drawback of the Sturges rule is its ineffectiveness when the data are not drawn from a sample with normal distribution. Another rule for normal distributions was proposed by Scott in 1979, based on the minimization of the AMISE (Asymptotic Mean Integrated Squared Error):

$$h^* \approx 3.5\sigma n^{-1/3}. \quad (2.21)$$

The standard deviation, σ , behaves as a scaling factor in equation (2.21). Scott also proposed modified versions of the rule in equation (2.21) for lognormal and t distributions. A rule with improved robustness was proposed by Freedman and Diaconis, by replacing the standard deviation in Scott's rule with a multiple of the interquartile range:

$$\hat{h} = 2(IQ)n^{-1/3}. \quad (2.22)$$

In the multivariate domain, for the histogram of a d -dimensional multivariate data set, the Scott rule becomes:

$$h_k^* \approx 3.5\sigma_k n^{-1/(2+d)}; \quad k = 1, \dots, d. \quad (2.23)$$

Histograms perform a discontinuous estimation of the probability density. The estimate obtained using the histogram has a constant value within a given bin, B_k , and in order to ensure that the estimate is a probability density function, it is necessary that the integral over its support to be 1:

$$\hat{f}(x) = \frac{V_k}{nh}, \quad x \in B_k \quad (2.24)$$

In equation (2.24), v_k represents the number of observations in the bin B_k , while the division by nh ensures that the integral of the estimate over its support is unitary.

2.2.2. Frequency polygons

Frequency polygons constitute a solution for generating a continuous estimate of the probability density. The polygons are generated using line segments to connect the mid-points of adjacent bins. Fig. 2.2 presents the frequency polygon corresponding to the histogram in Fig. 2.1.

The MISE (Mean integrated Squared Error) analysis reveals negligible effects of the choice of bins' origin for both histograms and frequency polygons. Nevertheless, through visual analysis, differences may be observed from the point of view of the structures revealed. Using the same bin width for a given set of observations with different origin points, it is possible to build histograms or frequency polygons that reveal a unimodal, a bi-modal or m -modal structure of the data. The optimal bin width for frequency polygons is substantially larger than in the case of histograms, resulting in a much larger number of possible choices for the bin origin [8].

2.2.3. Averaged Shifted Histograms (ASH)

The ASH (Averaged Shifted Histograms) represent a solution proposed by Scott [8] to the problem of choosing the origin point. The solution consists in averaging multiple histograms having the same bin width, but different origin points. The method can also be applied to frequency polygons.

According to the method proposed by Scott, m histograms, having each the bin width h and the origin points at multiples of h/m ($0, h/m, 2h/m, \dots, (m-1)h/m$), are averaged, leading to an estimated probability density of:

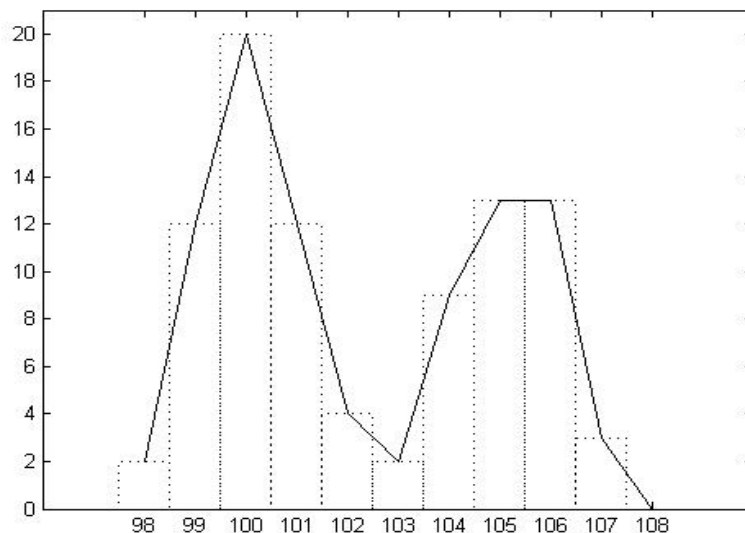


Fig. 2.2. Generating a frequency polygon.

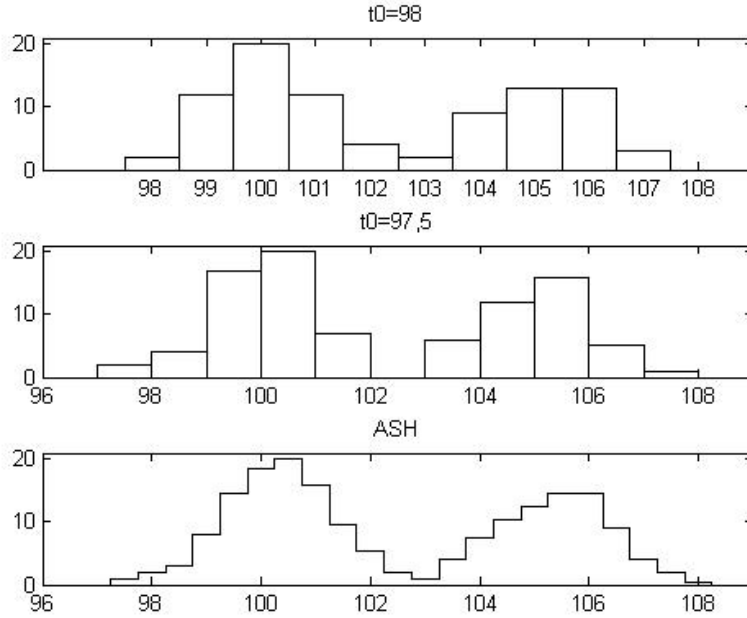


Fig. 2.3. The averaging of shifted histograms.

$$\hat{f}_{ASH} = \frac{1}{m} \sum_{i=1}^m \hat{f}_i \quad (2.25)$$

Fig. 2.3 presents the process of averaging of the shifted histograms for the data set used in Fig. 2.1, for $m=2$.

The shape of the estimate obtained by averaging the shifted histograms maintains the histogram appearance, having a similar aspect to a histogram with a finer bin width of h/m . Any of the initial histograms can be obtained by merging m consecutive fine bins from the averaged histogram, ASH. On the other hand, the value of ASH in the bin B_k is the average of the m bins in the original histograms which contain the fine bin B_k . This leads to the following form of the ASH estimate of the probability density:

$$\hat{f}(x; m) = \frac{1}{m} \sum_{i=1-m}^{m-1} \frac{(m-|i|)v_{k+i}}{nh} = \frac{1}{nh} \sum_{i=1-m}^{m-1} \left(1 - \frac{|i|}{m}\right) v_{k+i}, \quad x \in B_k. \quad (2.26)$$

The sum in equation (2.26) is a weighted average of the number of observations from the bins that fit a window centered on the bin B_k , which extends $(m-1)$ bins on both sides of bin B_k . The weights in equation (2.26) define the shape of the window as an isosceles triangle. Equation (2.26) can also be generalized in order to allow other window shapes:

$$\hat{f}(x; m) = \frac{1}{nh} \sum_{i=1-m}^{m-1} W_m(i) v_{k+i}, \quad x \in B_k. \quad (2.27)$$

In order to ensure the unitary integral of the estimate, it is necessary for the sum of the weights in the window to be 1, too. The weights can be defined using equation (2.28), where K is a continuous function – usually a probability density function.

$$w_m(i) = m \frac{K(i/m)}{\sum_{j=1-m}^{m-1} K(j/m)}, \quad i = 1-m, \dots, m-1 \quad (2.28)$$

ASH removes the effect of the choice of bins' origin, at the cost of an additional parameter, m . As long as the value of the newly introduced parameter, m , is higher than 1, its effect on the estimate of the probability density is negligible compared to the choice of the origin point.

Another important characteristic of the ASH is its asymptotic behavior: as the m parameter grows approaching infinity, the ASH estimate can be written as an estimator belonging to another important class of non-parametric estimators – kernel estimators:

$$\lim_{m \rightarrow \infty} \hat{f}(x, m) = \frac{1}{nh} \sum_{j=1}^n K\left(\frac{x - x_j}{h}\right), \quad K(t) = (1 - |t|) I_{[-1, 1]}(t), \quad (2.29)$$

$$I_{[a, b]}(t) = \begin{cases} 1, & t \in [a, b] \\ 0, & t \notin [a, b] \end{cases}.$$

2.3. Kernel density estimation

Kernel estimators were introduced by Rosenblatt (1956) and Parzen (1962). A kernel-based estimator is defined by the following equation:

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i), \quad K_h(t) = \frac{1}{h} K\left(\frac{t}{h}\right). \quad (2.30)$$

In this case, the parameter h represents the kernel bandwidth, having a similar effect to that of the bin width for the histogram based estimators. Usually the kernel used for estimation is symmetric and integrates to 1 over the entire real domain:

$$K(t) = K(-t); \quad (2.31)$$

$$\int_{-\infty}^{\infty} K(t) dt = 1. \quad (2.32)$$

In addition to the condition in equation (2.32), for a p order kernel all central moments up to order $p - 1$ must be 0, while the p order central moment must be non-zero:

$$\int_{\mathbb{R}} t^i K(t) dt = 0, \quad i = 1, 2, \dots, p-1 \quad (2.33)$$

$$\int_{\mathbb{R}} t^p K(t) dt \neq 0.$$

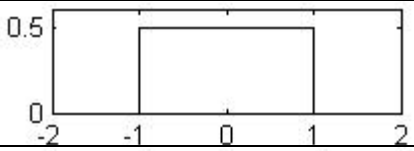
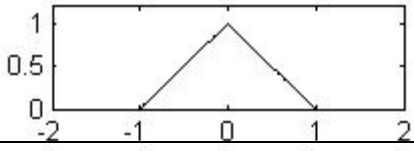
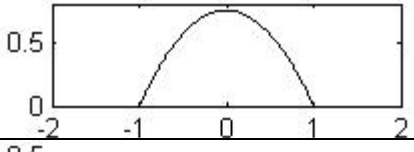
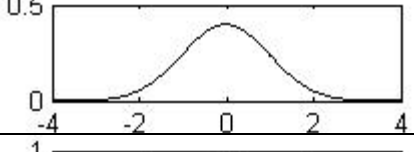
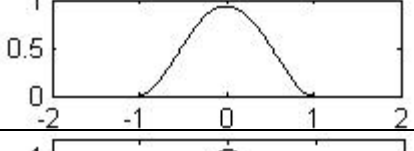
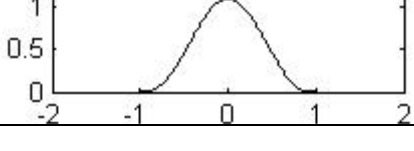
Due to the symmetry condition in equation (2.31), the order, p , must be an even number (for a symmetric kernel all odd central moments are 0). In the case of a second order kernel it is possible to choose a non-negative kernel, which might also be a probability density.

Kernels of orders higher than 2 also need to take negative values. In this case, it is also possible for the estimate to have negative values over some intervals. For this reason second order kernels are preferred in applications.

Table 2.1 presents some of the most widely used second order kernels. All the kernels presented in this table have finite support, except for the normal kernel, which has an infinite support.

Actually, two approaches can be used in practice for the kernel-based estimation, both leading to the same final result. The first approach for building a kernel estimator places a kernel at each observation point x_i and generates the estimate by summing all the kernels. The second approach places a kernel at each estimation point and calculates the value of the estimate at that point as a weighted average of the samples covered by the kernel window, using the weights provided by the kernel itself.

Table 2.1. Widely used second order kernels.

Kernel name		K(t)
Uniform	$\frac{1}{2} I_{[-1,1]}(t)$	
Triangle	$(1 - t) I_{[-1,1]}(t)$	
Epanechnikov	$\frac{3}{4}(1 - t^2) I_{[-1,1]}(t)$	
Normal	$\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}$	
Biweight (quartic)	$\frac{15}{16}(1 - t^2)^2 I_{[-1,1]}(t)$	
Triweight	$\frac{35}{32}(1 - t^2)^3 I_{[-1,1]}(t)$	

For both approaches the value of the estimate at a given point results as a weighted average over all the observations, the weights depending on the distance of the observation with respect to the estimation point. If the kernel has a finite support, only the observations contained within the kernel window are taken into account for calculating the estimate, the other observation having null weights.

2.3.1. Parameters that affect the quality of kernel based estimation

There are 2 main parameters that influence the quality of the kernel-based estimation: the kernel bandwidth and the kernel's shape. In the early years of kernel based estimation, the research in the field focused on finding the best kernel type, but the conclusion was reached that the estimate is influenced to a much greater extent by the kernel bandwidth than by the kernel type [8].

As was also the case with the histogram based estimation, the h parameter is the one that has a crucial influence on the quality of the estimate. Fig. 2.4 reveals this aspect, presenting a comparison between 3 estimates obtained using triangle kernels with different bandwidths.

In all 3 cases the observations sample used for estimation contains 10 values drawn from a normal distribution with mean 100 and standard deviation 1, and 8 values drawn from a normal distribution with mean 105 and standard deviation 1. For this data set the value $h=0.7$ produces a noisy estimate; the value $h=1.4$ produces an estimate which reveals the bimodality of the probability density, without other artifacts; the value $h=3.5$ produces an excessively smoothed estimate, hiding the bimodality of the real distribution. Due to its effect on the estimate, the kernel bandwidth, h , is also known as the smoothing parameter. A too small value for h leads to a high variance of the estimate. A too large value leads to an increase of the estimator's bias.

The choice of the kernel may also have some influence, from case to case, but usually to a much lesser extent than the smoothing parameter. The use of different kernels does not lead to identical estimates, when the same bandwidth is used for all kernels, but the use of different bandwidths may lead to comparable results. Fig. 2.5 presents – for the same observations sample as in the previous example – the estimate obtained using a triangle kernel with bandwidth of 1.4 (a) alongside two estimates obtained using the normal kernel with bandwidths of $h=0.6$ (b) and $h=1.4$ (c). The estimate obtained using the normal kernel with bandwidth of 1.4 is excessively smoothed compared to the one obtained with the triangle kernel

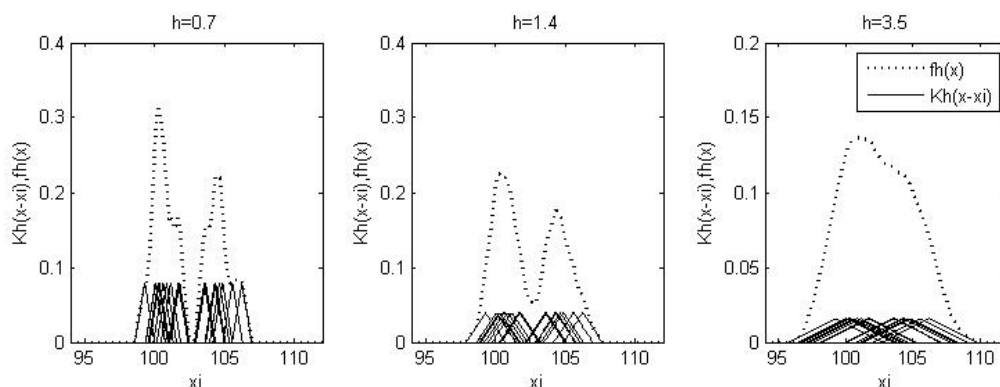


Fig. 2.4. The influence of the kernel bandwidth over the quality of the estimate.

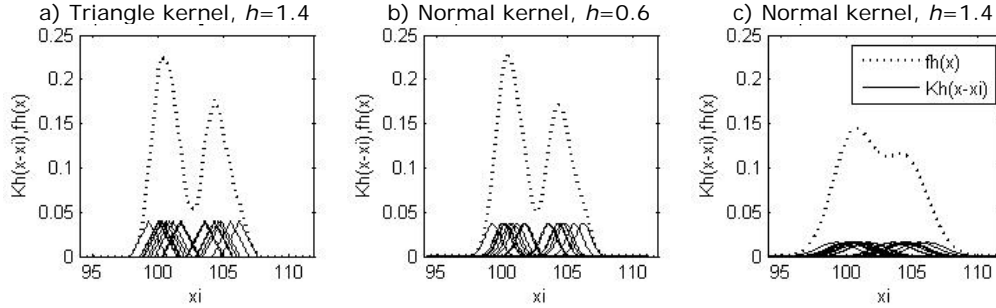


Fig. 2.5. Estimates obtained with triangle (a) and normal kernels (b, c).

of the same bandwidth. On the other hand, the estimate obtained using the normal kernel with bandwidth of 0.6 is almost identical to the one obtained using the triangle kernel with bandwidth of 1.4.

Obviously, there is no kernel that can be recommended for any possible situation. Finite support kernels are not continuously differentiable (i.e. they are useless for estimating the derivatives), while infinite support kernels (e.g. normal kernel) are relatively inefficient compared to the finite support ones.

2.3.2. Criteria for assessing the quality of estimation

In the early years of kernel density estimation, the estimates were evaluated subjectively, based on aesthetical criteria. Later, the researchers tried to develop objective methods for calculating the optimal bandwidth of the kernel. Multiple criteria were proposed for assessing the quality of the estimation. Each criterion measures a certain quantity in order to assess the quality of the estimation, resulting in different definitions of the optimal bandwidth, depending on the measured quantity.

The quality of the estimation is assessed with respect to the distance between the real probability density, f , and the estimated one, \hat{f}_h . The main problem is how to evaluate this distance. Most evaluation methods rely on distances in the L^2 space, as they allow for easier analysis compared to the distances in the L^1 space. Two of the most widely used distances are ISE (Integrated Squared Error) – defined by equation (2.34) – and MISE – defined by equation (2.35).

$$ISE(h) = \int \left[\hat{f}_h(x) - f(x) \right]^2 dx \quad (2.34)$$

$$MISE = E \left\{ \int \left[\hat{f}_h(x) - f(x) \right]^2 dx \right\} \quad (2.35)$$

The smoothing parameters that minimize these distances are denoted by \hat{h}_0 for ISE and h_0 for MISE. Both $ISE(h)$ and its minimizer, \hat{h}_0 , are random variables, depending on the observations sample for which they are calculated. Both minimizers also depend on the total number of observations in a sample, n .

The MISE can be expressed as a sum between the IV (Integrated Variance) and the ISB (Integrated Squared Bias):

$$\begin{aligned}
MISE(h) &= \int E[\hat{f}_h(x) - f(x)]^2 dx \\
&= \int \underbrace{E\{\hat{f}_h(x) - E[\hat{f}_h(x)]\}^2}_{\sigma_{\hat{f}_h}^2} dx + \int \underbrace{\{E[\hat{f}_h(x)] - f(x)\}^2}_{bias(\hat{f}_h)} dx \quad (2.36) \\
&= IV(h) + ISB(h).
\end{aligned}$$

The kernel based estimator, $\hat{f}_h(x)$, is actually obtained as an average between n independent identically distributed random variables, $K_h(x - x_j)$. Equations (2.37) and (2.38) calculate the expected value and the variance of the kernel based estimator, using the expected value and the variance of the kernel K_h :

$$\begin{aligned}
E(\hat{f}_h) &= E(K_h) = K_h(x - t) f(t) dt = K_h * f = \\
&= \int K(w) f(x - hw) dw, \quad (2.37)
\end{aligned}$$

$$\sigma_{\hat{f}_h}^2 = \frac{1}{n} \sigma_{K_h}^2 = \frac{1}{n} \{E(K_h^2) - [E(K_h)]^2\}. \quad (2.38)$$

Similarly to equation (2.37), the expected value of K_h^2 can be expressed using the substitution $w = \frac{x-t}{h}$:

$$\begin{aligned}
E(K_h^2) &= \int K_h^2(x-t) f(t) dt = \int \frac{1}{h^2} K^2\left(\frac{x-t}{h}\right) f(t) dt \\
&= \frac{1}{h} \int K^2(w) f(x-hw) dw. \quad (2.39)
\end{aligned}$$

The 2 terms which sum to give $MISE(h)$ in equation (2.36) can be further expressed, considering the equations (2.37), (2.38) and the notation $R(g) = \int g^2(x) dx$. The functional $R(g)$ is also known as the roughness of function g . The integrated variance can be obtained by integrating the equation (2.38):

$$\begin{aligned}
IV(h) &= \frac{1}{n} \int \{E(K_h^2) - [E(K_h)]^2\} dx \\
&= \frac{1}{nh} \int \left[\int K^2(w) f(x-hw) dw \right] dx - \frac{1}{n} (K_h * f)^2(x) dx. \quad (2.40)
\end{aligned}$$

By reversing the integration order in equation (2.40) based on Fubini's theorem [9] and given that f is a probability density (i.e. its integral over the entire real domain is 1), the $IV(h)$ becomes:

$$\begin{aligned}
IV(h) &= \frac{1}{nh} \int \left[K^2(w) \underbrace{\int f(x-hw) dx}_1 \right] dw - \frac{1}{n} \int (K_h * f)^2(x) dx \\
&= \frac{R(K)}{nh} - \frac{1}{n} \int (K_h * f)^2(x) dx.
\end{aligned} \tag{2.41}$$

Considering K , as a symmetrical k -order kernel, by substituting the convolution as in the last expression in equation (2.37) and by using the Taylor series representation of $f(x-hw)$, $E(\hat{f}_h)$ can be written as:

$$\begin{aligned}
K_h * f &= \int K(w) f(x-hw) dw = \int K(w) \left[f(x) - hwf'(x) + \frac{1}{2} h^2 w^2 f''(x) + \dots \right] dw \\
&= f(x) \underbrace{\int K(w) dw}_1 - hf'(x) \underbrace{\int wK(w) dw}_0 + \dots + \frac{h^k}{k!} f^{(k)}(x) \int w^k K(w) dw + \dots \\
&= f(x) + \frac{h^k}{k!} f^{(k)}(x) \int w^k K(w) dw + \dots
\end{aligned} \tag{2.42}$$

The integrated variance becomes:

$$IV(h) = \frac{R(K)}{nh} - \frac{R(f)}{n} + O\left(\frac{h^k}{n}\right). \tag{2.43}$$

Using the Taylor series decomposition from equation (2.42), the bias of the estimator \hat{f}_h can be written as:

$$bias(\hat{f}_h) = E[\hat{f}_h(x)] - f(x) = \frac{h^k}{k!} \mu_k(K) f^{(k)}(x) + O(h^{k+2}), \quad \mu_k(K) = \int w^k K(w) dw \tag{2.44}$$

By integrating the square of the bias from equation (2.44), the ISB(h) can be written as:

$$ISB(h) = \frac{h^{2k}}{(k!)^2} \mu_k^2(K) R(f^{(k)}) + O(h^{2k+2}). \tag{2.45}$$

When $n \rightarrow \infty$, it is necessary and sufficient that $h_0 \rightarrow 0$ and $nh_0 \rightarrow \infty$, in order to ensure that $MISE(h_0) \rightarrow 0$. The AMISE is defined by keeping only the dominant terms in the formulas of the IV(h) (2.43) and ISB(h) (2.45):

$$AMISE(h) = \frac{R(K)}{nh} + \frac{h^{2k}}{(k!)^2} \mu_k^2(K) R(f^{(k)}). \tag{2.46}$$

When $h \rightarrow 0$ and $nh \rightarrow \infty$, the MISE consists of a dominant term (AMISE) and a sum of other terms which are negligible compared to the AMISE:

$$MISE(h) = AMISE(h) + o(AMISE(h)). \quad (2.47)$$

The quality of approximation of the MISE by its asymptotic value AMISE was intensively studied by statisticians. Though in many situations the approximation proved to be satisfactory with sample sets of reasonable size, situations were reported when the approximation became satisfactory only when the size of the sample set grew to millions of observations. The conclusion was reached that the main reason for the poor performance of the approximation of MISE by AMISE in such situations was the poor quality of the approximation of ISB by its dominant term. Nevertheless, the inclusion of additional terms in the approximation of ISB does not bring significant improvements to the quality of the approximation. On the other hand, the approximation of the IV by its dominant term proved to be generally effective [10].

Equation (2.46) emphasizes the influence of the smoothing parameter, h , on the estimator. It can be noticed that the term coming from the integrated variance, IV, is inversely proportional to h , while the term coming from the ISB is proportional with h^{2k} . This observation indicates that a too small value of h decreases the bias at the cost of increasing the variance, while a too large value of h decreases the variance while increasing the bias. The smoothing parameter which minimizes the AMISE is denoted h_{∞} and can be calculated by solving the equation:

$$\begin{aligned} \frac{d}{dh} AMISE(h) = 0 &\Leftrightarrow \\ -\frac{R(K)}{nh^2} + h^{2k+1} \left[\frac{\mu_K^2}{k!} \right]^2 R(f^{(k)}) &= 0 \Leftrightarrow \\ \frac{1}{h^2} \left\{ 2kh^{2k+1} \left[\frac{\mu_K^2}{k!} \right]^2 R(f^{(k)}) - \frac{R(K)}{n} \right\} &= 0. \end{aligned} \quad (2.48)$$

The solution of equation (2.48) is:

$$h_{\infty} = \left[\frac{R(K)(k!)^2}{2k\mu_K^2(K)R(f^{(k)})} \right]^{\frac{1}{2k+1}} n^{-\frac{1}{2k+1}} \quad (2.49)$$

It can be noticed that the growth of n in equation (2.49) causes h_{∞} to decrease at a rate of $n^{-\frac{1}{2k+1}}$. The value h_{∞} of the smoothing parameter ensures an optimal compromise between bias and variance and AMISE becomes in this case:

$$\begin{aligned}
 AMISE(h_{\infty}) &= \left(\frac{2k+1}{2k}\right) \left[\frac{2k\mu_k^2(K)R(K)^{2k}R(f^{(k)})}{(k!)^2} \right]^{\frac{1}{2k+1}} n^{-\frac{2k}{2k+1}} \\
 &= F_1(K)F_2(f)n^{-\frac{2k}{2k+1}}
 \end{aligned} \tag{2.50}$$

2.3.3. The optimal kernel

In equation (2.50), it can be noticed that the AMISE converges at a rate of $n^{-\frac{2k}{2k+1}}$. $AMISE(h_{\infty})$ contains a term that depends on the kernel K and a term that depends on the unknown probability density, f . The optimal kernel is the kernel which minimizes the term $F_1(K)$. Epanechnikov started to investigate the problem of finding the optimal kernel for second order kernels (the most widely used kernels) in 1969 [8]. For second order kernels $\mu_2^2(K) = \sigma_K^2$ and the following equations describe the $AMISE(h)$, h_{∞} and $AMISE(h_{\infty})$:

$$AMISE(h) = \frac{R(K)}{nh} + \frac{1}{4}\sigma_K^4 h^4 R(f''), \tag{2.51}$$

$$h_{\infty} = \left[\frac{R(K)}{\sigma_K^4 R(f'')} \right]^{1/5} n^{-1/5}, \tag{2.52}$$

$$AMISE(h_{\infty}) = \frac{5}{4} [\sigma_K R(K)]^{4/5} R(f'')^{1/5} n^{-4/5}. \tag{2.53}$$

The convergence rates are $n^{-1/5}$ for h_{∞} and $n^{-4/5}$ for $AMISE(h_{\infty})$, respectively. The optimal kernel is the kernel which minimizes the term $[\sigma_K R(K)]^{4/5}$. The solution found by Epanechnikov is the kernel which received his name:

$$K_2^*(t) = \frac{3}{4}(1-t^2)I_{[-1,1]}(t). \tag{2.54}$$

The efficiency of a kernel is defined as [10]:

$$eff(K) = \left[\frac{F_1(K^*)}{F_1(K)} \right]^{\frac{2k+1}{2k}}. \tag{2.55}$$

The meaning of the kernel efficiency as defined in equation (2.55) is that the MISE obtained with the optimal kernel K^* using n observations can be achieved

with kernel K using $n \cdot \text{eff}(K)$ observations. All the second order kernels presented in Table 2.1 have efficiencies close to 1, with the uniform and normal kernels as the less efficient ones, requiring 1.075 and 1.05 times, respectively, more data than the optimal Epanechnikov kernel. It was determined that even the less efficient second order kernels have efficiency values below 2. These data indicate that the choice of the optimal kernel is far less important for practical applications compared to the choice of the kernel's bandwidth. The kernel choice is only important when the derivatives of the probability density function need to be estimated. In this case it is necessary for the chosen kernel to be differentiable enough in order to allow consistent estimates of the derivatives of the probability density.

2.3.4. The optimal bandwidth

Any of \hat{h}_0 , h_0 and h_{∞} can be considered as optimal bandwidth, given that it minimizes a L^2 distance between the estimator and the real probability density. Although in practice none of the three bandwidths can be used, as they all depend on the unknown probability density, f . Therefore, for practical applications an estimate of the optimal kernel bandwidth, \hat{h} , needs to be calculated. The problem is which of the three bandwidths should be estimated from an observations sample.

AMISE reveals the behavior of the estimated probability density at too small or too high values and some optimal bandwidth selection methods use this criterion considering that AMISE is a good approximation of MISE. Nevertheless, in some situations, the aforementioned assumption only holds true when the sizes of the sample sets exceed millions of observations.

Between \hat{h}_0 and h_0 the second one is preferred, given that \hat{h}_0 is a random variable and is more difficult to estimate. It was also demonstrated that the optimal bandwidth converges much slower towards \hat{h}_0 (convergence rate of order $n^{-1/10}$) than towards h_0 (convergence rate of order $n^{-1/2}$). Despite these arguments, some researchers suggested that none of the L^2 distances are adequate, as they do not reflect the human subjects' perception about the quality of estimation. However, the criteria based on L^2 distances and their minimizers are still used until new objective methods, able to provide results closer to the human subjective perception, will be developed.

2.3.5. Bandwidth selection methods

Bandwidth selection is of crucial importance for kernel-based density estimation, and is heavily influenced by the purpose of the estimation (i.e. the optimal bandwidth selection method is application dependent). The most important classes of bandwidth selection methods are presented next: quick and dirty methods, cross-validation methods, plug-in methods and Fourier based methods. The paragraph ends with some considerations on the criteria that should be fulfilled by the optimal selection method.

The quick and dirty methods rely on AMISE and its minimizer h_{∞} . The unknown quantity in the expression of h_{∞} (2.52) is the roughness of the second derivative of the unknown probability density, $R(f'')$. The empirical method

evaluates the unknown quantity by replacing the unknown probability density function, f , with a reference probability density function. The reference function is rescaled in order to have the same variance as the observations sample. When the normal kernel is used for estimation by replacing the unknown probability density with the normal reference probability density, the estimated bandwidth is:

$$\hat{h}_{emp} = \left(\frac{4}{3}\right)^{1/5} \hat{\sigma} n^{-1/5} = 1.06 \hat{\sigma} n^{-1/5}. \quad (2.56)$$

In equation (2.56) $\hat{\sigma}$ denotes the standard deviation of the observations sample. Considering the constant in equation (2.56) is close to 1, Scott proposed a simplified version of the expression of \hat{h}_{emp} by approximating the constant with 1, in order to reduce the complexity of the calculations. Another version of the empirical method improves the robustness, by allowing the use of the inter-quartile range, IQ, as a measure of the spreading of the data instead of the standard deviation:

$$\hat{h}_{emp} = 1.06 \min\left(\hat{\sigma}, \frac{IQ}{1.34}\right) n^{-1/5}. \quad (2.57)$$

The oversmoothing method is another method from the “quick and dirty” category, which imposes a lower limit on the unknown quantity $R(f'')$, leading to an upper limitation of the bandwidth:

$$\hat{h}_{os} = 3 \left[\frac{R(K)}{35\sigma_K^4} \right]^{1/5} \hat{\sigma} n^{-1/5}. \quad (2.58)$$

The quick and dirty methods have relatively good results for unimodal densities, but for multimodal densities they tend to have an oversmoothing effect and hide the real nature of the data.

Cross-validation methods are basically extensions of the methods used in parametric modelling [11]. They rely on a statistical procedure which involves splitting the observations sample into multiple subsets. The initial analysis is realized on a single subset, while the other subsets are used for validating the results of the initial analysis. The initial subset is referred to as training set, while the remaining subsets are referred to as validation sets or test sets.

The most widely used cross validation methods are the k -fold cross-validation methods, which split the initial sample into k subsets of which only one subset is kept for validation, the other $k-1$ subsets being used as training data. The cross-validation process is repeated k times, each subset being used only once for validation. The final estimation is obtained either by averaging the results of the k cross-validations, or by another combination of the k results. A particular case of this category of methods is the leave-one-out method, where the number of subsets is equal to the number of observations (each subset has a single observation).

The cross-validation of the pseudo-probability is a method which requires the bandwidth to be chosen such as to maximize the pseudo-probability function

$\prod_{i=1}^n \hat{f}_h(x_i)$. In this case the problem has a trivial solution for $h = 0$. In the

expression of the pseudo-probability \hat{f}_h is substituted by the leave-one-out version, $\hat{f}_{h,i}$ (the estimate built without the x_i observation):

$$\hat{f}_{h,i} = \frac{1}{n-1} \sum_{\substack{j=1 \\ j \neq i}}^n K_h(x - x_j). \quad (2.59)$$

This method minimizes the Kullback-Leibler distance between \hat{f}_h and f . It also has satisfactory properties in the L^1 space, but fails to offer consistent estimates for heavy tail densities [10].

UCV (Unbiased Cross-Validation), also known as LSCV (Least Squares Cross-Validation), aims at estimating the minimizer of ISE, \hat{h}_0 . Equation (2.34) can be further developed as:

$$ISE(h) = \int \hat{f}_h^2(x) dx - 2 \int \hat{f}_h(x) f(x) dx + \int f^2(x) dx, \quad (2.60)$$

where the last term is independent of the bandwidth, h (and therefore does not require minimization) and the first term is known. Therefore, the only term in equation (2.60) that needs to be estimated is the second one. By substituting in the middle term \hat{f}_h with its estimate obtained by the leave-one-out method, the quantity to minimize becomes:

$$UCV(h) = R(\hat{f}_h) - \frac{2}{n} \sum_{i=1}^n \hat{f}_{h,i}(x_i). \quad (2.61)$$

$UCV(h)$ is an unbiased estimator of the quantity $ISE(h) - R(f)$ because $E[UCV(h)] = E[ISE(h)] - R(f) = MISE(h) - R(f)$.

The estimator of \hat{h}_0 is \hat{h}_{UCV} , which minimizes $UCV(h)$. The method is widely used due to its intuitive motivation. Despite the fact that the convergence rate of \hat{h}_{UCV} towards \hat{h}_0 is very slow (order $n^{-1/10}$), it is the best convergence rate for the estimation of \hat{h}_0 . The reduced convergence rate makes the method very sensitive to changes in the sample. Another disadvantage is represented by the fact that $UCV(h)$ often has multiple minima, but this problem can be reasonably alleviated by choosing the highest value of h for which a local minimum is obtained. An even more problematic situation is when $UCV(h)$ has no minimum [10].

Similar to the “quick and dirty” methods, BCV (Biased Cross-Validation) aims at minimizing the AMISE by replacing the unknown quantity $R(f'')$ with an estimate. In this case the estimate is based not on a reference density, but on the roughness of the second derivative of the estimate of the probability density:

$$\begin{aligned}
R(\hat{f}_h^n) &= \int \left(\frac{1}{n} \sum_{j=1}^n K_h^n(x - x_j) \right)^2 dx = \frac{1}{n^2} \int \sum_{i=1}^n \sum_{j=1}^n K_h^n(x - x_i) K_h^n(x - x_j) dx = \\
&= \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \int K_h^n(x - x_i) K_h^n(x - x_j) dx = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n K_h^n * K_h^n(x_i - x_j) \quad (2.62) \\
&= \frac{1}{n} K_h^n * K_h^n(0) + \frac{1}{n^2} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n K_h^n * K_h^n(x_i - x_j).
\end{aligned}$$

Given that

$$K_h^n(t) = \left[\frac{1}{h} K\left(\frac{t}{h}\right) \right]^n = \frac{1}{h^3} K^n\left(\frac{t}{h}\right) \quad \text{and} \quad (2.63)$$

$$\begin{aligned}
K_h^n * K_h^n(t) &= \int K_h^n(x) K_h^n(t - x) dx = \frac{1}{h^6} \int K^n\left(\frac{x}{h}\right) K^n\left(\frac{t-x}{h}\right) dx = \\
&= \frac{1}{h^5} \int K^n\left(\frac{x}{h}\right) K^n\left(\frac{t-x}{h}\right) d\frac{x}{h} = \frac{1}{h^5} K^n * K^n\left(\frac{t}{h}\right), \quad (2.64)
\end{aligned}$$

equation (2.62) can be written as:

$$\begin{aligned}
R(\hat{f}_h^n) &= \frac{1}{nh^5} K^n * K^n(0) + \frac{1}{n^2 h^5} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n K^n * K^n\left(\frac{x_i - x_j}{h}\right) = \\
&= \frac{1}{nh^5} R(K^n) + \frac{1}{n^2 h^5} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n K^n * K^n\left(\frac{x_i - x_j}{h}\right). \quad (2.65)
\end{aligned}$$

The authors of this method (Scott and Terrell) have proven [12] that $R(\hat{f}_h^n)$ is a biased estimator of $R(f^n)$, because $E[R(\hat{f}_h^n)] = R(f^n) + \frac{R(K^n)}{nh^5} + O(h^2)$. For this reason, the two authors of the method have proposed to estimate $R(f^n)$ by:

$$\hat{R}(f^n) = R(\hat{f}_h^n) - \frac{R(K^n)}{nh^5}. \quad (2.66)$$

The score function obtained based on this estimate is:

$$\begin{aligned}
BCV(h) &= \frac{R(K)}{nh} + \frac{h^4 \sigma_K^4}{4n^2} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n K'' * K''(x_i - x_j) = \\
&= \frac{R(K)}{nh} + \frac{\sigma_K^4}{4n^2 h} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n K'' * K''\left(\frac{x_i - x_j}{h}\right)
\end{aligned} \tag{2.67}$$

The bandwidth \hat{h}_{BCV} that minimizes the score function in equation (2.67) has the same convergence rate towards h_0 or \hat{h}_0 as \hat{h}_{UCV} , but the constant is usually much smaller. The problem of multiple minima may still appear, but less often than in the case of UCV. As opposed to the UCV case, for BCV the multiple minima problem is solved by choosing the smallest value of h for which a local minimum occurs. The problem of no minima also exists for BCV, even with one of the most widely used kernels, the normal kernel.

SCV (Smoothed Cross-Validation) relies on the MISE decomposition in equation (2.36) as the sum between the IV and ISB. Since the IV can be satisfactorily estimated by its dominant term in the Taylor decomposition, $\frac{R(K)}{nh}$, the problem remains to find a good estimate for ISB, too. The initial method estimates the unknown probability density in the expression of the ISB with \hat{f}_g , which is an estimator built by using a kernel L (possibly different from K) that has the bandwidth g (also possibly different from h). Using this estimation, the ISB becomes:

$$ISB(h) = \int (K_h * f - f)^2(x) dx = \frac{1}{n^2} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n (K_h * K_h - 2K_h + \delta) * L_g * L_g(x_i - x_j). \tag{2.68}$$

In equation (2.68), δ represents the Dirac function. By removing the diagonal terms (for which $i=j$) from the summation, similar to the case of BCV, and considering that $n \approx (n-1)$, the score function can be written as:

$$SCV(h) = \frac{R(K)}{nh} + \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n (K_h * K_h - 2K_h + \delta) * L_g * L_g(x_i - x_j). \tag{2.69}$$

Given that the minimized quantity for UCV can be written in a similar form, $SCV(h)$ can be considered a smoothed version of $UCV(h)$ [10]. The bandwidth \hat{h}_{SCV} which minimizes $SCV(h)$ depends on the bandwidth, g , of the second kernel. The convergence rate of \hat{h}_{SCV} towards h_0 is of the order $n^{-1/2}$ (optimal convergence rate towards h_0). With the initial version of SCV, the optimal convergence rate can be achieved using high order kernels (at least of the order 6). It was demonstrated that the optimal convergence rate can also be achieved with 2nd order kernels if the diagonal terms in equation (2.68) are not removed and the

bandwidth of the second kernel depends on the bandwidth of the first one. The advantage of this improvement is crucial for practical applications, where high order kernels offer poor results. The main disadvantage is the relatively high asymptotic integrated variance [10].

Plug-in methods aim at minimizing the AMISE, estimating the unknown quantity $R(f^{(p)})$ in a sequence of iterative steps. In the first step $R(f^{(p)})$ is estimated using an initial bandwidth, h_1 . The $\hat{R}(f^{(p)})$ obtained in the first step is then used for calculating a new bandwidth value. The above described process repeats iteratively until bandwidth convergence is achieved. Methods in this category have convergence rates of order $n^{-4/13} - n^{-5/14}$ with second order kernels. The best convergence rate in this category, using second order kernels, is achieved when an additional term is taken into account in the expression of the asymptotic ISB, leading to the following expression of $AMISE_2(h)$:

$$AMISE_2(h) = \frac{R(K)}{nh} + \frac{h^4 \mu_2^2(K) R(f'')}{4} - \frac{h^6 \mu_2(K) \mu_4(K) R(f^{(3)})}{24}. \quad (2.70)$$

Finding the bandwidth which minimizes this expression of the AMISE involves complex calculations, including solving 7th degree equations. Therefore, the authors of the method using the above expression of AMISE have proposed an asymptotically equivalent solution for the bandwidth which minimizes $AMISE_2(h)$ in equation (2.70):

$$\hat{h} = \left(\frac{\hat{J}_1}{n} \right)^{1/5} + \hat{J}_2 \left(\frac{\hat{J}_1}{n} \right)^{1/5}, \quad \hat{J}_1 = \frac{R(K)}{\mu_2^2(K) \hat{R}(f'')}, \quad \hat{J}_2 = \frac{\mu_4(K) \hat{R}(f^{(3)})}{20 \mu_2(K) \hat{R}(f'')}. \quad (2.71)$$

Convergence rates of $n^{-1/2}$ can only be achieved with this type of methods by estimating the roughness of the second and third derivative of the probability density function using high order kernels [10].

Good simulation results were also reported with Fourier transform based methods, which achieve convergence rates of order $n^{-1/2}$ towards h_0 , at the cost of the complex calculation of Fourier transforms, required in order to perform a frequency analysis of the characteristic function of the sample [10].

Methods based on bootstrap were explored more recently and rely on the substitution of the MSE with a bootstrapped version, $MSE_{n,s}^*$. The resampling can be performed either from a subsample of the data set or from a pilot density. The bandwidth choice in this case consists in estimating the s parameter in the equation:

$$h = n^{-1/5} s. \quad (2.72)$$

Relying on equation (2.72), the bootstrap method in [13] proposes the bandwidth:

$$h_n = n^{-1/5} \arg \min_s MSE_{n,s}^*. \quad (2.73)$$

The *optimal method* should fulfill simultaneously multiple criteria: reduced asymptotic variance, convergence rate of order $n^{-1/2}$ and good results in practical applications with relatively small size samples using only second order kernels. Many bandwidth selection methods are able to achieve convergence rates of order $n^{-1/2}$, but the only one which is able to achieve it using only second order kernels is the SCV. Though, SCV has the disadvantage of the high asymptotic variance. Methods that have both optimal asymptotic variance and convergence rate using only second order kernels were also proposed, but with unsatisfactory results in practical applications. Therefore, the conclusion is that there is no bandwidth selection method that can be recommended as optimal for any application, but the choice has to be made taking into account the specificity of the application at hand.

2.3.6. Kernel density estimation for multivariate data

The estimation in the case of multivariate (multidimensional) data requires the use of multidimensional kernels. A multidimensional kernel can be obtained starting from a one-dimensional kernel that is used in each dimension, but with different bandwidths. The estimate of the probability density in this case is:

$$\hat{f}(x) = \frac{1}{nh_1 \dots h_d} \sum_{i=1}^n \left[\prod_{j=1}^d K \left(\frac{x_i - x_{ij}}{h_j} \right) \right], \quad x = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{id} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{nd} \end{pmatrix}. \quad (2.74)$$

The bias of the estimator in equation (2.74) is:

$$E\hat{f}(x) - f(x) = \frac{1}{2} \sigma_K^2 \sum_{i=1}^d h_i^2 f_{ii}(x) + O(h^4), \quad f_{ij} = \frac{d^2 f}{didj}, \quad (2.75)$$

while the asymptotic values of the ISB and IV are:

$$AISB = \frac{1}{4} \sigma_K^4 \left[\sum_{i=1}^d h_i^4 R(f_{ii}) + \sum_{i \neq j} h_i^2 h_j^2 \int_{\mathbb{R}^d} f_{ij} f_{jj} dx \right], \quad (2.76)$$

$$AIV = \frac{R(K)^d}{nh_1 \dots h_d} - \frac{R(f)}{n} + O\left(\frac{h}{n}\right).$$

Finding the optimal bandwidths in d dimensions requires solving a system of d non-linear equations with d unknowns. No general formula is available for calculating the individual bandwidths. Solutions are available for certain particular cases (e.g. $d \leq 2$, one bandwidth for all dimensions, certain types of kernels). For the normal kernel the bandwidth formula is:

$$h_i^* = \left(\frac{4}{d+2} \right)^{1/(d+4)} \sigma_i n^{-1/(d+4)}. \quad (2.77)$$

A simplified version of the formula in equation (2.77) was proposed by Scott [8], who noticed that the quantity $\left(\frac{4}{d+2}\right)^{1/(d+4)}$ always has values very close to 1, and therefore can be approximated with 1:

$$\hat{h}_i = \hat{\sigma}_i n^{-1/(d+4)}. \quad (2.78)$$

The generalized expression of the multidimensional kernel estimator is:

$$\hat{f}(x) = \frac{1}{n|H|} \sum_{i=1}^n K[H^{-1}(x - x_i)]. \quad (2.79)$$

In equation (2.79), H represents a $d \times d$ transform matrix. The multidimensional product kernel is a particular case, for which the transform matrix is a diagonal matrix, containing the bandwidths of the d kernels.

The bandwidth selection methods presented for one-dimensional kernels can also be extended for the multidimensional case. Though the convergence rates for the multidimensional methods are slower than for their one-dimensional counterparts, it is very important to use different bandwidths for each dimension, as it is very unlikely for a given bandwidth to be optimal for all dimensions even with normalized data.

2.3.7. Adaptive smoothing

An adaptive estimator can be obtained using an estimator with the d -dimensional kernel K , that has the same bandwidth in all dimensions. The bandwidth is allowed to vary both with respect to the point of estimation and with the particularities of the unknown probability density, f .

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i), \quad h = h(x, x_i, f). \quad (2.80)$$

The adaptive bandwidth function in equation (2.80) is a smooth function. For finite size samples, two main categories of adaptive estimation methods can be considered. The first category uses a fixed bandwidth for all the n points of the sample, but the bandwidth is allowed to vary at each estimation point x :

$$\hat{f}_1(x) = \frac{1}{n} \sum_{i=1}^n K_{h_x}(x - x_i), \quad h_x = h(x, x, f). \quad (2.81)$$

The second category of adaptive estimation methods uses different bandwidths at each point, x_i , of the sample, but – once chosen – the bandwidths are used for estimating the probability density at all the estimation points x :

$$\hat{f}_2(x) = \frac{1}{n} \sum_{i=1}^n K_{h_i}(x - x_i), \quad h_i = h(x_i, x_i, f). \quad (2.82)$$

The k -nn estimators are amongst the most widely used estimators belonging to the first category. For the k -nn estimators, the adaptive bandwidth h_x is the

distance from the estimation point to the nearest k -th point of the sample and can be written as [8]:

$$h_x = d_k(x, \{x_i\}) \approx \left[\frac{k}{nV_d f(x)} \right]^{1/d}, \quad (2.83)$$

where V_d represents the volume of a d -dimensional sphere with unit radius. The k -nn estimators provide the best asymptotic estimation, but their main disadvantage is that they are not actually probability density functions, as they integrate to infinity. Another drawback is represented by the infinite dimensionality of the bandwidth function.

For the second category of adaptive estimators, the bandwidth function below was proposed [8]:

$$h_i = hd_k(x_i, \{x_1, x_2, \dots, x_n\}) \approx h \left[\frac{k}{nV_d f(x)} \right]^{1/d}. \quad (2.84)$$

The estimators from this category are probability density functions if non-negative kernels are used and are also easier to express in practical applications, given the finite dimensionality of the bandwidth function (n dimensions).

Due to the smoothness of the bandwidth function, h , its values in a given neighborhood are very close to each other, and therefore the computational complexity can be reduced by using the same bandwidth value within the neighborhood. The choice between the estimators in equations (2.81) and (2.82) depends on the application at hand and on the application-specific difficulties related to the specifications of the adaptive function. While for one-dimensional data the use of adaptive bandwidth in practical application does not bring outstanding performance, for multidimensional data significant improvements of the quality of estimation can be obtained based on the adaptive bandwidth.

2.4. Conclusions

Parametric methods are completely determined up to the parameter/vector of parameters level, being easy to interpret and offering very good estimates under the right initial assumptions. On the other hand, when the initial assumptions are inaccurate, the results of parametric methods may become inconsistent, providing a wrong image of the structure in the data.

Non-parametric methods have the advantage of flexibility, requiring no initial restriction on the model, but may be difficult to interpret and under some circumstances they may also provide inaccurate estimation results. A compromise between the two main categories of estimation methods is offered by the so-called semi-parametric methods, which have two terms – a parametric one and a non-parametric one – which allows them to combine the advantages of the basic categories of estimation methods [14].

Kernel based density estimation is one of the most powerful non-parametric estimation techniques, in this case the quality of the estimation being influenced mainly by the kernel bandwidth and less by the choice of kernel. The problem of optimal kernel bandwidth selection has only application-dependent solutions, no generally applicable solution being available to date. Second order kernels are the most widely used, given that they are always positive, and they can also be

probability density functions. The Epanechnikov kernel is the most efficient finite support second order kernel, while the normal kernel is the most popular infinite support kernel.

Given that the kernel also acts as a smoothing operator, the non-parametric kernel based techniques have a wide range of applications beyond the probability density estimation: robust estimation, finding the local maxima of a function, non-parametric regression [8], mean-shift based analysis methods [15], obtaining of robust objective functions for use in random sampling algorithms etc.

The kernel based estimates provide another wide range of applications: the segmentation threshold selection, automatic threshold selection for M estimators in robust regression [16], determining the variance of the median in a sample [17] etc.

Despite the fact that most estimation methods were intensively studied with the purpose of global optimization of the estimate, the local optimization of the estimate also has a great importance, especially for the image processing domain. Some methods that focus on the discrete polynomial local optimization are presented in [18].

Though the non-parametric statistics were initially developed by mathematicians more than half a century ago and they have been intensively studied since, their practical applications – especially for multidimensional data – were severely limited for multiple decades by the reduced amount of computational resources available on the processing systems. In the last 15 years the practical importance of the non-parametric methods grew due to the significant improvement of the available computational power, though multidimensionality poses multiple problems that still await solutions.

3. VIDEO TRACKING

Video tracking is a complex process consisting of locating and tracking one or more moving objects in a video sequence. In most applications the tracking has to be realized in real time, using a processing system and a video sequence acquired by a video camera, as shown in Fig. 3.1. An algorithm is used to analyze the video sequence, in order to identify the positions of the moving objects of interest (targets) in each frame. The tracking algorithm estimates the motion parameters of each target. The data obtained using the tracking algorithm can further be used by the processing system for deciding some actions if necessary.

Real-time objects tracking has multiple applications in various fields like video surveillance [19], [20], perceptual user interfaces [21], [22], augmented reality [23], [24], object-oriented video compression [25], [26], robotics [27], [28], automotive [29], [30], medical applications [31], [32] etc.

Video tracking algorithms need to handle a large variety of dynamic events that may occur simultaneously in the scene [33]. The main problems that have to be handled during the tracking coordination are:

- tracking initialization,
- update,
- finalizing the tracking.

In real life scenes, various uncontrollable factors make the task of finding robust solutions for the above mentioned problems very difficult:

- temporal variations in the 2D shapes of the objects due to perspective transformations and/or due to deformable objects tracking
- occlusion or other types of interactions between the objects present in the scene
- objects appearing in / disappearing from the scene
- objects splitting and merging
- motion blur
- the presence of a large number of objects
- clutter etc.

The main problem in video tracking is to find a correspondence between the positions of the tracked object in consecutive frames. The difficulty of this problem increases significantly if the motion speed of the objects is high, relative to the frame rate. A motion model can be used to describe the changes of the object's appearance for the various types of movement possible. For 2D objects the motion model is an affine transformation of the initial image of the object. For rigid 3D

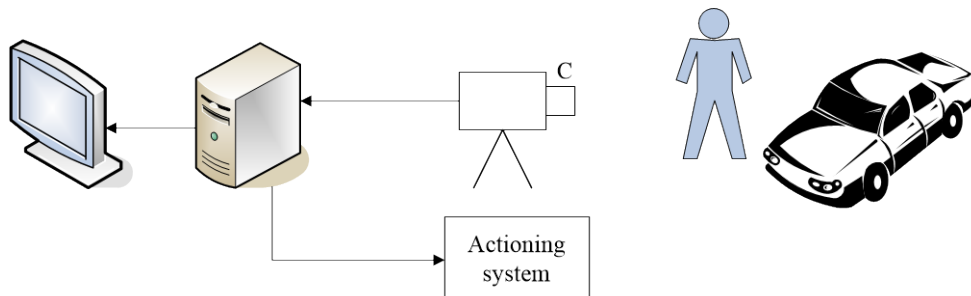


Fig. 3.1. Block diagram of a video tracking system.

objects, the motion model defines the object's aspect with respect to its position and orientation in the 3D space. In the case of deformable objects, a mesh can be used to cover the surface of the objects and the motion of the mesh's nodes describes the motion of the object. In video compression applications, the motion model is defined by the motion vectors that describe the translation of video objects based on key frames.

In video tracking literature, it is often considered [34] that a video tracking system has two main components:

- Target representation and localization and
- Filtering and data association

Target representation and localization is usually a bottom-up approach, which deals with the appearance changes of the tracked object [34]. Usually the algorithms in this category are not computationally intensive. Kernel based (mean shift) tracking [34], [35], blob tracking [36], and registration [37] are amongst the most popular algorithms in this category.

Filtering and data association is mainly a top-down process which involves using *a priori* data about the tracked object and/or the scene. The process handles the objects' dynamics and the evaluation of different hypothesis [34]. Algorithms falling into this category are usually much more computationally intensive than those for target representation and localization. The most widely known algorithms belonging to this category are the Kalman filter [38], applicable to linear Gaussian processes, and the particle filter [39], useful with non-linear and non-Gaussian distributions. A special attention to this category of algorithms is given in the book [40].

The two components can be mixed within a video tracking system, and their contributions vary from application to application. The effectiveness and robustness of the tracking process are influenced by the proper choice of the weights of the two components. In some applications (e.g. face or hands tracking) a good representation of the target is crucial, while in other applications (e.g. video surveillance) the dynamics of the subject or the camera movement are more important than the target representation [34].

The methods belonging to the first category are also known in literature as deterministic methods, having the Mean-shift algorithm as main exponent. The methods belonging to the second category are also known as stochastic methods and their main exponent is the particle filter. Recently, algorithms that combine the advantages of the two categories were developed under the name of Mean Shift (Embedded) Particle Filter [41], [21].

3.1. Target representation and localization

Recently, kernel based probability density estimators received more and more attention from the researchers and are considered to be amongst the most popular estimators of the probability density function (pdf) [42]. The mean shift algorithm used in video tracking also relies on kernel based density estimation. General aspects of kernel based probability density estimation were presented in detail in chapter 2.3. The next two sub-sections present the particularities of kernel estimators when the goal is to estimate not the probability density function, but its gradient. Then the principle of the mean shift algorithm for finding the maxima of the probability density function is presented. The particularities of the mean shift algorithm for video tracking are presented in sub-section 3.1.4.

3.1.1. Probability density estimation using kernel based operators

A d -dimensional kernel based estimator is defined as:

$$\hat{f}_H(x) = \frac{1}{n} \sum_{i=1}^n K_H(x - x_i),$$

$$K_H(t) = |H|^{-1/2} K(H^{-1/2}t), H = \begin{pmatrix} h_1^2 & 0 & \dots & 0 \\ 0 & h_2^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & h_d^2 \end{pmatrix}. \quad (3.1)$$

The multivariate kernel can be obtained either as a product of multiple one-dimensional kernels or by rotating a one-dimensional kernel in the d -dimensional space. Theoretically, the transform matrix, H , can be completely parameterized, but in practical applications – especially when real-time processing is required – a simplified version of the matrix is used. The simplified version of the matrix is a diagonal matrix and contains on the main diagonal the squares of the bandwidths of the kernels in the d dimensions. The computational complexity can be further reduced, at the cost of optimality loss [8], by using the same bandwidth, h , in all dimensions. In this case the density estimator becomes:

$$\hat{f}_h(x) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right). \quad (3.2)$$

The kernels commonly used in video tracking are radially symmetric and can be written in the following format:

$$K(x) = c_{k,d} k(\|x\|^2). \quad (3.3)$$

In equation (3.3), $c_{k,d}$ is a normalization constant, which guarantees that the kernel has a unitary integral over its support. The $k(\cdot)$ function defines the profile of the kernel and is only defined for positive values, in order to allow a radially symmetric construction of the kernel. The pdf estimator becomes:

$$\hat{f}_h(x) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^n k\left(\left\|\frac{x - x_i}{h}\right\|^2\right). \quad (3.4)$$

The Epanechnikov kernel – which is not continuously differentiable – and the normal kernel – which has an infinite support – are the most widely used kernels [8], [10]. The Epanechnikov and the normal kernel are symmetrical kernels and can be defined using a profile function:

$$k_E(x) = \begin{cases} 1 - x, & 0 \leq x \leq 1 \\ 0, & x > 1 \end{cases} \Rightarrow K_E(x) = \begin{cases} \frac{1}{2c_d} (d+2) (1 - \|x\|^2), & \|x\| \leq 1 \\ 0, & \|x\| > 1 \end{cases} \quad (3.5)$$

$$k_N(x) = e^{-\frac{1}{2}x}, \quad x \geq 0 \Rightarrow K_N(x) = (2\pi)^{-d/2} e^{-\frac{1}{2}\|x\|^2} \quad (3.6)$$

In order to analyze the feature space defined by the probability density function f , the maxima of this function must be identified. The problem is reduced to finding the zeroes of the first derivative (one-dimensional case) or gradient (multi-dimensional case) of the pdf. The mean shift algorithm is able to identify the zeroes of the first derivative/gradient of the pdf, without the need for probability density estimation.

3.1.2. Estimating the gradient of the probability density function

For multidimensional kernels having the same bandwidth, h , for all dimensions, the gradient of the pdf can be expressed as:

$$\begin{aligned} \hat{\nabla} f_{h,K}(x) &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^n (x_i - x) g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) = \\ &= \frac{2c_{k,d}}{nh^{d+2}} \left[\underbrace{\sum_{i=1}^n g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}_{F1 \sim \hat{f}_{h,G}(x)} \right] \left[\underbrace{\frac{\sum_{i=1}^n x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^n g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x}_{F2 = ms_{h,G}(x)} \right]. \end{aligned} \quad (3.7)$$

In equation (3.7) the profile function $g(x)$ is defined as $g(x) = -k'(x)$. The profile g can be used to create the kernel $G(x) = c_{g,d} g(\|x\|^2)$, if the condition

$$\sum_{i=1}^n g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) > 0 \quad (3.8)$$

is fulfilled. All the profiles used in practical applications fulfill the condition (3.8). The K kernel is often referenced as the shadow of the G kernel [42]. The Epanechnikov kernel is the shadow of the uniform kernel, while the normal kernel and its shadow have the same shape.

The F1 term in equation (3.7) is proportional to the probability density at x estimated by using the kernel G , while the F2 term represents the mean shift from the center of the kernel window (the weighted mean is calculated using the weights given by the kernel G). The mean shift can be expressed as [42]:

$$ms_{h,G}(x) = \frac{1}{2} h^2 c \frac{\hat{\nabla} f_{h,K}(x)}{\hat{f}_{h,G}(x)}. \quad (3.9)$$

3.1.3. The mean shift algorithm

The local mean obtained using the weights provided by the kernel G is shifted towards the area of the feature space which contains most of the points. It can be observed in equation (3.9) that the mean shift vector and the estimated gradient have the same direction, always pointing towards the location in which the probability density increases the most. Therefore, based on the mean shift vectors, it is possible to define a trajectory to a local maximum of the estimated probability

density. The maxima of the estimated probability density are stationary points and the mean shift algorithm allows to find them by iterating the following 2 steps until convergence is reached:

- calculate the mean shift vector at the current point, $ms_{h,G}(x)$,
- translate the kernel with $ms_{h,G}(x)$.

It has been proven [42] that it suffices to use a kernel with a convex and monotonically decreasing profile, in order to guarantee the convergence of the algorithm. If the 2 conditions are fulfilled, both the trajectory and the estimated pdf are convergent, and the estimated pdf is also a monotonically increasing function.

For other gradient based algorithms, the shift in the direction of the local gradient does not ensure the convergence, unless infinitesimal steps are used and therefore the step size choice is a problem. The mean shift algorithm can guarantee the convergence under the specified conditions, based on the adaptively calculated value of the mean shift vector's magnitude, which involves using an adaptive step size (the step size decreases as the algorithm gets closer to the searched maxima).

The number of steps required for convergence is related to the specific kernel G used within the algorithm. The uniform kernel allows the algorithm to converge within a finite number of steps, while the other kernels (which give different weights to the points according to the distance from the window center) allow the algorithm to converge when the number of steps tends to infinity. This is why, when the later kernel types are to be used in practical applications, a lower limit on the value of the mean shift has to be imposed, in order to avoid reiterating the algorithm infinitely.

The use of the normal kernel within the algorithm ensures a smooth trajectory (the angle between two consecutive mean shift vectors is always smaller than 90 degrees) towards the stationary point [42] and the results obtained using this kernel are superior to those obtained with a uniform kernel. Nevertheless, the large number of steps required for convergence limits the use of the normal kernel in practical application, where the uniform kernel is often preferred.

The algorithm is attracted by the local maxima, given that the local maxima are unique stationary points within a small neighborhood (within a small sphere). Once the algorithm reaches a point situated close enough to a local maximum, it will converge towards it. The reunion of all the points that converge toward the same local maximum (mode) is referred to as the basin of attraction of that local maximum.

If the local maxima of the probability density are to be found, after the identification of the stationary points using the mean shift algorithm, the stationary points have to be filtered in order to keep only the local maxima. In order to test a stationary point to establish whether it is or not a local maximum, the stationary point is perturbed by a small magnitude noise vector. If the algorithm applied starting from the perturbed point converges again (up to a given tolerance) towards the same point, that point is a local maximum.

3.1.4. The mean shift algorithm in video tracking

3.1.4.1. Target representation

In order to perform the tracking of an object within a video sequence, a set of representative features – that allow for the object to be identified – has to be chosen. These features define the so called feature space. The reference model used

for the object is represented by the probability density function within the chosen feature space.

The target model with the probability density function q is assumed to be centered on the spatial position 0 of the initial frame. In the next frame the tracked object slightly changes its spatial position and, in order to identify the new position of the object, the tracking algorithm needs to select one of the multiple candidate locations. The probability density function evaluated around the candidate position y is $p(y)$. The chosen candidate location is the one for which the probability density function $p(y)$ is closest to the probability density function q of the model.

Both probability density functions, q and $p(y)$, need to be estimated based on the data available in the 2 frames. The robustness of the tracking algorithm can be increased if non-parametric probability density estimation is used. Theoretically, any of the known non-parametric probability density estimators can be used: histograms, averaged shifted histograms, kernel estimators etc. The best estimations are obtained with kernel based estimators [8], but their downside is the high computational power required. The histogram is one of the most basic estimators of the probability density, but has the advantage of low computational power requirements, which recommend it for tracking applications that need to run in real-time [34].

In order to assess the similarity between the probability density function for the candidate location, $p(y)$, and the probability density function of the model, q , a similarity function, $\rho(y)$, is used. The local maxima of the similarity function correspond to the presence of objects similar to the target in the second frame.

If only spectral data are used for target representation, completely discarding the spatial information, the obtained similarity function may have large variations for locations that are relatively close to each other in the analyzed scene. Under such circumstances, it is difficult to apply gradient based methods in order to identify the local maxima, while the alternate option – the exhaustive search – is not appropriate for real-time applications.

The use of spatial information, in addition to the spectral data, for representing the target in the feature space, may lead to a smooth characteristic of the similarity function. The spatial information can be taken into account in the estimation of the probability density function by weighting the pixels within the selected image region, according to their distance from the center of the region [34]. By masking the object with a spatially isotropic kernel, the weights of the pixels around the center of the analyzed region can be increased, to the detriment of the pixels situated near the borders of the region. The pixels situated near the borders receive lower weights, as they are more likely to suffer from interferences from the background or occlusions from other objects in the scene. Usually, in order to increase the robustness of the estimation, a kernel with a convex and monotonously decreasing profile is used.

The region T , chosen as model, is usually an ellipse-shaped region of the image, but the effect of different dimensions in different directions can be removed by normalizing the ellipse to a unity circle [34]. Considering the chosen region as being centered on the spatial position 0 , and the normalized positions of the pixels

within the region, $\{x_i^*\}_{i=1..n}$, a function $b(x_i^*)$ can be defined, in order to associate

to each pixel the corresponding bin in a m -bin histogram. Using these notations and the variable u for iterating through the bins of the histogram, the estimate of the probability density function obtained using the histogram is:

$$\hat{q}_u = C \sum_{i=1}^n k\left(\|x_i^*\|^2\right) \delta(b(x_i^*) - u), \quad u = 1 \dots m, \quad C = \frac{1}{\sum_{i=1}^n k\left(\|x_i^*\|^2\right)}. \quad (3.10)$$

In a similar manner, the probability density for a candidate region in the second frame, centered at the spatial position y and containing the pixels at the positions $\{x_i\}_{i=1 \dots n_h}$, is:

$$\hat{p}_u(y) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{y - x_i}{h}\right\|^2\right) \delta(b(x_i) - u), \quad u = 1 \dots m, \quad C_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\|\frac{y - x_i}{h}\right\|^2\right)}. \quad (3.11)$$

In equations (3.10) and (3.11), C and C_h represent normalization constants leading to a unitary value of the sum of elements in all the bins of the histogram (they guarantee that the probability density function integrates to 1). In (3.11), h is a scale factor which determines the number of pixels of the candidate region. Considering that, for a given candidate region, the center of the region, y , is one of its pixels, the constant C_h does not practically depend on y .

3.1.4.2. The similarity function

The similarity function, ρ , inherits the characteristics of the kernel profile used for spatial weighting. Therefore a kernel with a differentiable profile leads to a differentiable similarity function, for which finding the local maxima can be achieved through fast kernel-based methods.

There are many options available for measuring the similarity degree between 2 probability densities: Minkovski distances in L_p , weighted mean variance, Kolmogorov-Smirnov distance, χ^2 statistics, Kullback-Leibler distance [43], distances based on the Bhattacharyya coefficient [34] etc. Each of these distances proved to offer some degree of efficiency for various computer vision applications [34], [43], but for video tracking applications good results were obtained especially when using distances based on the Bhattacharyya coefficient [34].

The Bhattacharyya coefficient between the probability densities defined in equations (3.10) and (3.11) is defined as:

$$\hat{\rho}(y) \equiv \rho(\hat{p}(y), \hat{q}) = \sum_{u=1}^m \sqrt{\hat{p}_u(y) \hat{q}_u}. \quad (3.12)$$

From a geometrical point of view, the coefficient in equation (3.12) can be considered as the cosine of the angle between the m -dimensional vectors $\left(\sqrt{\hat{p}_u(y)}\right)_{u=1 \dots m}^T$ and $\left(\sqrt{\hat{q}_u}\right)_{u=1 \dots m}^T$, while from a statistical point of view it represents the correlation between the two vectors.

Based on the Bhattacharyya coefficient, a statistical measure can be defined for the distance between the 2 estimated probability densities as:

$$d(y) = \sqrt{1 - \hat{\rho}(y)}. \quad (3.13)$$

The distance defined in (3.13) inherits the advantages of the Bhattacharyya coefficient (i.e. geometrical interpretation and some degree of scale invariance – limited by quantization effects), but also comes with an additional advantage by imposing a metric structure.

3.1.4.3. Target localization

The location of the tracked object in the second frame is assumed to be the one for which the distance in (3.13) is minimum. The minimization of the distance is equivalent to the maximization of the Bhattacharyya coefficient. As spatial information is taken into account, both for the target model and target candidate representation, the Bhattacharyya coefficient (and consequently the distance based on it) is a smooth function, allowing the use of gradient based methods like the mean shift algorithm for finding the maxima.

For the current frame, the search starts at the position \hat{y}_0 , which is the position of the target in the previous frame. The Bhattacharyya coefficient can be approximated using the Taylor series decomposition around $\hat{p}_u(y_0)$:

$$\hat{p}(y) \approx \frac{1}{2} \sum_{u=1}^m \sqrt{\hat{p}_u(\hat{y}_0) \hat{q}_u} + \frac{C_h}{2} \sum_{i=1}^{n_h} w_i k \left(\left\| \frac{y - x_i}{h} \right\| \right), \quad w_i = \sum_{u=1}^m \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0)}} \delta[b(x_i) - u]. \quad (3.14)$$

The maximization of the Bhattacharyya coefficient reduces to the maximization of the second term in equation (3.14), as the first term does not depend on y . The second term in equation (3.14) is actually an estimate of the probability density around the position y in the current frame, using w_i for weighting the data.

In order to find the maximum of this function around \hat{y}_0 , the mean shift algorithm is applied by recursively shifting the kernel from the position \hat{y}_0 to the position \hat{y}_1 , which is calculated using:

$$\hat{y}_1 = \frac{\sum_{i=1}^{n_h} x_i w_i g \left(\left\| \frac{\hat{y}_0 - x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n_h} w_i g \left(\left\| \frac{\hat{y}_0 - x_i}{h} \right\|^2 \right)}, \quad g(x) = -k'(x). \quad (3.15)$$

The block diagram of the algorithm is presented in Fig. 3.2. Since the starting point of the search is the position \hat{y}_0 , first the estimate of the probability density and the Bhattacharyya coefficient are calculated around this position. Next, the weights $\{w_i\}_{i=1 \dots n_h}$ are calculated, then, in the third step, the new location \hat{y}_1 is determined. In the fourth step, the probability density estimation is calculated and the Bhattacharyya coefficient around the new location is evaluated. Next, the

estimated values of the coefficient at the new position \hat{y}_1 and at the initial position \hat{y}_0 are compared.

While the coefficient at the new position is smaller than the one at the initial position, the new position is updated recursively to the position situated at half the distance between the previous position and the initial one, followed by re-estimating the Bhattacharyya coefficient around the updated location. After exiting the above described loop, in the final step a condition for terminating the algorithm is checked,

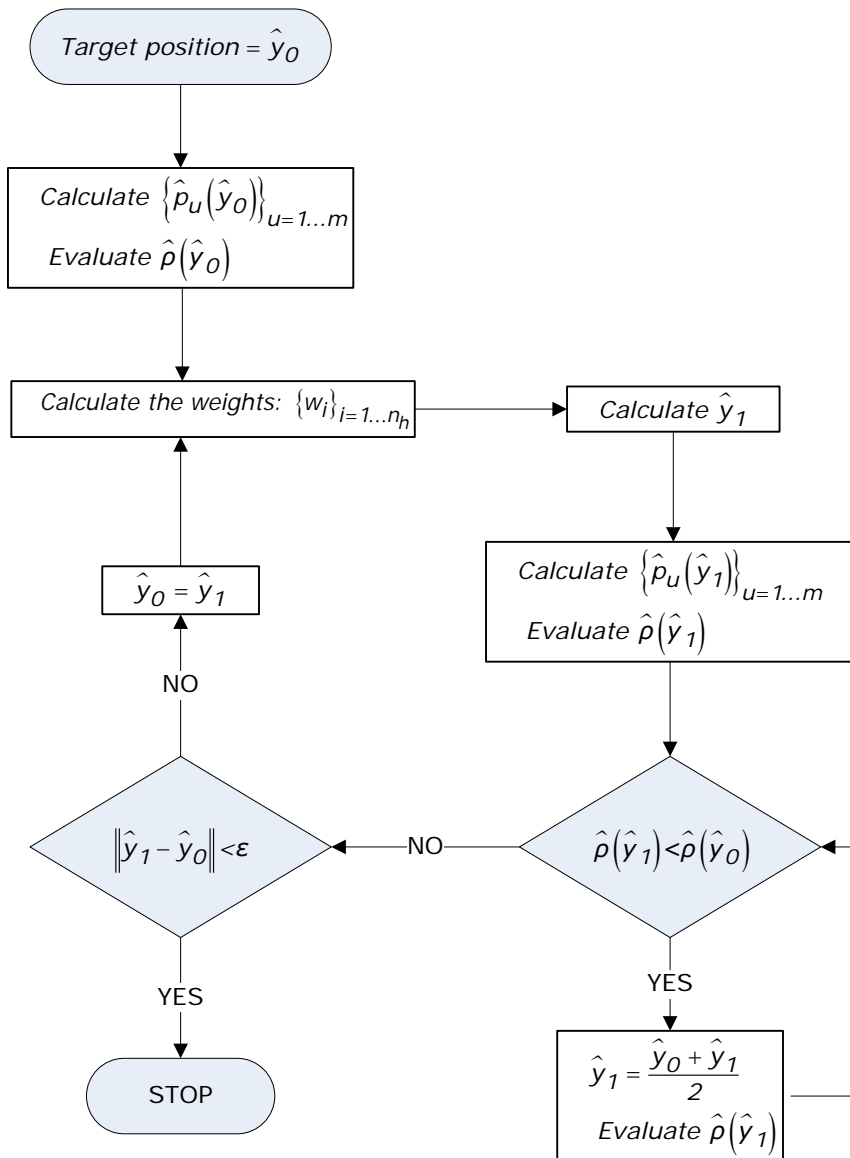


Fig. 3.2. Target localization using the mean shift algorithm.

and, if not fulfilled, the initial position \hat{y}_0 is replaced with the current new position \hat{y}_1 and the algorithm re-iterates starting with step 2.

The threshold ε used in the final step of the algorithm is introduced in order to allow the termination of the algorithm when the two locations (new and old) are within the same pixel area. An upper limit can also be imposed on the number of iterations, taking into account that the average number of iterations required for convergence is 4 [34].

Possible target scale changes are dealt with by running the algorithm multiple times with different bandwidths of the kernel. The use of 3 bandwidths h , $h - \Delta h$ and $h + \Delta h$ with $\Delta h = 0.1 \cdot h$ is sufficient for most practical applications. The kernel bandwidth which produces the highest value for the Bhattacharyya coefficient is selected, but, in order to avoid hypersensitivity to scale changes, the new value of the bandwidth is determined as a weighted average between the old value and the new one, with a higher weight for the older value.

The fifth step of the algorithm seldom improves the results (in approximately 1‰ of the situations) and can be skipped in order to save computational time and resources. Additional computational power savings can be obtained by using the Epanechnikov kernel for calculating the new position in the third step of the algorithm, considering that the derivative of the kernel profile in this case is a constant, thus reducing the computation of the new location to a weighted average [34].

The neighborhood of the initial location in which the algorithm may find the new location is known as the operational basin of attraction. The size of the operational basin of attraction has to be at least equal to that of the target model. In order to have exactly one maximum of the Bhattacharyya coefficient within the operational basin of attraction, the target representation needs to accurately describe the tracked object.

3.2. Filtering and data association

The filtering and data association process relies on the discrete-time modeling of dynamic systems through the state-space [34], [44]. The dynamic analysis and estimation of the states of a moving target rely on 2 basic models:

- the system model, also known as the dynamic model and
- the measurement model, also known as the observations model.

The system model describes the target evolution in time through the state sequence $\{x_k\}_{k \in \mathbb{N}}$. Each state x_k is a (possibly non-linear) function depending on the previous state x_{k-1} and the noise v_{k-1} that affects the previous state, which belongs to the i.i.d. sequence $\{v_k\}_{k \in \mathbb{N}}$:

$$x_k = f_k(x_{k-1}, v_{k-1}). \quad (3.16)$$

The measurement model connects the sequence of measurement results, $\{z_k\}_{k \in \mathbb{N}}$ and the states sequence. The model is described by the equation:

$$z_k = h_k(x_k, n_k). \quad (3.17)$$

Similar to the system model, $h_k(\cdot)$ is a (possibly non-linear) function, while $\{\eta_k\}_{k \in \mathbb{N}}$ is a noise sequence.

The final goal of the filtering and data association process for tracking is to recursively estimate the state x_k using the measurement sequence available at the time moment k , $z_{1:k}$. This is equivalent to the process of building the pdf $p(x_k / z_{1:k})$. Theoretically, the optimal solution to this problem is offered by the Bayesian filter with two recursive steps: prediction and update.

In the prediction step, the *a priori* pdf for the current moment ($t = k$) is determined, using the equation of the dynamic model (3.16) and the pdf of the state calculated at the previous time moment ($t = k - 1$), $p(x_{k-1} / z_{1:k-1})$. The *a priori* probability density is calculated according to the Chapman-Kolmogorov equation:

$$p(x_k / z_{1:k-1}) = \int p(x_k / x_{k-1}) p(x_{k-1} / z_{1:k-1}) dx_{k-1}. \quad (3.18)$$

In the update step, the *a posteriori* pdf $p(x_k / z_{1:k})$ is calculated taking into account the newly obtained measurement result at time moment k and the probability, $p(z_k / x_k)$, to observe the measurement result z_k given the system state x_k . The Bayes rule is applied in this step for updating the *a posteriori* probability density using the *a priori* probability density of the state x_k – obtained in the prediction step and the result of the most recent measurement z_k :

$$p(x_k / z_{1:k}) = \frac{p(z_k / x_k) p(x_k / z_{1:k-1})}{p(z_k / z_{1:k-1})}, \quad (3.19)$$

$$p(z_k / z_{1:k-1}) = \int p(z_k / x_k) p(x_k / z_{1:k-1}) dx_k.$$

For the particular case when both the f_k and h_k functions are linear and the noise sequences $\{v_k\}$ and $\{\eta_k\}$ are Gaussians, the optimal solution is given by the Kalman filter and the resulting *a posteriori* probability density is also Gaussian. If the functions f_k and h_k are non-linear, the EKF (Extended Kalman Filter) can be used to obtain a normal *a posteriori* probability density, after linearizing the 2 functions. Other available options for approximating the optimal solution are the UKF (Unscented Kalman Filter) [45], the approximate grid based methods and the particle filters [44].

3.2.1. Kalman filters

The use of Kalman filters leads to an optimal solution, at the cost of imposing restrictions both on the functions that define the dynamic and the observations models and on the 2 noise sequences. The resulting *a posteriori* pdf is Gaussian in this case and is characterized by the mean and the covariance.

If the functions that describe the 2 models are linear, they can be described by the following equations:

$$x_k = F_k x_{k-1} + v_{k-1} \quad (3.20)$$

$$z_k = H_k x_k + n_k \quad (3.21)$$

F_k and H_k are the matrices that define the 2 linear functions. For simplification, the noise sequences $\{v_{k-1}\}$ and $\{n_k\}$ are assumed to be statistically independent, having zero mean and covariances Q_{k-1} and R_k . The recursive relations that define the Kalman filter can be written as [44]:

$$p(x_{k-1} | z_{1:k-1}) = N(x_{k-1}; m_{k-1|k-1}, P_{k-1|k-1}) \quad (3.22)$$

$$p(x_k | z_{1:k-1}) = N(x_k; m_{k|k-1}, P_{k|k-1}), \quad (3.23)$$

$$m_{k|k-1} = F_k m_{k-1|k-1}, \quad P_{k|k-1} = Q_{k-1} + F_k P_{k-1|k-1} F_k^T$$

$$\begin{aligned} p(x_k | z_{1:k}) &= N(x_k; m_{k|k}, P_{k|k}), \\ m_{k|k} &= m_{k|k-1} + K_k (z_k - H_k m_{k|k-1}), \\ P_{k|k} &= P_{k|k-1} - K_k H_k P_{k|k-1} F_k^T, \\ K_k &= P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \end{aligned} \quad (3.24)$$

In equations (3.22), (3.23) and (3.24), $N(x; m, P)$ refers to a normal distribution with mean m , covariance P and argument x , while the term K_k is known as the Kalman gain.

3.2.2. Particle filters

The most general class of filters used in the filtering and data association process is represented by the particle filters. Particle filters rely on Monte Carlo methods. They impose no restrictions on the functions f_k and h_k and on the two noise sequences $\{v_k\}$ and $\{n_k\}$. In computer vision, the particle filters were introduced under the name of ConDensation (**Conditional Density Propagation**) algorithm by Isard and Blake [39]. The basic algorithm is known under the name of SIS (Sequential Importance Sampling) and many other versions are derived from it: SIR (Sampling Importance Resampling), ASIR (Auxiliary Sample Importance Resampling) or the RPF (Regularized Particle Filter).

The SIS algorithm is a method of implementing the recursive Bayesian filter through Monte Carlo simulations. The *a posteriori* pdf that is to be determined is represented through a set of random samples, each sample having an associated weight. Estimates are calculated using these samples and their corresponding weights. The higher the number of available samples, the better this method tends to approximate the optimal Bayesian estimate. Considering the set of samples (particles) $\{x_{0:k}^i\}_{i=1:N_s}$ and the corresponding weights $\{w_k^i\}_{i=1:N_s}$ at the time moment $t = k$, the *a posteriori* probability can be expressed as a weighted average of the N_s samples:

$$p(x_{0:k} / z_{1:k}) \approx \sum w_k^i \delta(x_{0:k} - x_{0:k}^i). \quad (3.25)$$

As the samples cannot be taken from the $p()$ density, they are obtained from a different density, $q()$, known as importance density, which can be chosen freely. The desired pdf can still be determined as a weighted average of the samples obtained this way, if the weights are chosen so as:

$$w_k^i \propto \frac{p(x_{0:k}^i / z_{1:k})}{q(x_{0:k}^i / z_{1:k})} \quad (3.26)$$

If $q()$ can be factorized as:

$$q(x_{0:k} / z_{1:k}) = q(x_{0:k} / x_{0:k-1}, z_{1:k}) \cdot q(x_{0:k-1} / z_{1:k-1}), \quad (3.27)$$

The samples at time moment k can be obtained based on the samples at the previous time moment, $k - 1$, by adding the new state x_k^i . The weights at time moment k can be calculated recursively, based on the weights at the previous time moment. For the simplified case when the state at time moment k depends only on the previous state (not on the entire trajectory), the current weights can be determined using the equation [44]:

$$w_k^i = w_{k-1}^i \frac{p(z_k / x_k^i) p(x_k^i / x_{k-1}^i)}{q(x_k^i / x_{k-1}^i, z_k)}. \quad (3.28)$$

The *a posteriori* pdf can be approximated as:

$$p(x_k / z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i). \quad (3.29)$$

A known problem for this algorithm is the so-called degeneration process: after a few iterations most particles, except for one, will have the weights almost zero. This means that all calculations performed for the particles with small weights will have negligible effects on the final result. The effective size of the sample set (which is always lower or equal to the size of the sample set) is a measure of the degree of degeneration of the algorithm and can be estimated as:

$$\hat{N}_{eff_k} = \frac{1}{\sum_{i=1}^{N_s} (w_k^i)^2}. \quad (3.30)$$

Small values of the estimated effective size indicate an advanced degree of degeneration. The use of a very large sample set may diminish the effects of degeneration, but this solution is not appropriate for practical applications, as it would drastically increase the computational power requirements. In practical applications, the effects of degeneration are limited through an adequate choice for the $q()$ function and through resampling. The optimal importance density function, $q()$, is [44]:

$$q(x_k | x_{k-1}, z_k)_{opt} = p(x_k | x_{k-1}, z_k). \quad (3.31)$$

The optimal importance density in (3.31) can be used only in isolated situations. As an alternative, the *a priori* pdf, $p(x_k | x_{k-1})$, can be used, with the drawback of ignoring the measurement result. Resampling is based on the idea of replacing the sample set and the corresponding weights when a significant degeneration degree is detected (i.e. \hat{N}_{eff_k} in equation (3.30) falls below a given threshold). The new sample set and weights are more appropriate for the calculation of the *a posteriori* pdf. The resampling removes particles with small weights and generates new particles based on those with larger weights, which leads to another problem, known as impoverishment of the data set or loss of diversity. If the process noise, $\{v_{k-1}\}$, is small, it is possible that (after a reduced number of resampling iterations) all the remaining particles will be derived from the same initial particle.

There are multiple types of particle filters, each of them being adapted, from the performance point of view, for a specific application domain. When designing a particle filter for a given application, the most important aspect is to choose an appropriate importance density function, as any refinements in the algorithm rely on the chosen importance function [44].

3.3. Hand tracking solutions

The tracking of hands and/or fingers has been extensively studied, considering its importance in applications like human computer interfaces [46], augmented reality [47], sign language recognition [48], handwriting recognition systems [49], etc.

Hand and finger tracking methods can be classified as model based and view based methods. The model based methods [50], [51] use articulated models of the hand and analysis by synthesis allowing for a very detailed characterization of the hand posture and fingers pose. These methods achieve a good robustness to disturbing factors like camouflage and partial occlusions, but their practical usage is limited due to the high computational power and expensive equipment requirements. View based methods [52], [53] rely on the visual appearance of the hand in image sequences and require less computational power than the model based ones, allowing for real-time implementation even with cheaper hardware equipment. However, the methods in this category are able to provide only a global, less detailed characterization of the hand posture and fingers pose which limits the possible applications to those which do not need details that are not available.

Recently, many successful solutions to the hand tracking problem rely on active systems like Kinect [54] or time-of-flight cameras [55]. These approaches take advantage of the depth information provided by the special devices, but their application range is also limited due to the current unavailability of the active systems in some scenarios.

Hand tracking performed using a single camera is still attracting for researchers in the field of vision based human computer interaction [46], [56] given the current ubiquity of webcams which enable a potential development of applications based on this type of tracking to a wide variety of devices, including the mobile ones.

Human body parts tracking based on a single feature, like in [22], is not robust, as in real-life situations the feature may become unreliable. Increased robustness can be achieved when multiple features are used. In [53] Flocks of Features were used for hand tracking with significant performance improvement compared to [22]. Further performance improvements were reported in [57], based on velocity-weighted features and color cue.

Skin color is widely used as a feature for hand tracking. Many skin detection algorithms are available in the literature, using various skin classification strategies in a wide variety of color spaces. [58] contains a good survey on the performances of skin color classification methods in different color spaces.

Video sequences may suffer of motion blur especially when acquired with cheap webcams and under poor lighting conditions with fast moving targets, but few authors analyze the hand tracking performance in such conditions. In [59] blurred target templates were used together with an SVM classifier to improve the performances of a mean-shift tracker in motion blurred sequences, at the cost of increased computational complexity. Good results in tracking blurred targets is also reported in [60] where a unified sparse framework is used to deal with both, the tracking and motion from blur, problems.

Line strip features extracted by specific processing of low level features were used in different formats for finger detection and tracking in [49] and [2].

3.4. Conclusions

There are 2 main categories of tracking methods: deterministic – with mean shift as main representative algorithm – and stochastic – best represented by particle filters. Each of the 2 categories has advantages and disadvantages.

Particle filters maintain multiple hypotheses simultaneously and use a stochastic motion model for the prediction of the new position of the tracked object. Algorithms from this category have better performance in following the target through cluttered environments and are able to recover after errors or temporary target confusion. The main disadvantage of these algorithms is represented by the high computational power requirements [41].

The mean shift algorithm and other algorithms related to it are able to track the object using a single hypothesis, requiring significantly less computational power. The main disadvantage in this case is that the algorithm is not able to recover from errors – once the algorithm reaches the basin of attraction of another local maximum, it will converge to the new maximum (which does not represent the target object).

The Kalman filter was used together with the mean shift algorithm [34] in order to provide a position prediction. Hybrid algorithms – like the Mean Shift (Embedded) Particle Filter [21], [41] – were also developed, combining the advantages of mean shift and particle filters. The hybrid algorithm achieves good tracking performance in cluttered environments and reduces the computational effort required for each frame by using a reduced number of particles. Though the hybrid algorithm, once implemented, allows only for minor variations of the target model, extensions are available in order to allow an adaptive target model, at the cost of increased computational power requirements.

4. ROBUST INTEGRATION OF MULTIPLE FEATURES FOR HAND/FINGER TRACKING

The tracking of human body parts represents one of the most challenging problems in the implementation of a gesture-based human computer interaction (HCI) system. Hand and finger tracking represent important parts of gesture-based user interfaces. Tracking algorithms which rely on a single feature may have good performance in severely constrained environments, but often lose the target when the tracked feature becomes unreliable. Increased tracking robustness can be achieved when multiple features are used. Features which can help to distinguish the target object from other objects are necessary, both during tracker initialization and during the actual tracking process.

Fig. 4.1 presents the process of tracking human body parts as a link between the low level image processing layers and the high level gesture

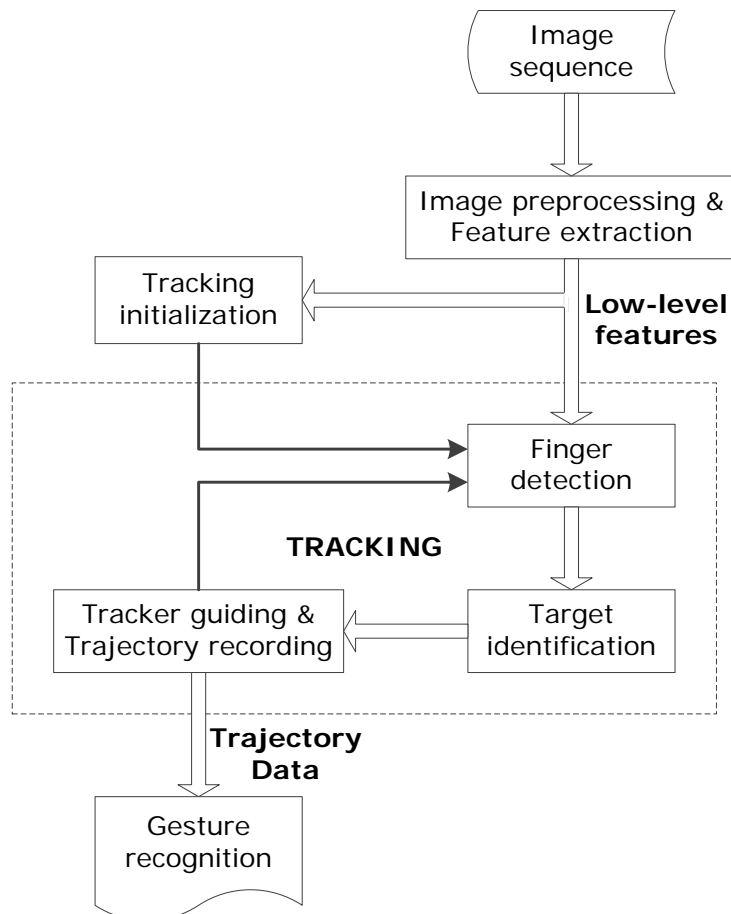


Fig. 4.1. The main processing stages of a vision-based gesture recognition system.

recognition layers, within the block diagram of a dynamic gesture recognition system. In a vision-based dynamic gesture recognition system, the tracking part is responsible with the processing of the low-level feature data, converting them to trajectories of the target. The trajectory data obtained by the tracking algorithm are then passed to the gesture recognition layers for further processing.

The next section discusses the feature selection process in the context of hand/finger tracking.

The problem of tracking initialization is presented in section 4.2, as well as a solution I have proposed in paper [1].

Sections 4.3, 4.4 and 4.5 present a method of integrating information from multiple features for robust tracking of the finger in challenging situations. The tracking method, presented initially in papers [3] and [4], includes multiple processing layers which are applied successively to the data. Section 4.3 focuses on the processing stages that lead to finger detection, then section 4.4 describes the processing steps required for target identification and validation. Section 4.5 presents the high-level processing layers, responsible for the tracker guiding and trajectory recording.

The last sections of this chapter analyze the performance of the tracking algorithm described in sections 4.3 – 4.5.

4.1. Features used in hand/finger tracking

The success of a tracking algorithm is strongly influenced by the features used for identifying and tracking the target.

Most early tracking algorithms rely on a single feature for tracking the target. In hand/finger tracking, the most commonly used low-level features are color, contours, foreground and motion.

In many situations the object to track is a foreground object; therefore, an accurate background subtraction algorithm can significantly reduce the amount of data to process for a tracking algorithm.

Trackers that rely on a single feature are likely to lose the target when that feature becomes unreliable. The use of multiple features leads to an increase in robustness. Both the tracking algorithm proposed in [3] and [4] and the tracking initialization method proposed in [1] rely on multiple characteristics (features) of the finger: foreground object, color, shape, proportionality.

Color segmentation and background subtraction are used as preprocessing steps, providing the low level input data for the detection and tracking algorithms. As the finger to track is a foreground object, background subtraction is used first to separate foreground objects from the background. Although this step can significantly reduce the search area and the data to process, it is not sufficient to uniquely identify the finger, as in most situations the finger is not the only foreground object in the image. The next feature that is used to identify the finger is the color (skin color). Background subtraction and the removal of non-skin colored foreground objects significantly reduce the data for further processing. Using the finger size information available from the previous frame, finger features are extracted, leading to a sparse representation of data through line strips. Shape and proportionality criteria are then applied on the remaining data, in order to identify the line strips that belong to the finger. The shape and proportionality are assessed within the tracking algorithm.

4.1.1. Color

In image and video processing applications, color is an essential characteristic of objects. The frames obtained from the camera are represented based on the Red, Green, Blue (RGB) color space.

RGB is a device specific color space, as it was designed for CRT displays. RGB uses additive color mixing, which describes what kind of light must be emitted in order to produce a given color, starting from complete darkness. CMY (Cyan, Magenta, Yellow) is another device specific color space, designed for printers and uses subtractive color mixing (i.e. it describes what kind of inks must be applied on the paper in order to obtain a given color starting from white).

In real situations, lighting is not constant and a multitude of factors can cause variations in lighting (e.g. shadows from other objects, self-shadowing, switching between sunlight and overcast, artificial light etc.). The same object, exposed to different lighting conditions, appears to have different colors. Since the color of the object is used for tracking, the separation of brightness from chrominance is essential in order to have at least one component of the color model invariant to lighting changes. The RGB color space is not well suited for tracking, as it does not isolate the brightness and chrominance information (all the three components vary with illumination changes). A lot of research has been conducted in order to define lighting invariant color functions. The most widely used functions are the chromatic color models: normalized RGB, YUV, YCrCb, HSV and CIELAB [61].

CIE (fr. *Commission Internationale de l'Eclairage*) defined in 1931 the first color space based on measurements of human color perception, CIEXYZ. CIEXYZ is the basis for almost all other color spaces. The same CIE defined the CIELUV color space as a modification of CIEXYZ, in order to display color differences more conveniently. The necessity of a more perceptual linear color space led in 1976 to the definition of the CIELAB color space. In CIELAB, the three coordinates represent the lightness of the color (L^*), its position between magenta and green (a^*) and its position between yellow and blue (b^*).

YUV and YCrCb are other color spaces, which use one lightness and two chrominance coordinates. YUV is used in PAL TV systems, while YCrCb is used in the very popular image and video compression standards JPEG and MPEG.

The HSV (Hue, Saturation, Value) color space attempts to describe the perceptual color relationships more accurately than the RGB, while preserving a low computational complexity. The three coordinates represent the color (hue), color's concentration (saturation) and brightness (value). A conic representation of the HSV space is shown in Fig. 4.2. Hue is defined as an angle between 0° and 360° , while saturation and value range each from 0 to 1.

The relations (4.1), (4.2) and (4.3) define the transformation from RGB to HSV.

$$H = \begin{cases} 0, & \text{if } MAX = MIN \\ \left(0 + \frac{G - B}{MAX - MIN}\right) \cdot 60^\circ, & \text{if } R = MAX \neq MIN \\ \left(2 + \frac{B - R}{MAX - MIN}\right) \cdot 60^\circ, & \text{if } G = MAX \neq MIN \\ \left(4 + \frac{R - G}{MAX - MIN}\right) \cdot 60^\circ, & \text{if } B = MAX \neq MIN \end{cases} \quad (4.1)$$

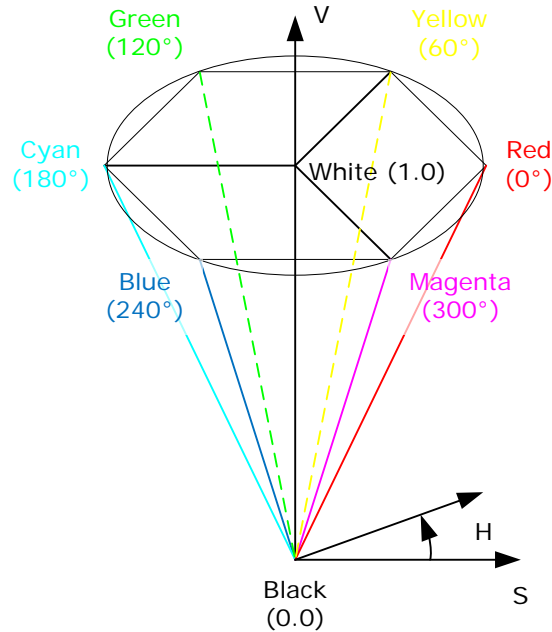


Fig. 4.2. Conic representation of the HSV color space.

$$S = \begin{cases} 0, & \text{if } MAX = MIN \\ \frac{MAX - MIN}{MAX}, & \text{otherwise} \end{cases} \quad (4.2)$$

$$V = MAX. \quad (4.3)$$

The HSV color space is usually preferred in human skin tracking applications [22], because hue is less sensitive to different skin colors and because it is more robust to illumination changes. It was observed [22] that the hue of human skin is the same for all races, except the albinos. Different races' skins differ only in color saturation (i.e. dark-skinned people have greater saturation, while light-skinned people have lower saturation).

It must be noted that for all the lighting invariant color functions mentioned above, the lighting invariance is guaranteed only under some particular assumptions. Violating these assumptions can severely influence the results of color analysis methods which use these color spaces.

As shown in Fig. 4.2, HSV assumes that black is represented as $R = G = B = 0$, and all colors meet each other at this point at reduced brightness. Another assumption of HSV is correct white balance (i.e. all unsaturated colors – grays – have $R = G = B$). Violation of these assumptions may be caused by incorrect white balance, non-ideal camera sensitivity and heterogeneous lighting [61].

Theoretically, hue is invariant with illumination changes, but in the case of pixels with low brightness, the R , G and B values obtained from real cameras are low and severely affected by the camera noise. Transforming these values to HSV leads to low values of V and S , and noisy H , as shown in Fig. 4.2. The hue may also become noisy when the saturation is low, regardless of V . Depending on the lighting

source, brightly illuminated objects of light colors (i.e. high V) can get a hue close to that of the human skin. In order to use hue for tracking, all the pixels which may produce a confusing hue for the tracker must be removed from analysis.

Many skin detection algorithms are available in literature, using various skin classification strategies in a wide variety of color spaces. Most skin color segmentation methods are influenced by the choice of color space. A good survey on skin color segmentation methods is presented in [58]. The most widely used skin color classification methods are: classification based on direct color space thresholding, classification based on histogram models, Gaussian classification, Elliptical boundary model, MLP (Multi-Layer Perceptron), SOM (Self Organizing Map), Maximum entropy based classification and classification using Bayesian networks.

Direct color space thresholding methods can achieve good skin detection rates, at the expense of high false positive rates. The methods relying on histogram models require very large training data sets in order to achieve good results and also have high storage space requirements. The classification methods based on Gaussian models can lead to good results using less training data than the histogram based methods, but their initialization and iterative training are still computationally intensive. The “elliptical boundary model” has similar performance to Gaussian based models, but can only be used for binary classification. Neural network based methods and Bayesian network based methods have similar results and both outperform other skin classifiers, but they require extensive training and are computationally intensive.

In the tracking and tracking initialization algorithms proposed in [1], [3] and [4], I used explicit threshold based methods for skin color segmentation in the RGB and HSV color spaces, considering the need for real-time execution, not only for skin segmentation, but also for additional complex tasks, like tracking algorithm and gesture recognition. In the RGB color space, an enhanced version of the filtering approach proposed by Tomaz et al. in [62] is used, in order to exclude non skin-colored pixels from further processing. Given the range $[0, 255]$ used for representing the R, G and B components, pixels which fulfill any of the following conditions are excluded from further processing:

- $(B > 160)$ and $(R < 180)$ and $(G < 180)$ – too blue,
- $(G > 160)$ and $(R < 180)$ and $(B < 180)$ – too green,
- $(G < 30)$ and $(R > 150)$ and $(B < 30)$ – too red,
- $(R < 100)$ and $(G > 100)$ and $(B < 100)$ – too dark,
- $(|R - B| < 20)$ and $(|R - G| < 20)$ – near grey),
- $(G > 200)$ – too green,
- $(R + G > 400)$ – too yellow,
- $(G > 150)$ and $(B < 90)$ – yellow like (non-skin),
- $B / (R + G + B) > 0.4$ – too blue,
- $G / (R + G + B) > 0.4$ – too green,
- $R < 100$ and $G \in (100, 140)$ and $B \in (110, 160)$ – ocean

Details on the thresholding applied in the HSV color space are provided in section 4.2.2.

Direct threshold based methods have low computational requirements compared to other skin color segmentation methods. Despite the fact that they provide good true positive rates at the cost of high false positives, the threshold based methods are a good option when color is not the only feature taken into account in order to detect the target (i.e. hand/finger). The additional features can help in significantly reducing the false positive rate and lead to good results in target identification.

4.1.2. Background subtraction

Background subtraction represents a widely used approach in computer vision applications for discriminating moving objects from the rest of the scene. Some of the most popular applications of background subtraction are: video surveillance, content based video coding, HCI, motion capture etc. Tracking algorithms can also take advantage of background subtraction's results, given that the target is always a foreground object. Because of its wide range of applications, background subtraction attracted many researchers in the field of computer vision and – as a result – many background subtraction solutions emerged. Nevertheless, none of the available solutions is able to outperform the others from all points of view and for any application field. A short review of the main background subtraction algorithms can be found in [63], while a more recent and comprehensive survey on background subtraction is presented in [64]. Some of the most widely used background subtraction algorithms are: frame differencing, running Gaussian average, temporal median filter, Mixture of Gaussians, kernel density estimation, codebook, spatial co-occurrence of image variations etc.

Frame differencing is the simplest method, where the background model is represented by a frame captured when no foreground objects are assumed to be present in the scene. The method is not computationally intensive, but neither is it robust, being able to provide good results only when all background pixels do not change over time (assumption which generally does not hold for real life scenarios).

The running Gaussian average [65] models the background at each pixel position using a Gaussian pdf, with the mean and variance updated based on the last n values of the pixel. The computational complexity as well as the memory requirements remain low for this method, too. The running average used in updating the parameter of the background model increases the robustness of the algorithm compared to frame differencing, but the accuracy is relatively limited compared to other algorithms.

The temporal median filter [66], [67] and [68] algorithms rely on using the median of the n temporal samples (or of a subset of them) instead of the mean. The median filter increases the robustness of background detection, compared to the algorithms relying on the temporal averaging of pixels, but requires large buffers to store the previous n values for all pixels.

A very popular approach for modelling complex non-static backgrounds is the use of probabilistic parametric methods [69], [70] and [71]. GMMs (Gaussian Mixture Models) represent the background model for each pixel, using a weighted sum of Gaussian distributions in a color space. GMMs suffer trade-off problems. When few Gaussians are used, fast changing backgrounds cannot be modelled accurately, while a large number of Gaussians significantly increases the computational complexity of the background subtraction algorithm. Another trade-off concerns the learning rate: quick rates are necessary to adapt to sudden changes in the background, but a too fast learning rate may lead to the inclusion of foreground pixels into the background model.

Non-parametric models of background, using kernel based density estimation, were proposed in [72], [73] and [74], in order to try to overcome the drawbacks of the parametric models. The algorithms based on non-parametric models are able to adapt to sudden changes in the background and to detect foreground objects accurately. The memory and computational power requirements in this case are significant, especially for methods using the mean shift algorithm and for scenarios where long time periods are needed for sampling the background.

The codebook model [75] also makes no parametric assumptions on the background model and is able to model mixed backgrounds through the use of multiple codewords. The algorithm builds a background model from long term observations of the scene. The background model for each pixel is represented as a codebook containing one or more codewords. The number of codewords/codebook may vary from pixel to pixel. The detection consists in testing the input image against the background model with respect to color and brightness. Pixels are classified using a nearest neighbor method. If the pixel is within a chosen color range of a codeword and its brightness also lays within a given range around the brightness value corresponding to the same codeword, the pixel is classified as background, otherwise as foreground. The memory requirements are moderate while the computational complexity of the algorithm itself is low and the accuracy competes with GMM and kernel density estimation methods. A drawback of the algorithm is that it gives best results with a custom color space representation, thus requiring time consuming color space conversions.

Some authors [76], [77] and [78] explored the idea of using the spatial co-occurrence of pixels in addition to the temporal averaging. The methods in this category offer relatively good accuracy, having moderate computational complexity and memory requirements.

Considering the level where the background model is created, background subtraction algorithms can also be classified as pixel-based or region-based background subtraction algorithms. Recently, hybrid solutions using hierarchical background models [79], [78] were also proposed, in order to combine the advantages of the two representations.

For the background subtraction preprocessing steps of the algorithms proposed in papers [1], [3] and [4] I chose the codebook method [75], considering the availability of an implementation of this method in the OpenCV library and the satisfactory results it provides under normal conditions of illumination and illumination variations.

4.2. Tracking initialization

An important aspect concerning the practical applicability of vision-based gesture recognition systems and of video trackers, in general, is the initialization of the tracker. The initialization of the tracker is the process in which the tracker identifies for the first time the object to be tracked. The initialization can be implemented in various ways, depending on the nature of the target and the tracking principle. Considering the degree of involvement of a human operator, the tracker initialization can be classified as manual or automatic.

In the field of video tracking, the researchers usually focus on the tracking algorithm and seldom pay attention to the initialization of the tracker. Many authors either choose to leave open the problem of tracker initialization, or use manual initialization. Nevertheless, for gesture recognition and HCI applications, the availability of a tracking initialization procedure, which produces repeatable results requiring no external operation, is important.

In the case of manual initialization, a human supervisor needs to indicate the target object to the tracker (e.g. indicate using a pointing device). This type of initialization may be suitable in order to prove the functional principle of a tracker or gesture recognition system [6], [80], [81], but is generally not adequate for practical applications like gesture-based interfaces.

The automatic initialization does not require any intervention from a human supervisor. In this case, the system must be able to automatically identify the target and focus on it [82], [83] and [84]. This type of initialization is used especially in tracking algorithms that rely on the “tracking by detection” principle, where the automatic detection of target candidates in every frame represents the core of the tracking algorithm.

In paper [1], I proposed a semi-automatic finger tracking initialization method for use in monocular vision environments, with protection against unwanted triggering of the tracking process. The initialization method was developed for the initialization of a finger (index) tracker used in dynamic gesture recognition. The proposed method automatically initializes the tracker only when the target hand/finger appears in a specific area of the image. The semi-automatic initialization solution was later used for the initialization of the hand/finger tracking method proposed in paper [3] and extended in paper [4].

I classified the method proposed in [1] as semi-automatic not because it would need some degree of intervention from a third party, but because it requires some degree of attention from the user who intends to use the gesture based interface (which relies on the finger tracking results). The user’s attention is necessary in order to avoid tracking the finger (and subsequently gesture recognition) when the user does not announce his interest in communicating through the gesture based interface. In a gesture based interface application, the goal is not to track any finger which appears in the scene and try to identify valid gestures from its movements, but to track the finger of a user who consciously tries to communicate through the gesture based interface.

Apparently, a semi-automatic initialization is inferior to a completely automatic one – in many applications, a completely automatic initialization of the tracker is preferable to a semi-automatic one. However, automatic initialization requires more flexibility in hand detection (any hand should be detected anywhere within the visible scene), leaving more room for unintended gesture detection. The proposed method imposes constraints on the initialization in order to avoid accidental recognition of unintended gestures, by requiring the user to announce his intention to communicate through the gesture based interface. Since in the proposed approach initialization occurs only in the specified area and only if the hand is maintained in that area for a given (short) time interval, the probability of unintended tracker initialization is reduced (i.e. the tracker cannot be initialized when a user hand moves around the interest area if the user did not confirm his intention to access the interface).

Another advantage of the constraints imposed for the semi-automatic initialization is that they allow a more reliable identification of the target, which makes it an attractive option not only for the particular case of dynamic gesture-based interfaces, but also for the more general category of the semi-supervised trackers. Semi-supervised trackers learn the target model in the first frame, then, during the tracking process perform no (or insignificant) updates; therefore, it is important to have a reliable model of the target from the initialization phase.

The following subsections provide a description of the hand/finger tracking initialization solution proposed in [1].

4.2.1. Conditions for hand/finger detection

According to the constraints imposed, the tracker can only be initialized by the presence of a hand, in a specific position and in a specific area of the image. To



Fig. 4.3. Screen capture – user tries to initialize the tracking process.

guide the user on the required hand pose and location, while waiting for the initialization of the tracker, a hand contour is displayed on a monitor, over the image captured by the camera used by the HCI, as shown in Fig. 4.3.

The tracker for which the proposed method was developed is part of a dynamic gesture recognition system. Therefore, tracking should only start when a user wants to use the gesture based interface. When the tracker's target is a hand or a finger, an object in a frame must simultaneously fulfill the following criteria, in order to trigger the initialization of the tracker:

- foreground object
- color (skin)
- shape/pose
- location within the image

First of all, the target object (i.e. hand/finger) must be a foreground object. In fact, this is a general condition that can be applied for trackers of any type, as normally the target of a tracker is a foreground object.

Another characteristic of the target is the uniform color (skin color).

The first two criteria significantly reduce the data for processing, but a third criterion of shape/pose is required, in order to distinguish a hand/finger from other skin colored foreground objects. Also, the constraints on shape/pose together with those on location within the image help to avoid false triggering of the tracker initialization. The hand may have various appearances in a monocular vision frame. Accidental triggering of the tracker initialization must be avoided, because the user must consciously start to use the gesture based interface.

4.2.2. The hand detection algorithm

A preliminary processing of the video stream for the detection of the hand/finger is background subtraction. Background subtraction is an important step towards hand segmentation, resulting in considerable reduction of the processing data. In applications which use the assumption that the only foreground object in the scene is the hand to track, this step can directly locate the position of the hand. Such an assumption is generally not acceptable for practical situations and therefore additional steps are required in order to distinguish the hand/finger to track from other foreground objects.

The next step of processing implements the color criterion, which is applied to the foreground objects detected in the previous step. HSV color space is useful for identifying skin colored objects. Skin appears to have the same hue for all humans (except for albinos) [22]. Different races' skin differs only in color saturation (i.e. dark-skinned people have greater saturation, while light-skinned people have lower saturation). Considering this property, a simple threshold based skin detector can be implemented, in order to distinguish between skin and non-skin foreground objects. Two auxiliary binary images are generated using thresholds in the 3 dimensions of the HSV color space:

- an image of valid skin color pixels – in which the pixel positions, for which all the skin color criteria are fulfilled, are set to white and the remaining pixels are set to black and
- an image of pixels which cannot be directly classified as skin or non-skin – in which the pixel positions for which the hue is not reliable are set to white and the remaining pixels are set to black.

The values of S and V range between $[0,255]$, while H is circularly defined in the range $[0,180]$. A confidence interval in the H domain is defined $[170,50]$ so that it covers the hue range for normal human skin color. Thresholds are also required in the S and V domains, in order to identify the pixels for which the hue is not reliable:

- pixels with too low saturation,
- pixels with too low brightness (value),
- pixels with too high brightness.

In the saturation domain, a single threshold (set to 30) is required, in order to identify the pixels with a too low saturation. A minimal (40) and a maximal (245) threshold are imposed in the value (brightness) domain.

Pixels with a reliable hue, within the skin confidence interval, are considered skin colored pixels and marked correspondingly in the image of valid skin pixels. Pixels which do not fit the limitations in the saturation and value domains do not present a reliable hue. These pixels cannot be directly classified as skin or non-skin colored, based on their hue and therefore they are marked in a separate auxiliary image. A decision whether these pixels are to be considered skin or not is made later, based on the shape and location constraints.

The shape/pose and location criteria are implemented together using a hand shaped binary mask of 100×200 pixels (for image resolution of 640×480 pixels). This mask is used together with the auxiliary binary images obtained after the previous step for detection of the hand presence. Thresholds are applied on the percentages of pixel matches in order to decide whether a hand is detected or not. First, the region of interest of the image of valid hue skin colored pixels is compared with the hand shaped mask. Both images are binary, and a pixel-wise comparison is made in order to determine the percentage of matching pixels. The percentage of matching pixels in the two images is compared with a threshold to decide whether further investigation of non-reliable hue pixels is necessary. If the percentage of matching pixels is below this threshold, no reliable decision can be made on the hand presence, and in this case the hand is considered not detected. If the percentage is above this threshold, the second auxiliary image is taken into account. If any non-reliable hue pixels were marked in the second auxiliary image, they will be used to increase the matching percentage. White pixels from the second auxiliary image, which correspond to positions within the hand mask, are classified as skin colored and those which correspond to positions outside the mask are classified as

non-skin colored. The matching percentage is recalculated and compared with a new threshold (higher than the one used at the previous step). The hand is considered detected only if the percentage is above this threshold. The values of the thresholds were determined experimentally, in order to allow a comfortable initialization, while avoiding false hand detection.

4.2.3. Tracker initialization

The hand detection procedure described above is the basic part of the proposed tracker initialization method. In order to avoid false triggering, the tracker is not initialized after the first detection of the hand. A state machine controls the tracker initialization and the basic tracking functions. Three states are defined:

- SEARCH,
- CONFIRM and
- FOUND.

Fig. 4.4 presents the three states and the possible transitions between them. Fig. 4.5 presents the outline of the tracker initialization process. The first two processing steps – background subtraction and color space analysis – are applied to all frames, regardless of the current state. Then the processing is state dependent and different tasks are performed in each state.

The system starts in the SEARCH state. In this state, at each frame, hand detection is attempted. When the hand is successfully detected, the system advances to the next state: CONFIRM. The purpose of the CONFIRM state is to ensure that the user wants to communicate through the gesture-based interface (i.e. to avoid accidental triggering of the tracking). The CONFIRM state is maintained for a minimum time interval, T_{min} . There is also an upper limitation, T_{max} , of the time spent in the CONFIRM state, in order to allow the system to return to the SEARCH state if the initial detection of the target is not confirmed. The user is aware that he must keep the hand in the required position for a short time interval (T_{min}) in order to trigger the tracker, and therefore I found reasonable to impose a value of T_{max} of approximately $2 \times T_{min}$.

While the system is in the CONFIRM state, the user should maintain the hand in the required position. In this state, for each frame, a decision about the hand presence is made. Two counters are updated in every frame and help to decide when to leave the CONFIRM state:

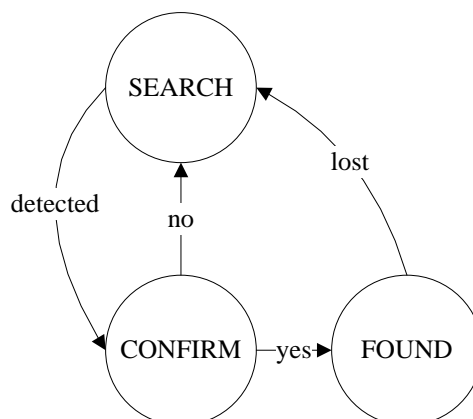


Fig. 4.4. State machine diagram.

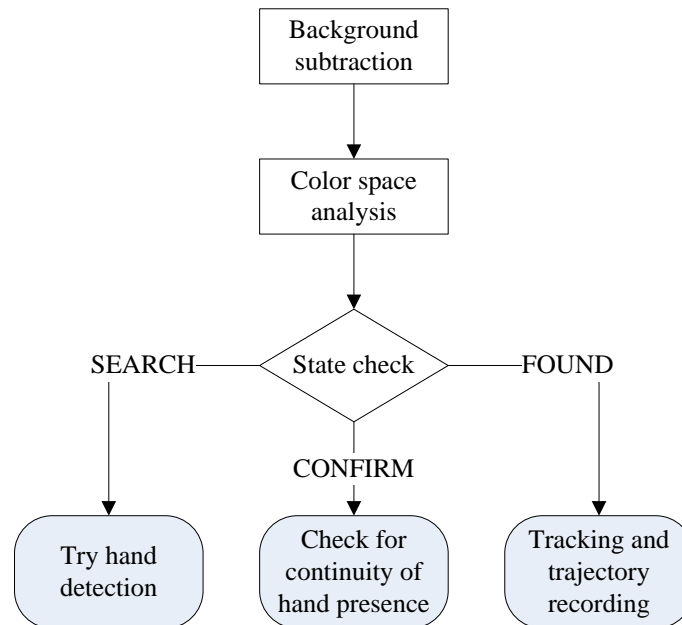


Fig. 4.5. Flowchart of frame processing for hand detection.

- a time/frame counter – counts the time (or the number of frames) elapsed since the beginning of the current CONFIRM state – and
- a hand detection success counter – a measure of hand detection rate.

The hand detection counter starts at 1 and is incremented with every frame in which the hand is detected. For every frame in which the hand is not detected the counter is decremented, but the decrement operation is limited to 0 (i.e. no decrementing takes place when the counter value is 0).

While the time counter is between T_{min} and T_{max} , the system may try to advance to the third state. At any moment within this time interval, if the hand detection success counter exceeds a specific threshold (approximately 70% of the number of frames processed during T_{min}), the tracker is initialized at the current location of the hand and the system advances to the 3rd state, FOUND. If the hand detection success counter does not reach the required threshold before T_{max} elapses, the tracker is not initialized and the system returns to the SEARCH state.

The FOUND state corresponds to the basic tracking operations, which are dealt with in the next sections. The system remains in this state as long as the target is not declared lost by the tracking algorithm. The target is assumed lost only if it is not detected for a relatively long time interval. When the target is considered lost, the system returns to the SEARCH state and the initialization procedure restarts.

4.2.4. Implementation details

The proposed method was implemented as part of a dynamic hand gesture recognition system. The algorithm was used for the initialization of a CAMSHIFT based hand tracker [6] and of a finger tracker, respectively. The application was developed using Microsoft Visual C++. In the implementation of the application, OpenCV library functions were used for various tasks like video capture (from

camera or from previously recorded *.avi files), background subtraction, color space conversions, pixel-wise operations etc. The video sequences were acquired using a common webcam at 640×480 resolution and approximately 15 fps.

The background subtraction was implemented using the codebook based method available in OpenCV.

The thresholds in the HSV color space, used to identify the reliable hue skin pixels and the non-reliable hue pixels, were set based on experiments. OpenCV uses 8 bits to represent each of the H, S and V components of a pixel. S and V each cover the full range available on 8 bits, [0 – 255]. H (hue), which is defined as an angle, should range from 0 to 360. In order to fit the 8 bit representation, in OpenCV, all hue values are divided by 2 and therefore the range is reduced to [0 – 180], as shown in Fig. 4.6.

Reliable hue pixels have saturation above 30, and value between 40 and 245. I determined empirically, by calculating hue histograms for manually selected skin colored areas, that the appropriate range for hue was [170 – 180] and [0 – 50]. It can be noticed in Fig. 4.6 that the two intervals are actually contiguous, due to the circular definition of the hue (180 and 0 represent the same color).

During the SEARCH and CONFIRM states, a hand contour is displayed on a monitor, in order to guide the user on the required hand pose and location as shown in Fig. 4.3. A rectangular binary mask is applied at the location of the displayed

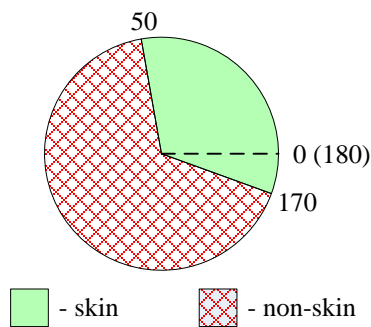


Fig. 4.6. Circular representation of the 8-bit hue.

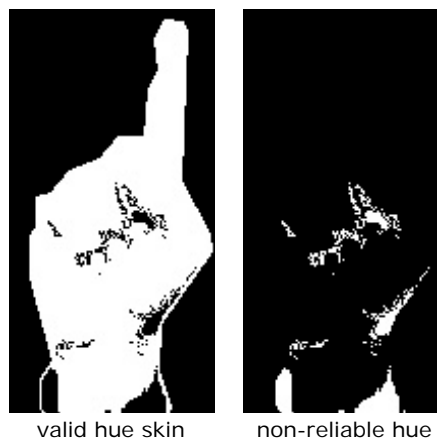


Fig. 4.7. Binary images after color space analysis in the hand mask region.

hand contour, in order to verify the shape/pose and location criteria. The two binary images obtained after the color space processing in the area of the rectangular hand mask, corresponding to the image in Fig. 4.3, are presented in Fig. 4.7.

A global threshold of 60% is used for matching pixels in the rectangular region, where the hand mask is applied. Some areas within the hand mask are considered critical and, therefore, tighter matching thresholds, of 70 - 85%, are applied separately to each of these areas. Fig. 4.8 presents the matching result image (white pixels indicate a match) and the 5 critical rectangular areas where the tighter thresholds are applied. White pixels indicate matching and black pixels indicate non-matching. In the example in Fig. 4.8, regions 1 and 4 have 100% matched pixels, regions 3 and 5 have 99% and region 2 has 93%. The tightest thresholds, of 85% are used for regions 1, 4, 3 and 5, while for region 2 a 70% threshold is used.

Region 1 should virtually contain 100% non-skin pixels, while region 4 should contain 100% skin pixels, regardless of the proportionality between hand/finger dimensions.

Regions 3 and 5 should contain non-skin pixels and they are treated together by applying an overall matching percentage threshold of 85%. The two regions are treated together in order to allow a more comfortable initialization procedure. Sometimes, one of them may have a lower matching percentage while the other virtually has 100% matching. Such an imbalance between the two regions may appear due to hand tilt and/or left/right position shift.

In region 2, a 70% threshold is applied. This region should contain skin pixels. The lower threshold used for this region is due to the fact that the matching percentage in this region is heavily influenced by two factors:

- the thickness of the index finger (the matching percentage lowers for users with thin fingers) and
- the possible tilt or position shift of the index finger with respect to the ideal position indicated by the guiding hand contour.

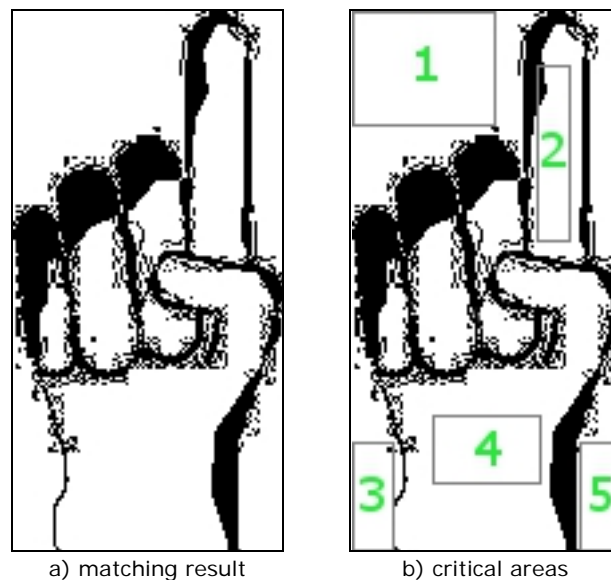


Fig. 4.8. Matching pixels in the rectangular hand mask region.

The global threshold is more relaxed because, while the hand mask is unique, different users have different hand/finger dimensions proportionalities.

In my experiments, the time limits used for the CONFIRM state were $T_{min} = 1s$ (15 frames) and $T_{max} = 2s$ (30 frames).

4.3. Finger detection

The problem of tracking the finger/hand is more complex than the one of tracking initialization. Some of the features used for initialization can also be used as low-level features for the tracking algorithm. The object to track (finger/hand) is a foreground object, has a relatively uniform color – human skin color, has a specific shape and has geometrical proportionality between width and height. Background subtraction and skin color segmentation can be used as preprocessing steps applied to the input video sequence, in order to extract low level features: skin color and foreground objects.

Taken alone, neither the skin color nor the foreground segmentation is able to offer reliable support for a robust tracking algorithm. For each of them, it is possible to obtain good true positive detection rates at the cost of high false positive rates. For single feature based trackers, this situation often leads to target loss due to confusion with other neighboring objects. Nevertheless, by integrating these low-level features with other geometrical features and by adding some reasoning in the tracking algorithm, a robust tracking solution may emerge. The high true positive rates can be exploited in order not to miss the presence of the real target, while the geometrical features and the reasoning can be used in the higher processing layers of the tracking algorithm, in order to significantly reduce the rate of false positives and correctly identify the target.

In the papers [3] and [4], I proposed a finger tracking solution which uses multiple low-level features and a multilayer processing approach, in order to achieve tracking robustness and real-time operation. The principle of the proposed algorithm is somehow similar to that of “tracking by detection”, which is very popular in algorithms for tracking multiple targets simultaneously. Tracking algorithms which use the “tracking by detection” principle perform in every frame a detection of all objects which have characteristics similar to the target and then try to establish correspondences between the objects detected in the current frame and the targets detected in previous frame(s). The envisaged range of applications for the proposed algorithm – user interfaces based on dynamic gestures or pointing – requires the tracking of a single target. A reduced area of the frame where the target is likely to be found can be selected, based on a prediction of the target position. This leads to a significant reduction in the amount of data to process, since the multiple layers of processing in the tracking algorithm are applied only to the limited region of interest, not to the entire frame.

In the proposed finger tracking algorithm, background subtraction and skin color segmentation are used as preprocessing steps for extracting the low-level features. After the preprocessing steps, the remaining significant data are scanned horizontally or vertically and represented as line strips as shown in Fig. 4.9. For each line strip, only the length and the midpoint’s coordinates need to be retained, leading to a very compact representation of the data used for finger detection. The line strips’ positions and lengths are easy to extract. Nevertheless, the line strips are thoughtfully chosen features, as they are able to capture the relevant information needed to detect the finger. All line strips that belong to the same finger have similar lengths and their centers must fit a curve that can be approximated by a line

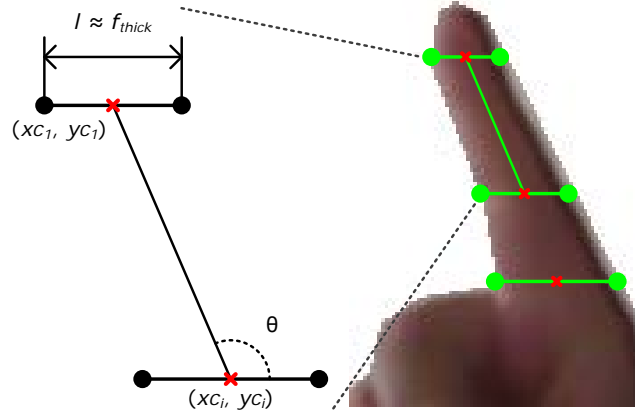


Fig. 4.9. Line strip features used for finger detection.

segment. The (ρ, θ) space is used to define the line. A similar approach for extracting line strips used further as features for finger tracking was used in paper [2].

The proposed finger detection algorithm uses a multi-layer approach, as shown in Fig. 4.10. The lower layers are responsible with the finger detection, while the topmost layer is recording the trajectory and guiding the tracking process. The information obtained at the topmost layer is passed to the trajectory processing and gesture recognition layers and is also used for guiding the tracker.

At the *basic layer*, a search area – selected by using data from the tracker guiding layer – is scanned for line strips. The result of this step is an array of line

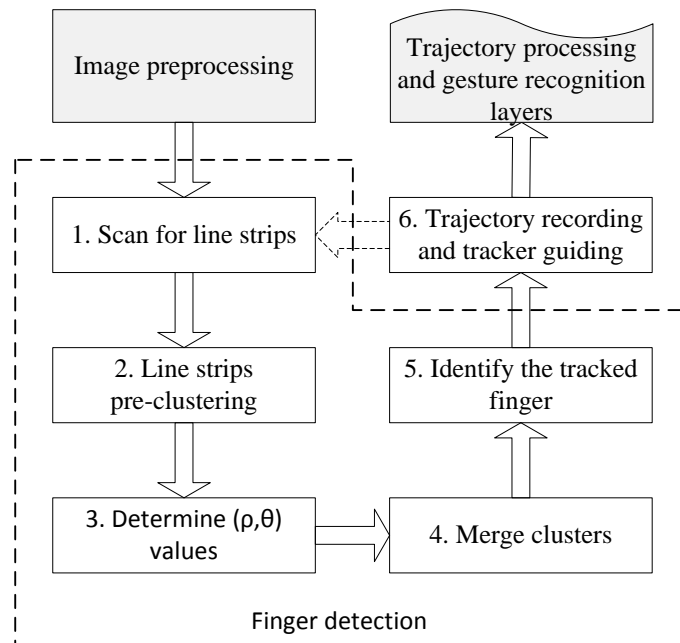


Fig. 4.10. The layers of the tracking algorithm.

strips identifiers (mid-point coordinates and length). Optionally, the coordinates of the line strips' ends may be recorded, too, in order to avoid recalculating them in the next steps. As the next steps of processing are similar for horizontal line strips and vertical line strips, in the following I will refer only to the processing of horizontal line strips in order to simplify the description of the algorithm.

At the next layer, the line strips are clustered based on spatial information, then at the third layer a pair of (ρ, θ) values are determined for each line strip by matching the current strip with strips situated at various distances within the same cluster. The fourth layer performs a cluster analysis based on the (ρ, θ) parameters and merges smaller clusters with similar (ρ, θ) values. The fifth layer identifies the cluster that matches the tracked finger and calculates the new position and dimensions of the finger. The topmost layer of the tracker records the trajectory and parameters of the finger movement.

At the *second layer*, the recorded line strips are clustered based on their positions. The goal of this step is to identify vertically contiguous groups of line strips. The clustering starts with the first element of the array (the topmost left line strip), which is assigned to the first cluster. Then, if available, a new line strip is added to the cluster from each subsequent line, given that its horizontal position is not significantly different from that of the previous line strip in the cluster. The limit imposed for the maximum allowed horizontal deviation between 2 consecutive line strips can be set between $[0.25 \div 0.5]$ of the expected finger thickness. Although the line strips that belong to a finger are virtually contiguous, in practical situations some small gaps may appear between the recorded strips, due to image imperfections like motion blur or non-ideal lighting conditions. Therefore, limited small gaps (e.g. up to 5 lines) are allowed within a cluster. If a larger gap is encountered, the cluster is closed and if there are still unclassified line strips, a new cluster is instantiated starting with the first unclassified line strip in the array.

After all the line strips in the array are classified, isolated clusters containing only a few line strokes are removed from further processing, as they do not contain relevant information.

At the *third layer*, within each cluster, for each of the line strips, a (ρ, θ) pair of parameters is calculated. The centers of the line strips belonging to a finger must fit a line segment, which can be identified using the θ (the angle of the line segment

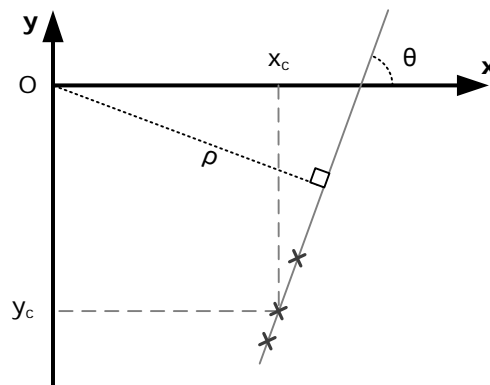


Fig. 4.11. The ρ, θ parameters of the line segment generated by the centers of the line strips belonging to a finger.

with the horizontal axis) and ρ (the distance from the origin of the coordinates system to the line strip) parameters. Fig. 4.11 presents an example of line segment containing the centers of line strips, and its ρ and θ parameters.

Within each significant cluster, for each line strip, multiple values of the angle parameter θ_i are determined by searching pair strips at increasing distances, first toward one end of the cluster and then toward the other. Fig. 4.12 illustrates the process of searching for pair elements for a given line strip, in order to calculate the multiple θ_i values that will be temporarily associated with it.

Each θ_i value is determined as the angle between the line that connects the mid-points of the two strips (the current line strip and the found pair) and the horizontal axis. The search for the first pair line strip starts at a minimum distance d

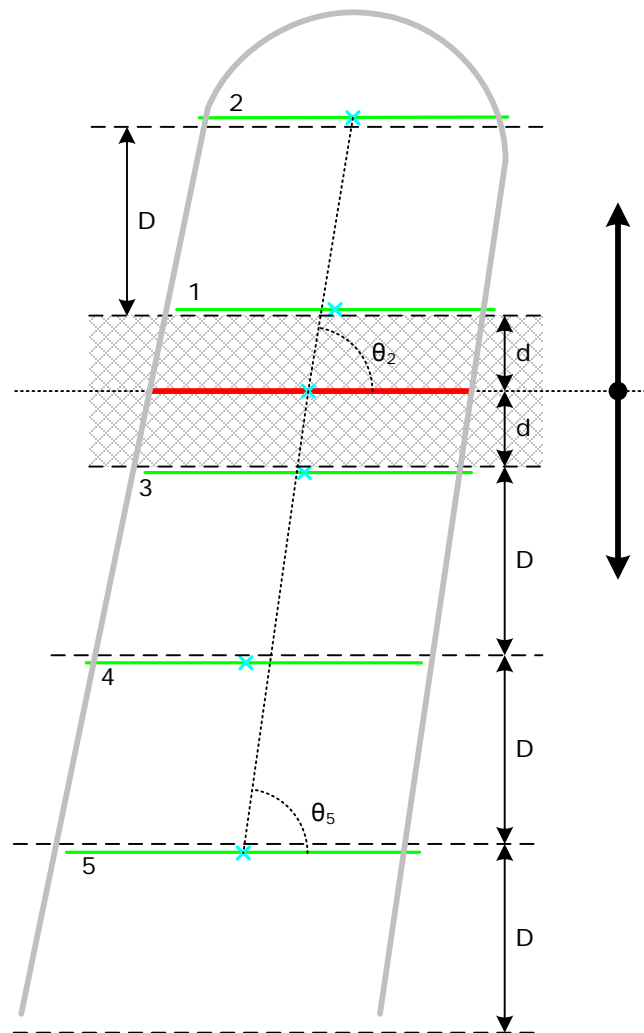


Fig. 4.12. The pair line strips used to determine the θ value associated to an analyzed strip.

from the analyzed strip, as pair strips that are too close to the analyzed strip may produce noisy angles. After a pair strip is found, the next pair will be searched at a distance incremented with D ($D > d$) from the previous searching start point and the process repeats, using the D distance increment, until the cluster end is reached. The value of d must be chosen so that it will allow finding at least one pair for all the line strips in the smallest valid cluster. The other distance increment, D , is greater than d and is chosen in order to allow the detection of 5-7 line strip pairs within a cluster with the expected size of the finger. Each θ_i value is stored together with the distance between the analyzed strip and the pair.

A weighted average of the θ_i values is calculated, in order to determine the θ value to be assigned to the analyzed line strip within the current cluster:

$$\theta = \frac{\sum_i w_i \cdot \theta_i}{\sum_i w_i}. \quad (4.4)$$

The weights w_i are proportional with the distance between the analyzed strip and the pair strip, as the influence of the noise caused by slightly misaligned strips diminishes with the increase of distance.

After calculating the θ value associated to a line strip, the corresponding value of ρ is calculated for a line which makes the angle θ with the horizontal axis and crosses the analyzed strip in the middle, as shown in Fig. 4.11, using the formula:

$$\rho = \frac{(y_c - x_c \cdot \text{tg}\theta)}{\sqrt{1 + \text{tg}^2\theta}} \quad (4.5)$$

After computing the θ and ρ values for each line strip, within each significant cluster, global values for θ and ρ respectively are calculated after a 2 steps removal of outliers. First, the median value of θ is calculated within the cluster, then, up to 10% of the line strips with extreme values of θ are removed from further analysis, if their θ value differs with more than 15° from the median value. A 2D histogram based on θ and ρ is built for the remaining valid line strips and the bin with the highest element count is selected. Around this bin, a search for other bins with high element counts is conducted and all neighboring bins that exceed a threshold value are selected as valid. Then line strips belonging to non-valid bins are removed from further processing. Using the remaining valid line strips, the cluster representative values of θ and ρ are calculated as mean values of the individual parameters of the line strips.

At the *fourth layer*, the clusters are checked for the presence of a hand at one of the ends, as a finger normally appears only as an extension of a hand. The clusters that are attached to a hand-like shape are marked as finger candidates. Each cluster which is not marked as a finger candidate after this process is checked for compatibility with finger candidates with respect to the θ and ρ values. If any match is found, the cluster is merged to the respective finger candidate, given that the total length of the new cluster does not exceed 150% of the expected finger length. The clusters that cannot be merged to finger candidates are removed from further processing.

The search for the presence of a hand starts at the base of the finger candidate and tries to find an area delimited by a polygonal contour which contains a large majority of skin colored, foreground pixels. Fig. 4.13 presents the procedure

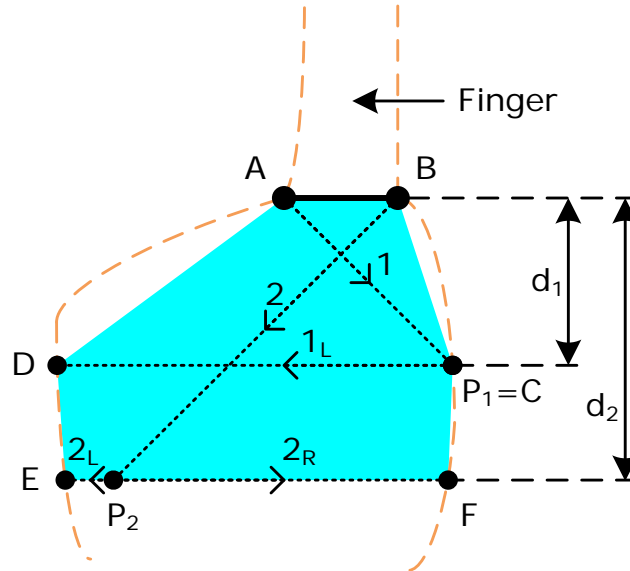


Fig. 4.13. The search for attached hand procedure.

for identifying the polygon's vertices. The polygon is usually a hexagon, but in the particular case when P_1 and P_2 are on the same horizontal line, it is reduced to a trapezoid. The ends of the finger-base line strip are the first two vertices, A and B. The other vertices are determined using two symmetrical search procedures, starting from A and B, respectively.

Starting from A, the left extremity of the terminal line strip, a diagonal search down at -45° for contiguous hand pixels is conducted – step 1 in Fig. 4.13. A similar search is conducted starting from B, the right extremity of the terminal line strip, also downwards, but at an angle of -135° with the horizontal axis – step 2 in Fig. 4.13. During the search, small gaps are allowed in order to deal with possible spots caused by imperfections within the acquired images (noise, motion blur, non-uniform lighting etc.). The search stops either when the contiguousness criterion is no longer met (the hand edge is detected) or when a limit value for the total horizontal displacement from the initial position equal to 3 times the finger thickness is achieved. The end point of the search in step 1 is P_1 and the end point of the search in step 2 is P_2 . The 4 remaining vertices are determined as the hand edges on the horizontal lines containing P_1 and P_2 , respectively.

If any of the searches in steps 1 or 2 ends due to hand edge encountering, the end point of the respective search is also a vertex of the polygon. This situation is illustrated in Fig. 4.13 for step 1, where the search end point P_1 and the C vertex of the polygon have the same coordinates. If the search in step 1 or 2 ends due to exceeding the threshold horizontal displacement, then from the corresponding P_i ($i=1,2$) end point a horizontal search in the same direction (right for $i=1$ and left for $i=2$) is conducted for a maximum displacement of one finger thickness, in order to try to reach the hand edge. The searched vertex is located at the hand edge, if encountered, or at the end point of the horizontal displacement limit. This situation is illustrated in Fig. 4.13 for step 2, which is followed by step 2L in order to locate the E vertex.

The last two vertices are located by conducting horizontal searches for the hand edge starting from P_i , in the opposite direction to the one used at step i . As with the previous cases, a maximum horizontal displacement limit is imposed, in this case with a value of 4 times the finger thickness. The end points of these searches will represent the last two vertices of the polygon. Steps 1L and 2R in Fig. 4.13 lead to the localization of the last two vertices, D and F respectively. The maximum displacement limits during the search for vertices are imposed in order to allow reasonably good hand localization, when the hand is in the neighborhood of other skin colored areas in the image (e.g. other hands or faces).

A hand is considered detected if the polygon constructed using the above described procedure simultaneously fulfills multiple criteria, referring to the total surface within the polygonal contour, the percentage of skin pixels inside the polygonal contour and geometrical considerations.

The first criterion requires that the total surface inside the polygonal contour exceeds a given threshold, determined as a function of the size of the finger candidate. The part of the hand contained within the polygon has in most situations an area larger than the finger. Therefore, the finger area, $S_f = f_l \cdot f_{thick}$, is a good choice for the area threshold. Alternatively, an area, $S_q = (2 \div 3) \cdot f_{thick}^2$, proportional to the area of a square with the side length equal to the finger thickness, can be used as threshold.

The second criterion also consists in exceeding a given threshold, in this case for the percentage of skin colored foreground pixels within the polygonal contour. Under ideal conditions, all pixels within the polygonal contour should belong to the hand, as shown in Fig. 4.13. Practically, due to limitations of the image acquisition systems, a small percentage of the pixels within the polygonal contour may be classified as non-skin colored and/or as background pixels. Therefore, a value between 75% ÷ 85% for the percentage threshold must be chosen, in order to deal with the imperfections in the acquired images.

The third type of criteria that need to be fulfilled for hand detection, concerns the geometrical proportions within the polygon. Two proportionality criteria are evaluated: one in the horizontal direction (line segments' lengths) and the other one in vertical direction (distance between line segments). The line segments involved in these evaluations are AB, CD and EF, which are all parallel to the horizontal axis. Some limits are imposed on the proportions between the length of line segments and on the distance between them. As the procedure of localization of the polygon's vertices imposes in both horizontal and vertical dimensions upper limits relative to the finger thickness, which is approximately equal to the length of AB line segment, only lower limits need to be imposed further. The horizontal proportionality criterion imposes a lower limit on the ratio between the maximum length of CD and EF line segments and the finger thickness:

$$\frac{\max(L_{CD}, L_{EF})}{f_{thick}} > 2. \quad (4.6)$$

The proportionality in the vertical dimension imposes a lower limit on the ratio between the distance between the horizontal sides of the polygon and the finger thickness:

$$\frac{\max(d_1, d_2)}{f_{thick}} > 1.5. \quad (4.7)$$

4.4. Target identification

The *fifth layer* is responsible with the identification of the tracked finger amongst the finger candidates selected at the previous layer. The finger candidates are classified based on a total score, s_t , which is calculated as a sum of the two scores, s_p and s_l :

$$s_t = s_p + s_l. \quad (4.8)$$

The s_p score is a measure of the cluster's quality as a finger candidate, with respect to the distance between its position and the predicted position of the finger. The s_l score evaluates the cluster's quality as a finger candidate from the size point of view.

As the predicted position is obtained based on the motion information in previous frames, more importance is given to the position criterion in order to favor a candidate that is close to the predicted position (if present) with respect to other candidates that do not fit the predicted motion model.

$$s_p = S_p - \min(S_p, \lfloor c_p \cdot \Delta p_r \rfloor), \quad \Delta p_r = \frac{\Delta p}{f_{thick}}, \quad (4.9)$$

$$s_l = S_l - \min(S_l, \lfloor c_l \cdot \Delta l_r \rfloor), \quad \Delta l_r = \frac{|f_l - \hat{f}_l|}{f_{thick}}. \quad (4.10)$$

The s_p score, calculated using equation (4.9), decreases with a term that is proportional to the relative distance between the actual and the estimated position (the ratio between the absolute distance, Δp , and the finger's thickness). This leads to a maximum score, S_p , in a limited area around the predicted position and gradually smaller scores as the relative distance between the candidate and the predicted position increases.

The s_l score, calculated using equation (4.10), has a maximum value, S_l , for small finger length variations, and lower values for the candidates with important length changes since the last frame. Similar to the s_p score, the s_l decreases with a term that is proportional to the relative length variation, Δl_r , which represents the absolute difference between the actual length and the predicted length of the finger reported to the finger's thickness.

The tracked finger is identified as the finger candidate with the highest total score, s_t , given that it exceeds a minimum threshold. Practical experiments have shown that a good value for the threshold is about half of the maximum possible value of the total score.

The identified finger is represented as an ellipse defined by a center, a minor axis, a major axis and a rotation angle. The ellipse is centered at the midpoint of the line segment that connects the centers of the first and the last line strips of the cluster. The thickness of the detected finger is calculated as the median of the lengths of valid line strips within the cluster, and will represent the ellipse's minor axis. The length of the detected finger is proportional to L , the length of the segment connecting the mid-points of the first and last line strips of the cluster:

$$f_l = a \cdot L. \quad (4.11)$$

The proportionality factor, a , in equation (4.11) is necessary, as the valid line strips do not cover the extremity of the fingertip due to size limitations (line

strips at the extremity of the fingertip are shorter than the minimal limit for line strip's length). Practical experiments have shown good results with a values in the range [1.1 – 1.2]. The obtained finger length represents the major axis of the finger ellipse. The rotation angle of the ellipse is the θ value of the cluster.

4.5. Tracker guiding and trajectory recording

The 5 layers described in the previous sections perform the low-level tracking tasks, leading to the identification and localization of the finger in the current frame. The topmost layer of the tracking algorithm is using the data provided by the lower layers in order to perform two tasks:

- finger trajectory recording, providing the data necessary to the gesture recognition layers and
- tracker guiding.

The data necessary to the gesture recognition layers consist of the finger position, orientation and optionally size, in every frame. The data required for tracker guidance consist of a predicted position and size of the finger in the next frame, based on which a search window is selected, in order to apply the finger detection steps.

The finger positions transmitted to the gesture processing layers are exactly the positions provided by the finger detection layers. The finger size is mainly required for tracker guidance, but, depending on the application, it may also be transmitted to the gesture recognition layers (e.g. when the finger length is used in the gesture recognition process). The value of the finger length or thickness in the current frame, v_{new} , is calculated using an adaptive weighted average between the previous value, v_{old} , and the measured value provided by the finger detection layers, v_m .

$$v_{new} = (1-a) \cdot v_{old} + a \cdot v_m, \quad 0 \leq a \leq 1. \quad (4.12)$$

The adaptive weighting is necessary in order to allow the system to follow the measured finger size quicker when the changes in finger size are consistent and slower when the changes are not consistent. The changes in finger size are assumed to be consistent only when both finger length and thickness have the same sign of variation and the trend is maintained during multiple consecutive frames. The trend is assumed to be consistent over multiple frames if the sign of the size variation with respect to the previous frame is the same for at least 2 of the last 3 frames. For this purpose, the variation sign, sv , is calculated each frame for both finger thickness and finger length values.

$$sv = \text{sign}(v_m - v_{old}). \quad (4.13)$$

The variation signs calculated using (4.13) for the current frame and for the previous 3 frames are used in the evaluation of the variation trend consistency.

For slowly following the measured finger size, when no consistent size variation is detected, a small value of a is used in equation (4.12), while for quickly following the variation, a larger value is necessary. I determined experimentally that good results are obtained using a values around 0.3 for slowly following the size changes and 0.7 for quickly updating the finger size.

The prediction of the future position of the finger, together with the selection of a reduced size search window around the predicted position, can help in reducing the amount of data (line strips) to process. By focusing the search around

an expected location in the image, the image areas that are too far from the predicted location of the finger are excluded from processing. The search window size must exceed the area covered by the finger enough to allow for relatively small variations of finger speed and/or movement direction. My experiments revealed that 10 times the finger thickness ($\pm 5 \times \text{fthick}$ around the expected position) and 4 times the finger length ($\pm 2 \times \text{fl}$ around the expected position) are appropriate values for the width and height of the search window.

A first order prediction is used in order to obtain the expected finger position in the current frame (frame n). The horizontal and vertical components of the motion vector are calculated using the finger positions in the previous 2 frames:

$$\begin{aligned}\Delta x_{n-1} &= x_{n-1} - x_{n-2}, \\ \Delta y_{n-1} &= y_{n-1} - y_{n-2}.\end{aligned}\quad (4.14)$$

Using the motion vector components and the time elapsed between the 2 frames, the horizontal and vertical components of the finger's speed are determined:

$$\begin{aligned}v_{x_{n-1}} &= \frac{\Delta x_{n-1}}{\Delta t_{n-1}}, \\ v_{y_{n-1}} &= \frac{\Delta y_{n-1}}{\Delta t_{n-1}}.\end{aligned}\quad (4.15)$$

The time elapsed between the previous frame and the current frame, Δt_n , is used together with the finger's speed vector components to determine the expected finger position in the current frame based on the finger position in the previous frame:

$$\begin{aligned}\hat{x}_n &= x_{n-1} + v_{x_{n-1}} \cdot \Delta t_n, \\ \hat{y}_n &= y_{n-1} + v_{y_{n-1}} \cdot \Delta t_n.\end{aligned}\quad (4.16)$$

In most situations, the above described prediction method leads to the selection of a search area which contains the tracked finger, allowing the finger detection layers to precisely locate the finger in the current frame. Nevertheless, isolated situations may occur when the finger – although detectable in the current frame – is not (entirely) contained within the initially predicted search area and therefore it cannot be located within this search window by the finger detection layers. Such exceptional situations may be caused by unexpected sudden changes in the finger's movement direction and/or speed, combined with long time intervals between consecutive frames (slow frame rate).

If the target is not detected within the initial search window, new searches are attempted in the neighborhoods of the initially predicted position. The search window is shifted in up to 8 possible different directions. The centers of the 8 new search windows are located at the 4 corners and the 4 midpoints of the sides of the initial search window. The order in which the 8 search windows are tested is established based on a priority list, which takes into account the main reasons which can lead to not finding the target within the initial search window. The first two directions to be tested are the direction which is nearest to the finger motion vector in the previous two frames and its opposite direction, then their neighboring directions follow and, finally, the last tested directions are those orthogonal to the first two. The priority between the direction which assumes the maintaining of the movement direction and the one which assumes a 180° turn is established based on

the records of finger motion speed in the last frames. An increase in movement speed indicates a higher likelihood that the finger will maintain the movement direction, while a decrease of the speed indicates a possible intention of the user to change the movement direction.

The relatively reduced size of the search window reduces the data to process and helps keeping the tracker focused on the real target in situations when other similar objects are present in the image. Repeating the search procedure, when the detection within the initial window fails, does not significantly increase the processing time. In most cases, a search in which no finger is detected is abandoned in the early stages of the finger detection algorithm, due to a reduced number of valid line strips to process.

4.6. Performance evaluation

4.6.1. Tracking initialization

The proposed tracker initialization method was tested with different backgrounds, both with daylight and artificial lighting. A number of 25 tests were performed by 3 users.

The histograms of the global matching percentages for 6 tracker initializations (1 with daylight and 1 with artificial lighting for each of the 3 users) are presented in Fig. 4.14. In each case, the frames taken into account begin with the frame in which the hand is detected for the first time and end with the frame in which the tracker is initialized.

It can be observed that in a single case – artificial light 2 – percentages below 60% are obtained. The 6 matching percentages below the 60% threshold correspond to frames in which the user's hand was not aligned correctly with the guiding hand-shaped contour. It can also be observed that, in the other 5 cases, all the matching percentages are higher than 65%.

In 5 of the cases presented in Fig. 4.14, the CONFIRM state ends after the minimum time interval, while for the other case – artificial light 2 – 2 additional

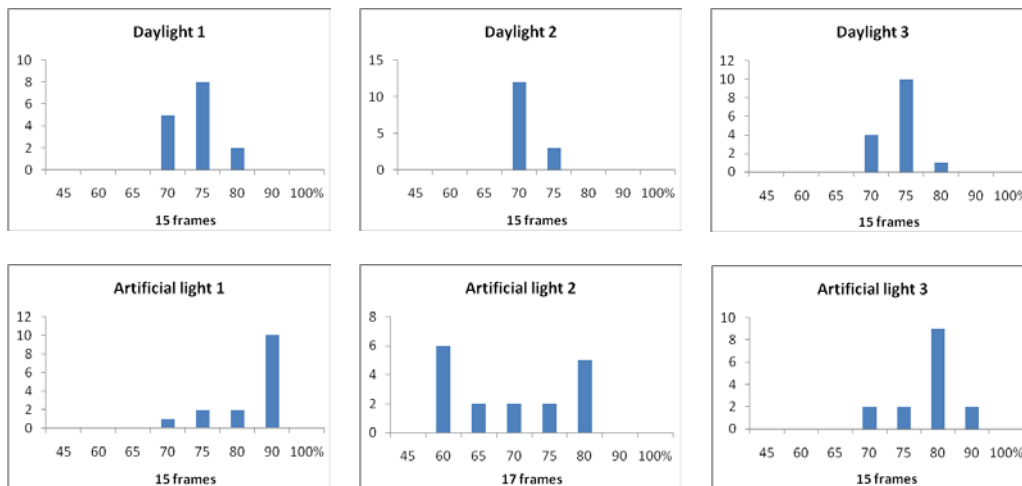


Fig. 4.14. Histograms of global matching percentages in the rectangular hand mask region.

Table 4.1. Summary of the initialization results.

	Criteria passed [%]	Min %	Max %	Mean %
Global	92	48	82	73
Region 1	96	76	100	97
Region 2	87	0	100	88
Region 4	91	39	100	94
Regions 3,5	95	80	99	96
All	78			

frames are necessary before advancing to the FOUND state and initializing the tracker.

Tracker initialization experiments were analyzed for the 25 cases (15 with daylight and 10 with artificial lighting) and Table 4.1 summarizes the results obtained from the point of view of matching percentages and thresholds fitting.

Considering each of the 5 threshold based detection criteria, detection was successful in more than 87% of all frames processed in the CONFIRM state. Actually, a success rate below 90% was obtained only for the region 2 criterion. The hand is considered detected in a frame only if all 5 criteria are met, and this happened for 78% of the frames analyzed. The last three columns of the table present the minimum, the maximum and the mean matching percentages corresponding to each of the 5 criteria. The mean was calculated by removing 3% of the worse and 3% of the best matching percentages. Low matching percentages correspond to frames in which the user's hand did not correctly fit the indicated shape. The results obtained indicate that the chosen combination of criteria allows the system to correctly identify the frames in which the user's hand is present at the required location.

The histogram of the number of frames spent by the system in the CONFIRM state is presented in Fig. 4.15. In 19 of the 25 tests, the tracker initialization occurred after the minimum time interval (15 frames – 1s at 15 fps).

Only in 2 cases, in which the user slightly moved the hand around the guiding contour in order to test the limits of the detection capacity, more than 20 frames were necessary to accomplish the requirements for tracker initialization. The measured time intervals, considered in Fig. 4.15, do not take into account the time necessary for the user to fit the hand correctly to the guiding shape. The average time needed to fit the hand to the guiding shape was below 3 s. This illustrates that the proposed method allows the user to easily initialize the tracker.

Additionally, the system was tested for resistance to false triggering. For this purpose, 3 types of tests were performed:

- random movements of the hand were performed around the initialization area,
- human subjects moving around the initialization area and
- global lighting change (most parts of the image appeared as foreground).

In the first case, when random hand movements were performed in the initialization area, the system advanced occasionally from the SEARCH to the CONFIRM state, but no false initialization occurred, as the conditions imposed in the CONFIRM state to allow the tracker initialization were not met.

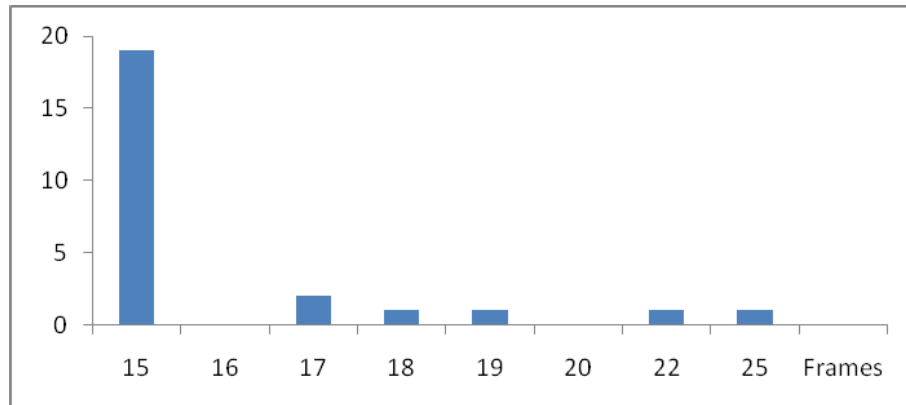


Fig. 4.15. Histogram of the number of frames in the CONFIRM state for the 25 experiments.

For the second test scenario, when human subjects moved around the initialization area, no false hand detection occurred and the system remained in the SEARCH state.

The resistance to false initialization due to global lighting changes was tested using 5 different backgrounds, both for increasing and decreasing lighting. During the tests performed, no false hand detection occurred and the system remained in the SEARCH state. When skin-like background was used, it was observed that, due to the lighting change, some areas of the background appeared as foreground and therefore 2 criteria (foreground object and color) were met for these areas, but during the tests performed no such area happened to take the shape of the hand required for initialization. The probability for such an area to take the required hand shape and size, at the required location in the image, in order to trigger a false tracker initialization, is extremely low. Therefore, we can consider that lighting changes in the scene are unlikely to cause false initializations, regardless of the background used.

The tests for resistance to false triggering, combined with the results of the 25 tests for tracker initialization, indicate the reliability of the proposed method.

The tracking initialization method presented in section 4.2 proved its reliability during the initial tests and was later successfully used for the initialization of the hand/finger tracking algorithm proposed in [3] and [4] and presented in sections 4.3 – 4.5. The method is easy to use from the user's point of view. While the multiple conditions imposed for initialization need low computational resources, they are able to provide a quick initialization and to prevent false triggering. The multi-cue approach allows the proposed initialization method to operate correctly under very different lighting conditions, with different backgrounds, without the need to readjust the settings of the thresholds. The advantage of a safe start is obtained at the price of a reduced flexibility regarding the initial position of the hand, and restrictions regarding the hand color uniformity (i.e. the user may not wear gloves, have extremely dirty hands etc.).

The 4 detection criteria together with the time constraints imposed provide a user friendly initialization procedure. The time interval when the user must keep the hand in a given pose at a specific location is short enough in order not to be considered a drawback and it is long enough to significantly reduce the chances of false triggering.

The proposed method can be used with a large variety of hand/finger trackers, as it only identifies the time when the object to be tracked is present at the specified location, so that the tracker can start and no restrictions are imposed on the tracking algorithm.

4.6.2. Tracking

The proposed tracking algorithm was tested on 8 video sequences taken in different conditions. The test sequences were acquired both under natural (4 test sequences) and artificial lighting (4 test sequences) and include simple and complex background, hand movement at various speeds, movement speed variation, partial occlusion with other objects, hand pose changes, superposing with other foreground objects, including hands with extended index finger. The test sequences were acquired using both stand-alone and notebook-integrated webcams, at resolutions of 640×480 (6 sequences) and 960×720 (2 sequences), respectively. The proposed method provided good results in most of the challenging situations in which it was tested.

Tracking results were very good for all 8 test sequences. In the absence of disturbing factors, the tracking results were accurate, both for the fingertip position and finger parameters (size and angle), with no target loss and no visible localization errors.

In sequences which contain moderate motion blur, the tracker is able to follow the target thoroughly, due to the possibility to choose between using the centers or left or right ends within the clustering steps (layers 2-4), which allows to choose the feature that produces the smallest angle distortion within the finger candidate cluster. In normal situations, when no motion blur is present, the mid points of the line strips produce the best clustering results. When motion blur is present, the finger becomes blurry, showing a rough surface, especially on its rear side (with respect to the movement direction), while on the opposite (front) side it has a slightly smoother appearance. For such situations, the line strips' ends on the front side of the finger produce more compact clusters than the very noisy line

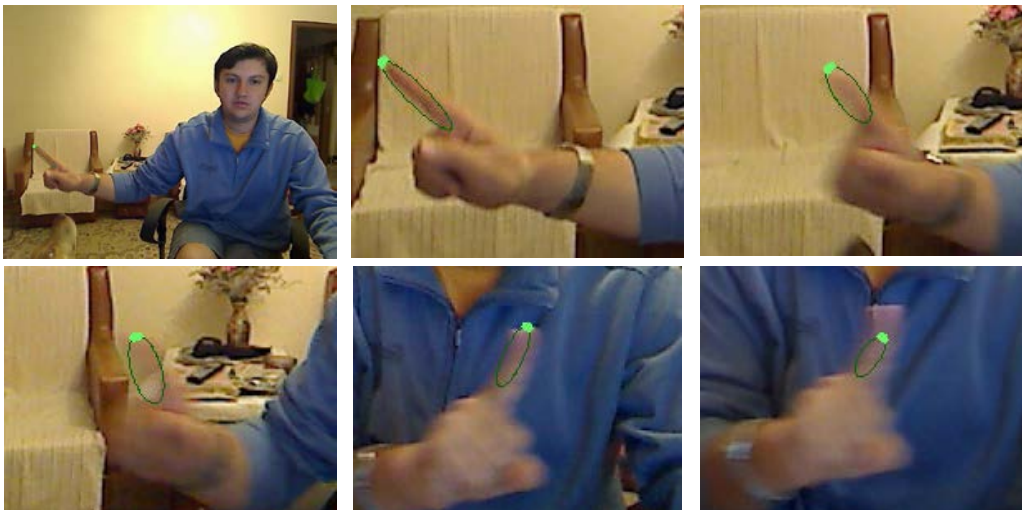


Fig. 4.16. Tracking under severe motion blur
– frames 163 (complete scene and hand detail), 177, 179, 224 and 226 of test sequence 2

strips' ends on the rear side or the mid points – for which the noise is an average between the two sides – increasing the probability to detect valid finger candidates. In Fig. 4.16, the scene in test sequence 2 is presented, together with details from frames with motion blur on skin-like colored background (frames 177 and 179) and on non-skin colored background (frames 224 and 226).

When strong motion blur is present, especially on skin-like colored background, the results of the preprocessing steps become unreliable for the line strip detection process and therefore the finger detection fails in such frames. Strong motion blur occurs at high movement speed of the finger, especially in low lighting conditions. Fortunately, the high motion speed usually occurs only for short time intervals and during straight movement strokes. Even if the target cannot be detected for a few frames, under the assumption of straight movement the use of a location prediction mechanism within the proposed algorithm allows to keep a good approximation of the finger position until the amount of motion blur reduces to a level that allows finger detection. In the test sequences used, 116 frames with significant motion blur were present in test sequences 1, 2 and 3. The tracker failed to detect the target in 3 of the 116 blurry frames. The frames in which the tracker failed to detect the finger contained an extreme level of motion blur on skin-like colored background. Although the finger was not detected within the 3 frames, the tracker was able to recover immediately after the motion blur returned within acceptable limits.

In another 29 blurry frames, on the non-skin colored background, the finger was detected with a smaller size, and/or with an angle deviation (e.g. frame 226 in Fig. 4.16). The size and angle errors can be explained by the color blending which occurs in the case of motion blur on non-skin colored background, which causes some blurred finger pixels to fall outside the color range used for skin segmentation.

Another challenging situation is to track the finger on skin-like colored background. The tests I conducted revealed good results, as long as the finger skin color differs from the background color enough (in order to allow the foreground/background segmentation algorithm to discriminate the finger from the background) and when motion blur was not too strong. The test sequences 1-4

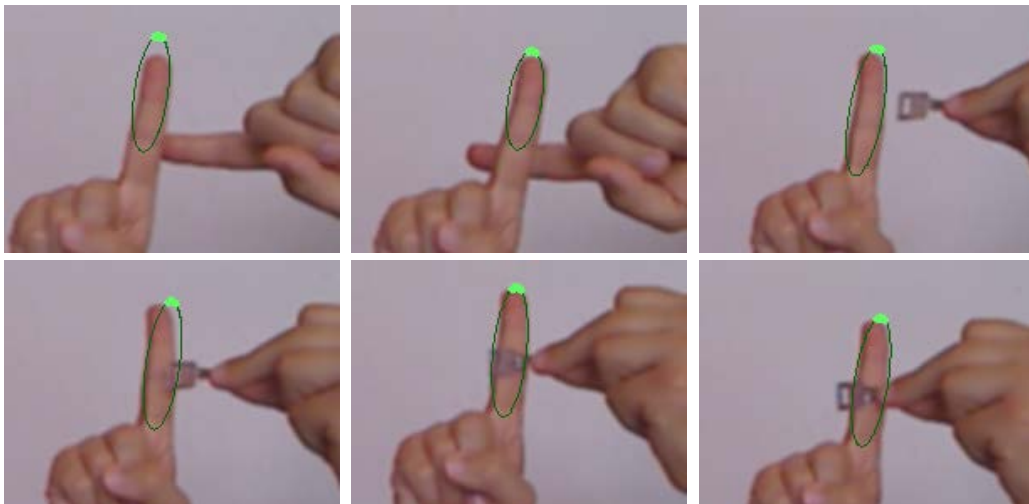


Fig. 4.17. Tracking results when the finger is partially occluded – frames 71, 74, 219, 222, 227 and 231 of test sequence 7

contained mainly skin-like colored background. In test sequences 1 and 2, artificial lighting was used, while in test sequences 3 and 4, only natural lighting was present. The tracking algorithm failed to detect the target on skin-like colored background only in the 3 frames with extreme motion blur mentioned above.

Partial occlusions are handled well by the proposed tracking algorithm, as long as the occlusion area is not covering a too large area of the finger. Tracking results in scenes with partial finger occlusion are presented in Fig. 4.17. When the occlusion occurs in the medial part of the finger (frames 222, 227 and 231), the clusters merging layer of the algorithm is able to detect the isolated parts of the finger and recognize they belong to the same finger, if they exceed the minimum cluster length imposed by the algorithm. When the occlusion occurs near the fingertip or near the finger base, the size of the detected finger decreases if the occlusion persists for more frames (frames 71 and 74). The detected finger size does not decrease instantly to the value corresponding to the finger part that is outside the occlusion area. The adaptive size updating procedure allows only a small size reduction during the initial frames of the occlusion (frame 71), but, if the occlusion persists (frame 74), the size will reduce faster after the threshold number of frames for fast following the size changes is exceeded.

The adaptive finger size updating procedure used within the proposed algorithm is useful, especially for handling scale changes in the video sequence. In the context of the envisaged application range of the proposed method, scale changes occur when the user moves the finger closer to or further apart from the fixed camera.

The tests' results revealed a good ability of the tracker to follow the scale changes with both size increase and decrease. Fig. 4.18 presents tracking results when the finger moves away from the camera in test sequence 3 (frames 22 – 49). Although the finger dimensions are halved within approximately 2 seconds, the adaptive size updating procedure allows the tracker to thoroughly follow the scale changes and to correctly identify the finger size and position. Similar results were also obtained when the finger size grew, even with fast motion in test sequence 4.

Accurate tracking results were also achieved when the target moved fast through the scene in various directions. Fig. 4.19 presents sample frames with tracking results under fast motion in test sequence 4, acquired at 960×720 resolution with natural light. The top row presents a fast movement in the horizontal direction between frames 64 – 72. The finger moves across more than half the frame width (approximately $15 \times f_{thick}$) within approximately 0.5 s (8 frames). The middle row presents a fast vertical movement which takes place between frames 79 – 82, covering almost half of the frame height (twice the finger length) within less than 0.25 s (3 frames). The bottom row illustrates a diagonal movement taking place between frames 110 – 113, with approximately the same parameters as for



Fig. 4.18. Finger moving away from the camera
– frames 22, 33 and 49 of test sequence 3.

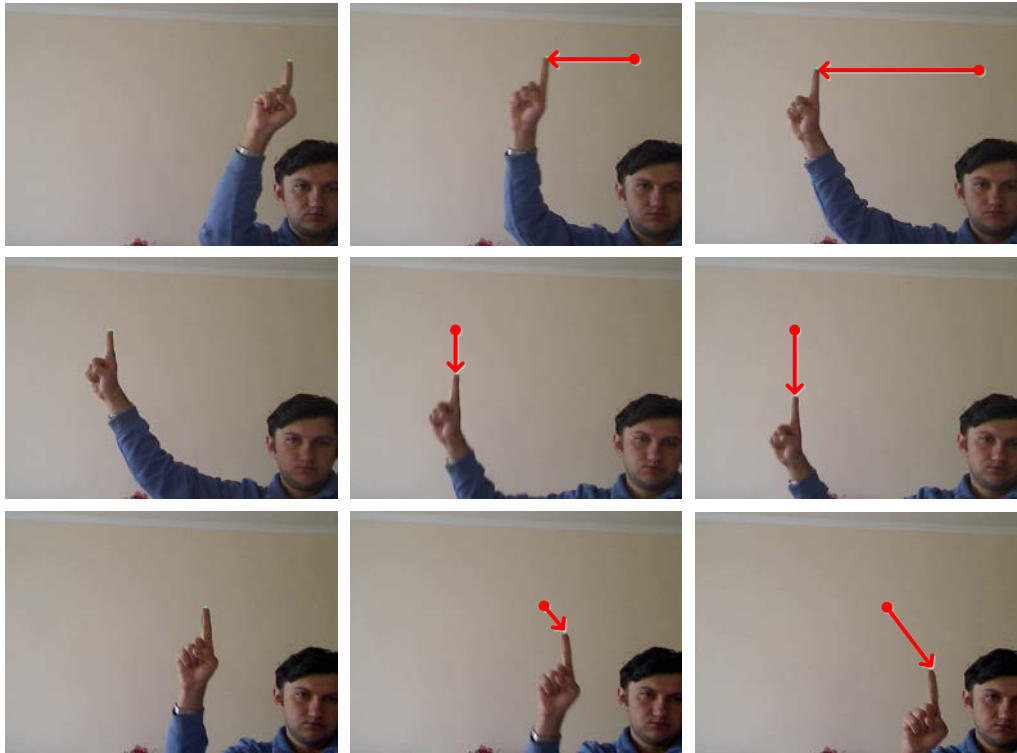


Fig. 4.19. Tracking the finger at fast motion speed:
top row – frames 64, 68 and 72; middle row – frames 79, 81 and 82;
bottom row – frames 110, 111 and 113 of test sequence 4

the vertical movement. The position prediction mechanism in the tracker guiding layer allows accurate tracking at various speeds, keeping the search area to a reduced size, which reduces the amount of data to process in the finger detection layers. Fast motion was present in another 4 test sequences and the tracking results were accurate, except for a few frames with extreme motion blur in test sequence 2.

My previous experiments with the CAMSHIFT tracker revealed real difficulties in dealing with the presence of other skin colored objects in the neighborhood of the target, often leading to irreversible target loss. Therefore, I tested the performances of the proposed tracking algorithm when the tracked finger was close to other fingers or to the face, which have similar color and are also foreground objects in the scene. In the neighborhood of the face, the proposed tracking algorithm performed well, being able to track the finger as long as it was not superposed on the face. The top row of Fig. 4.20 presents hand details in sample frames with tracking results in the neighborhood of the face, in test sequences 2, 5 and 7. Tracking the finger while crossing the face area, only succeeded at slow movement speed, but, due to the tracker guiding mechanism, the tracker was able to recover after the finger/face superposition period ended.

Tracking the finger and the hand when another hand with an extended finger is in the neighborhood of the target is a very challenging situation. The disturbing object not only has the same color and is a foreground object (as it is the case with the face), but also has similar geometrical characteristics. This type of



Fig. 4.20. Tracking the finger in the neighborhood of other skin colored objects.

situation occurred in the recorded test video sequences 5 and 7 and the tracker was able to stay focused on the real target even when the 2 fingers were close to each other as shown in the bottom row of Fig. 4.20. During tests with live video sequences, target confusion seldom occurred when the 2 fingers were parallel or almost parallel, with the tracked finger moving at relatively high speed toward the false target and abruptly stopping the movement when the two fingers got very close to each other.

Tracking also succeeded when total occlusion with another finger (stationary or moving in opposite direction) occurred. Fig. 4.21 presents hand details in a frame sequence from test video sequence 7, in which the tracked finger passes behind the false target, and the tracker is able to stay focused on the real target, mainly due to the finger identification tracker guiding layers.

Total occlusion caused by objects of different shape and/or color causes the proposed algorithm not to detect the finger during the occlusion period. Although the location prediction mechanism allows the tracker to recover from the occlusion state if the finger returns to a visible state either in the neighborhood of the location where it disappeared or in a location on the opposite side of the object that causes the occlusion, after a time interval that would allow the finger to pass across the object. In the context of the desired applications, it is assumed that the finger does

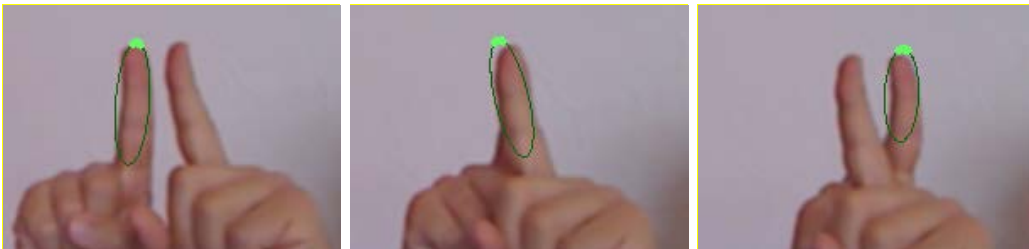


Fig. 4.21. Tracking results when the finger is partially occluded: frames 71, 74, 219, 222, 227 and 231 of test sequence 7.

not disappear behind objects that belong to the background and therefore the objects that cause the occlusion can be detected as foreground objects.

The tracking algorithm runs in real time for video sequences with resolutions of 640×480 and 960×720 pixels, respectively, on a notebook with an Intel Core 2 Duo T7500 processor with 2.2 GHz clock. The most time consuming part is represented by the preprocessing steps, which include time consuming operations (i.e. color space conversions and foreground/background segmentation), especially when applied to the entire frame. Execution times – measured under debug mode, without code optimization – for finger detection and tracker guiding vary between 7 – 45 ms (less than 25 ms in 78% of the frames and an average value of 19 ms). The finger detection and tracker guiding layers are computationally efficient due to the significant reduction of the data to process through the use of line strip features.

A comparison of tracking results obtained with the proposed algorithm, the CAMSHIFT algorithm and the algorithm in [2], is presented in Table 4.2. In sequences with fast motion, the proposed algorithm thoroughly follows the target. The CAMSHIFT algorithm also performs well, but only if no other skin colored object is present, while the algorithm in [2], using a zero order motion prediction, loses track of the target when it moves outside the fixed search window. When other similar objects are present near the target, the proposed algorithm and the algorithm in [2] are able to keep track of the real target, while the CAMSHIFT algorithm often confuses the real target with a similar object or merges multiple

Table 4.2. Tracking Results Comparison.

Test Scenario	CamShift	Algorithm [2]	Proposed Algorithm
Motion blur	Conditionally	No	Yes
Scale changes	Yes	Yes	Yes
Partial occlusion - similar	No	Confusion Possible	Yes
Partial occlusion - different	Conditionally	Yes	Yes
Fast motion	Conditionally	No	Yes
Similar objects around	No	Yes	Yes
Total occlusion recovery (similar)	Confusion Possible	Confusion Possible	Yes
Total occlusion recovery (different)	Conditionally	Conditionally	Conditionally
Across face	No	No	Conditionally

objects into a larger false target. When total occlusion from a similar object occurs, the proposed method is able to correctly recover based on the motion prediction mechanism, while the other 2 algorithms often confuse the real target with the occluding object. Total occlusion from different larger objects is potentially problematic for all 3 algorithms.

The proposed method, based on the motion prediction mechanism, is able to correctly recover from such a situation only if the target re-appears near the location of the disappearance or on the opposite side of the occluding object. The algorithm in [2], with its zero order motion prediction, can only recover near the location of the disappearance, while CAMSHIFT is able to recover in any position, given that no other skin colored object is present in the scene. Tracking the finger across the face has poor results with the proposed method (only works at slow motion speed under good lighting conditions), while the other 2 algorithms fail to track the target in this scenario.

Compared to the CAMSHIFT algorithm, which relies on a single feature, the proposed multi-cue algorithm is able to better keep track of the target, especially near other skin colored objects or when skin-like colors are present in the background. Also, the proposed algorithm is able to effectively track the finger under appearance scale changes. In my past experiments, the CAMSHIFT algorithm often lost the target during appearance scale changes, when other skin-like colored areas were present in the scene.

Compared to the tracking algorithm in [2], the proposed method showed similar results for scale changes, partial occlusions and skin like colored background and significantly improved results when the target moved very fast, when crossing another similar object and in scenes containing motion.

5. REAL-TIME DYNAMIC HAND GESTURE RECOGNITION

Human gestures are expressive human body motions, which generally contain spatial and temporal variation [85]. To handle these variations, an appropriate representation must be chosen. As they represent important communication means between human subjects, gestures also have an important potential for communication between human subjects and computers. The most important obstacle to overcome in order to achieve gesture based HCI, is to make human gestures understood by the computer (i.e. develop real-time algorithms that allow the computer to recognize human gestures).

A vast amount of work in gesture recognition has been performed in the area of computer vision, and is reviewed in [86], [87], [88] and [89]. These works can be divided into two categories: trajectory based and dynamics model-based analysis. The trajectory based approach matches curves in configuration space to recognize gestures [90]. The dynamics model-based approach learns a parametric model of gestures.

Recognition of human gestures is important for HCI, automated visual surveillance, video library indexing [85], control of video-games [91], remote control of home appliances, such as TV sets and DVD players [92], which in the future could be extended to the more general scenario of ubiquitous computing in everyday situations.

Gestures can be static (information is conveyed by specific static positions of the user or of specific human body parts) or dynamic (information is transmitted through temporal evolution of the shape and/or position of the user or of specific parts of the user's body). Combinations of the two types of gestures are also possible (e.g. sign language). Static gesture recognition relies on features like edges in order to encode in a symbolic form the shape of the object of interest (e.g. hand) and to classify the pose as a valid gesture when appropriate [93]. Dynamic gesture recognition relies on the analysis of temporal variations of shape and/or spatial position of the object of interest over a relatively short time span.

Depending on the parts of the human body involved in performing the gesture, gestures can be divided in:

- Finger, hand and arm gestures – recognition of hand poses, trajectories of hands and/or fingers, with a wide range of applications including sign language recognition,
- Head and face gestures – head movements (nodding, shaking), eye blinking, direction of eye gaze, emotions encoded in facial expressions (sadness, happiness, surprise),
- Body gestures – involve the movement of the entire body, like in recognizing the movements of a dancer or the human gait [87].

Dynamic hand gesture recognition methods can basically be divided into two categories: data-glove based methods and vision based methods. Glove-based approaches require the user to wear cumbersome, uncomfortable and expensive devices, containing various types of sensors, connected to the computer through one or more bunches of wires, which hinder the naturalness of the HCI. The main advantages of these methods are the availability of detailed information about all movements of the hand and fingers (including precise movements) and the

immunity to disturbing factors, like occlusions (including self-occlusions), clutter, lighting variations etc. Vision based methods rely on video camera(s) as input devices and have the potential to offer natural communication interfaces, without hindering the user in any way. On the other hand, vision based methods cannot capture fine movements of the hand and fingers and the tracking and gesture recognition processes need to overcome difficult obstacles, like occlusions, clutter scale changes, lighting changes etc. These obstacles make the vision based gesture recognition a challenging interdisciplinary research subject, involving computer vision, image processing, machine learning, bio-informatics and psychology [89].

The most important aspects that influence the success of a gesture recognition system are: robustness, computational efficiency, scalability, the user's tolerance and the naturalness of the gesture set used for communication. The robustness refers to the ability of the gesture recognition algorithm to overcome obstacles like those mentioned above. Computational efficiency is related to the requirement to perform all operations in real-time with cost-effective processing devices. Scalability refers to the easiness of adapting the gesture recognition system to different scales of possible applications (communication with large public information panels, with desktop computers, notebooks and small sized mobile devices). The user's tolerance requires the implementation of some protection against potentially destructive effects due to false gesture detection. The naturalness of gestures requires the gestures used for communication to be selected (defined) in such a manner, that they can be easily memorized and executed by the user within a reasonable time interval.

Vision-based gesture recognition systems in general are composed of three main [94] components: image preprocessing, tracking and gesture recognition. In individual systems, some of these components may be merged or missing, but their basic functionality [95] will normally be present. This chapter focuses on the last processing step – gesture recognition – given that the image preprocessing and tracking tasks were discussed in the previous chapter. Gesture recognition decides if the user is performing a meaningful gesture, based on the collected position, motion and pose clues.

The classical algorithms from the field of pattern recognition are Hidden Markov Models (HMM), correlation and Neural Networks. Especially the first two have been successfully used in gesture recognition, while the Neural Networks often have the problem of modeling non-gestural patterns [96]. HMM is a typical dynamics model and was proven to be robust in the recognition of gestures [97]. The HMM model has been extended to a more general model, named Dynamic Bayesian Networks [98].

In this thesis, hand gestures and a trajectory based approach are used. In the papers [5] and [6], I proposed a solution centered on the mean shift algorithm. The mean shift algorithm is used for trajectory filtering and segmentation in the gesture recognition process and – on the other hand – the trajectories used for testing the performance of the gesture recognition are provided by a mean shift based tracking algorithm (CAMSHIFT). The trajectory based approach was chosen, as skin detection is quite well developed and robust [10], and relatively robust low computational cost tracking algorithms are also available [11], [12].

The main goal in the design and implementation of the gesture recognition system proposed in [5] and [6] was to develop a low computational cost real-time algorithm, able to run on low-complexity hardware systems. The system must be able to learn from the user a small number of gestures, in order to recognize them later.

5.1. Trajectory based gesture recognition

Trajectory based gesture recognition represents a subspace of the more general class of dynamic gesture recognition. While the dynamic gestures may convey information through temporal evolution of the shape and/or position of the user or of specific parts of the user's body, trajectory based gestures convey the information exclusively through the temporal evolution of the position (spatial trajectory) of the target (e.g. hand, finger, head). Considering the target's shape variations in time, in addition to the trajectory, allows for encoding more information within the same time span of the gesture. On the other hand, the additional information comes at the cost of a significant increase in computational complexity. Vision based gesture recognition systems using a single camera – which cannot capture detailed information about all the subtle movements of the target – have to overcome many obstacles, in order to get access to the information encoded in the shape variations and rotations of the target. Therefore, using only the trajectory information may significantly reduce the complexity of the gesture recognition process and at the same time increase its robustness.

For trajectory based gestures, a tradeoff needs to be done between the naturalness of the gestures and the ease (i.e. the reduction of computational complexity) of gesture recognition processing. In this context, the definition of the gesture set may have a significant influence on both the computational complexity and the robustness of a trajectory based recognition system. Gestures which allow all kinds of trajectories have an increased naturalness potential, but the process of gesture recognition requires more complex processing and the chance of unintended gesture detection is increased. On the other hand, gesture sets which impose restrictions on the trajectory shapes allow for easier processing of the trajectory and reduce the chance of accidental (unintended) gesture detection, but the restrictions on the trajectory shapes may hinder the naturalness of the gestures.

Many solutions have been proposed in literature for dynamic hand gesture recognition in various frameworks. Amongst the most popular solutions are those based on HMMs. With the theoretical mathematical fundamentals developed since the '60s, HMMs were used in the modeling of human motion and in gesture recognition applications since the '90s [99], after they were proven to be very effective in speech recognition applications [100].

HMMs represent rich mathematical structures, which proved to be very adequate for modelling spatial-temporal information in a natural way [87]. This property kept the HMMs as one of the most appealing solutions to the researchers in the dynamic gesture recognition field to date [101], [102] and [103].

HMM relies on a Markov chain with finite number of states and a set of random processes, each of them associated to a state. At a given discrete moment in time, the process is in one of the states and generates an observation symbol based on the random function corresponding to the current state. Therefore, the transitions between states are governed by pairs of probabilities: transition probabilities (the probabilities to have transitions between states) and output probabilities (conditional probabilities of observing the output symbols given the current state). The term "hidden" in the name, states that all that is visible is only the sequence of observations. Some key issues in the use of HMMs are the evaluation (determination of the probability that the given sequence of observations was generated by the model), training (adjusting the model to maximize the probabilities) and decoding (use Viterbi algorithm to recover the state sequence).

Dynamic Bayesian Networks [98], which can be considered a generalization of HMMs, have also shown promising results in dynamic hand gesture recognition [104] and [105].

Finite State Machines (FSM) represent another popular approach with the recognition of dynamic hand gestures [106], [107] and [108]. FSM based solutions model the gestures using an ordered sequence of states in the space-time domain. The trajectories of the target are represented as sets of 2D points and in addition temporal information may be associated to each point. The FSM has to identify, from the unsegmented input trajectory, the sequences of states that match valid gestures. Valid gesture patterns can either be learned by initial training [106] or described using symbolic features [109].

Another major category of solutions used in the recognition of dynamic hand gestures is represented by the neural networks based approaches [110], [111] and [112]. Systems in this category can either be implemented as completely trained when in use, or with the ability to dynamically adapt to the user. Neural network based methods usually also handle spatial shape variations of the target, but they may require training and important computational resources.

Given that straight trajectories are easier to process and the gestures composed of successions of relatively straight lines (strokes) are also easy to learn and execute by a user, I proposed in [5] and [6] a trajectory based gesture recognition method for recognizing gestures of this type. The gesture set chosen contains gestures consisting of 2 – 4 straight strokes. Gestures containing a single stroke are even easier to detect, but their inclusion in the gesture set may increase the chance of detecting accidental hand movements as valid gestures. Though the proposed algorithm also allows easy recognition of gestures containing more than 4 straight strokes, the gestures consisting of a large number of strokes are not easy to learn and remember for the user and therefore the inclusion of such gestures would hinder the naturalness of the HCI.

In the gesture recognition solution I proposed in [5] and [6], the hand trajectory is first processed for gesture segmentation and symbolic encoding, followed by a gesture recognition step which identifies the gesture by matching the symbols with the prototypes of valid gestures stored in a table. A FSM approach is used for gesture segmentation.

5.2. Hand trajectory processing

In the framework used for the research reported in papers [5] and [6], the trajectories of the hand are extracted using the CAMSHIFT algorithm [22] available in the OpenCV library. The trajectory recording process is governed by a FSM with 4 states, as shown in Fig. 5.1:

- 1) gesture start – static position for a short time interval,
- 2) gesture motion – continuous movement of the hand,
- 3) gesture end – static position for a short time interval and
- 4) non-gesture motion – hand motion between gestures (not subject to further analysis).

The only data which is further processed is the trajectory recorded when the state machine is state 2. First, a smoothing filter is used on the trajectory. Then, the data (an array of trajectory points' coordinates) is converted into an array of segments' slopes (angles), which are further filtered for outliers' removal. Next, the trajectory is segmented by clustering the data based on the angles' values and the gesture is encoded with a data structure with the following fields: (number of

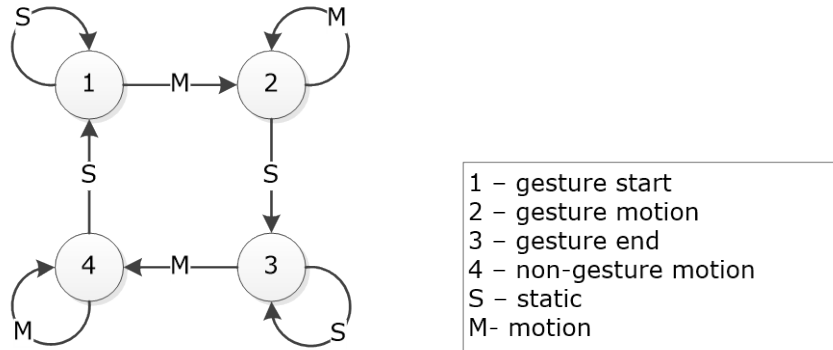


Fig. 5.1. Gesture segmentation state machine.

strokes), (strokes angle sequence – the number of angle values is equal to the number of strokes), [(strokes lengths sequence)]. The last field is only useful for gestures which also contain proportionality data as relevant information. In the proposed gesture set, some 4-stroke gestures encode this type of information (e.g. for distinguishing between a square and elongated rectangles). For the 2- and 3-stroke gestures, the lengths are not used – gesture recognition is performed in these cases based only on the number of strokes and the slopes' sequence.

5.2.1. Trajectory recording and smoothing

The recording of a new gesture trajectory is triggered by a movement of the user's hand, occurring after a short interval (1-2 seconds) of static position. Minimum thresholds are imposed to the amplitude and speed of the movement, in order to avoid false triggering due to tracking noise or hand trembling. The gesture trajectory recording ends when the movement speed falls below the imposed threshold for at least 1.5 seconds.

The consecutive centers of the tracked region define a relatively rough (noisy) trajectory, increasing the difficulty of strokes detection. Therefore, the trajectory is smoothed so that each new trajectory point, $t[i]$, is obtained as a weighted average of the new measured point, $m[i]$, and the previous trajectory point, $t[i-1]$:

$$t[i] = \alpha m[i] + (1 - \alpha)t[i-1]. \quad (5.1)$$

The weighting parameter must be carefully chosen, as a too small value would over-smooth the trajectory, while larger values lead to relatively rough trajectories.

5.2.2. Trajectory segmentation

A set of angles with the horizontal axis is computed over the recorded trajectory. Computing the angle for each small segment determined by two consecutive points of the trajectory may result in a very noisy angle set, with many false angle discontinuities. This noise is caused by angles between trajectory points that are relatively close to each other, because the image is sampled on a

rectangular grid. Selecting a reduced number of trajectory points using a fixed step (e.g. choosing each second or third point of the trajectory) results in a relatively smoothed angle set. To further improve the results, I chose to adaptively select trajectory points based on a threshold distance. An adequately chosen threshold distance significantly reduces the number of outliers in the resulting angle sequence. Even with the fixed step selection, a distance threshold must be imposed, in order to avoid computing the angle if the two points have the same position.

In order to split the trajectory into strokes, the ends of these strokes must be detected. The starting point of the trajectory is also the starting point of the first segment, and the end point of the trajectory is the end point of the last segment. All other strokes' end points (trajectory turning points) are detected as slope discontinuity points, the starting point of each segment being the end of the previous segment. A stroke discontinuity is detected as a point where the segment angle with the horizontal axis changes significantly. A threshold must be imposed on the minimum length of a stroke, in order to avoid detection of false strokes.

As the raw angle set computed adaptively over the trajectory may still contain some outliers – especially around the strokes' joint points – filtering is necessary in order to simplify the feature extraction process.

In typical situations, median filtering within a 5 angles window reduces the roughness of the angle sequence over a stroke, but used alone it is not very helpful when either very low or very high frequency noise is present.

A second filtering approach, which I implemented and tested, uses an intelligent classification filter with a 7 angles window. Each of the 7 angles in the window is assigned to one of 8 angle classes, and the number of angles in each class is counted. Ideally, all the angles over a stroke should belong to the same class. Exceptions may occur due to high segment angle noise or due to a neighboring stroke's end/joint point. If at least 4 angles in the window belong to the same class, the processed angle is replaced with that class's typical value. If either the resulting class differs from the current stroke's class or no class has at least 4 representatives in the window, the filter advances and checks the next angles to detect if there is a new stroke, or just a noisy angle group. A new stroke is assumed if all the angles for a group of consecutive segments that cover the minimum stroke length are filtered to the same class.

The third approach, which I implemented and tested, uses a mean shift based clustering of the angle set and is the most robust and efficient. A 2D anisotropic kernel obtained by multiplication of 2 1D symmetric kernels with different profiles is used. Considering that the data vector, $\mathbf{x} = [\theta, p]$, consists of the segment angle, θ , and position in the angle sequence, p , the kernels for the two dimensions are:

$$G_1(\theta) = c_{g_1} \cdot g_1\left(\|\theta\|^2\right), \quad G_2(p) = c_{g_2} \cdot g_2\left(\|p\|^2\right). \quad (5.2)$$

Considering different bandwidths for the 2 kernels, the resulting product kernel is:

$$\begin{aligned} G(x) &= \left[\frac{c_{g_1}}{h_\theta} \cdot G_1\left(\frac{\theta}{h_\theta}\right) \right] \cdot \left[\frac{c_{g_2}}{h_p} \cdot G_2\left(\frac{p}{h_p}\right) \right] \\ &= \frac{c_{g_1} c_{g_2}}{h_\theta h_p} \cdot G_1\left(\frac{\theta}{h_\theta}\right) \cdot G_2\left(\frac{p}{h_p}\right). \end{aligned} \quad (5.3)$$

Bandwidth selection is an important problem in nonparametric methods and the main categories of available solutions to this problem were presented in chapter 2.3. Product kernels obtained by multiplication of 1D kernels with identical profiles but different bandwidths in different dimensions were used in [72]. Anisotropic kernels were also previously used in computer vision for image and video segmentation [113] and video tracking [36]. In the related approach, bilateral filtering, Tomasi and Manduchi [114] also use separable kernels for the space and range domains. The relationship between the bilateral filters and the mean shift algorithm is emphasized in [115].

In order to guarantee the convergence of the mean shift algorithm, the shadows of both 1D kernels must be convex and monotonically decreasing, so that their resulting 2D product kernel is also convex and monotonically decreasing.

In the angle domain, a uniform kernel is used with an initial bandwidth, h , which may be later increased to merge clusters that are very close to each other in this domain. The shadow of the uniform kernel is the Epanechnikov kernel, which has a convex and monotonically decreasing profile, satisfying the convergence condition. If the mean shift algorithm is applied based only on the statistical information, some outliers may survive the filtering process (if they correspond to the cluster of a different stroke present in the trajectory).

Spatial information is taken into account using a spatial isotropic kernel (e.g. triangular, Epanechnikov, normal), that assigns larger weights to the angles of the segments in the middle of the window and smaller weights to those in the extremities. Each of the above mentioned isotropic kernels have shadows that satisfy the convergence condition.

Similar to the bilateral filtering approach, the mean shift is applied only for the angle dimension, while the kernel in the space domain does not change its central position between the mean shift iterations.

Using the uniform kernel for angles and the Epanechnikov kernel for the spatial information (the position in the angle sequence), the mean shift vector becomes:

$$ms_{h_\theta, h_p, G}(\theta) = \frac{\sum_{i=1}^n \theta_i G_1\left(\frac{\theta - \theta_i}{h_\theta}\right) G_2\left(\frac{p - p_i}{h_p}\right)}{\sum_{i=1}^n G_1\left(\frac{\theta - \theta_i}{h_\theta}\right) G_2\left(\frac{p - p_i}{h_p}\right)} - \theta. \quad (5.4)$$

Clusters with very few angles (not numerous enough to define a stroke) are removed from analysis and the corresponding angles are each assigned to the spatial neighboring cluster that is nearest in the angle domain. Finally, only clusters that correspond to plausible strokes are kept and the stroke ends are detected as points where the angle cluster changes in the angle sequence.

In the mean shift clustering process, special care is taken for the angles in the intervals $[180^\circ - h, 180^\circ]$ and $[-180^\circ, h - 180^\circ]$, due to the circular definition of the angle as shown in Fig. 5.2.

Therefore, if the center of the kernel window falls within the $[180^\circ - h, 180^\circ]$, angles from the $[-180^\circ, h - 180^\circ]$ interval will also be taken into account by converting them to the $[180^\circ, h + 180^\circ]$ domain. Similarly, when the angle kernel window is centered on a value within the $[-180^\circ, h - 180^\circ]$ interval, values from the $[180^\circ - h, 180^\circ]$ interval will be taken into account by converting them to the $[-$

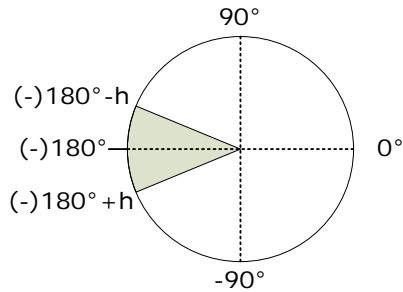


Fig. 5.2. Dealing with the circular definition of the angle for correct mean-shift clustering.

$180^\circ - h, -180^\circ]$. Finally, if the resulting filtered value is outside the interval $(-180^\circ, 180^\circ]$, it will be converted to fit this interval.

The results obtained with the second and third filtering approaches are further improved if the median filtering is used as a preprocessing step.

5.2.3. Feature extraction

After the stroke ends are detected, the parameters of the trajectory are extracted. The useful parameters which must be extracted are: the number of strokes, the strokes' angle sequence and the strokes' lengths.

A simple analysis of the mean shift filtered angle sequence obtained at the previous step allows the extraction of the required parameters: the angle and end points for each stroke and the total number of strokes. The strokes' lengths can be easily obtained using the coordinates of the stroke ends.

Each gesture known by the system is represented by a codified angle sequence. The angles of the 8 directions allowed and the associated codes are presented in Table 5.1. Opposite directions have complementary codes.

The angle of each stroke must be classified and assigned to one of the 8 classes. In order to assign the angle to a class, its value must fit a $\pm 20^\circ$ window around the standard value. A 5° guard space is left between consecutive windows as shown in Fig. 5.3.

Table 5.1. Codes associated to angle directions.

θ	0°	45°	90°	135°	-45°	-90°	-135°	180°
Code	0	1	2	3	4	5	6	7

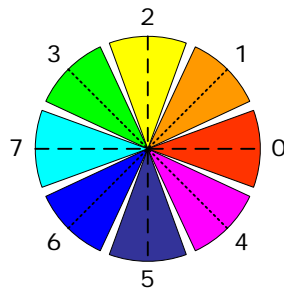


Fig. 5.3. Angle classes.

If a stroke's angle falls within a guard space it is not possible to classify it and the analysis is stopped, invalidating the current gesture.

The first angle of the sequence is the angle between the first stroke and the horizontal axis, while the next angles can be either the angles made by each stroke in the sequence with the horizontal axis or with the previous stroke. The second solution may increase the robustness to small global trajectory rotations, but reduces the possible gestures alphabet's size.

A global rotation correction may be applied to the strokes angle sequence if a large median deviation from the nearest class is detected. Fig. 5.4 shows an example of a 3-stroke gesture that exhibits a global -15° rotation. Direct classification of the strokes angle sequence (left side) produces the erroneous sequence $(0^\circ, -90^\circ, 0^\circ)$, while the global rotation correction allows a correct classification of the sequence to $(0^\circ, -135^\circ, 0^\circ)$.

At the end of the trajectory segmentation process, the endpoints of all strokes are known. As all valid strokes are assumed to be linear, it is straightforward to compute the approximate length for each stroke, as the Euclidean distance between its endpoints:

$$l_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}, \tag{5.5}$$

$(x_i, y_i), (x_{i+1}, y_{i+1}) - (x, y)$ coordinates of the i -th stroke endpoints.

The calculation of stroke lengths can be skipped for 2- and 3-stroke gestures, given that in these cases the proportionality between stroke lengths is not relevant for the gesture recognition step.

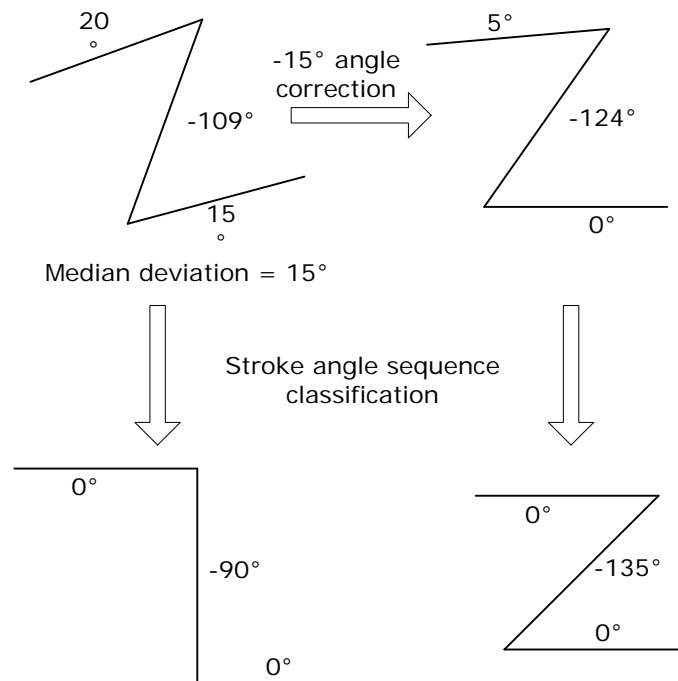


Fig. 5.4. Effect of global rotation correction.

5.3. Gesture recognition

The gesture discrimination process is very simple, and does not require complex classification based on curve matching. A gesture consists of a minimum of 2 strokes and is uniquely identified based on:

- number of strokes (>1),
- slope (angle) sequence and
- strokes proportionality (optional).

Each gesture prototype is encoded using an array of symbols containing the number of straight strokes, the slopes of the strokes and – optionally – proportionality information. A huge number of gestures can be defined in this way. The maximum number of distinct gestures consisting of 2, 3 and 4 strokes, without using proportionality as a discriminant, is:

$$NumG_{max} = \underbrace{8 \times 7}_{2\text{-stroke gestures}} + \underbrace{8 \times 7 \times 7}_{3\text{-stroke gestures}} + \underbrace{8 \times 7 \times 7 \times 7}_{4\text{-stroke gestures}} = 3192. \quad (5.6)$$

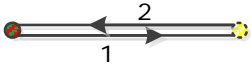
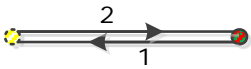
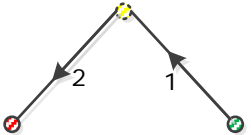
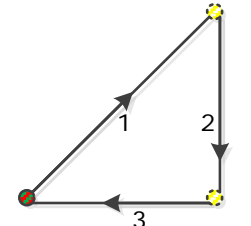
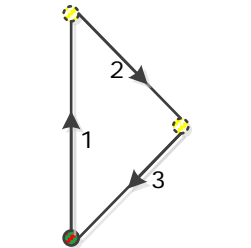
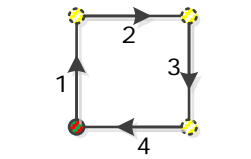
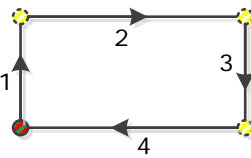



The first stroke may have any of the 8 defined reference slopes, while for each of the subsequent strokes the number of possible slopes is 7 (a new stroke cannot have the same slope as the previous one). This number can further be increased by adding proportionality information for some of the gestures. Depending on the application, a small amount of them should be selected (based on naturalness of execution and intuitiveness). The gestures used should be chosen so that the user can easily execute them and learn the gesture –action correspondence. Multiple gestures having the same intuitive meaning can be associated to a single action. For example the 'Up' command can be indicated either by a succession of 2 vertical strokes (upward followed by downward) or by any 2 consecutive strokes making an angle in the range (0° – 90°] pointing upwards.

Table 5.2 presents some examples of gesture prototypes and their symbolic representations (the slope sequences are encoded according to the correspondences in Table 5.1). Gestures 1 – 3 consist of 2 strokes, gestures 4 and 5 have 3 strokes, while the last 2 examples (gestures 6 and 7) consist of 4 strokes. Gestures 1 and 2 differ only in the order of the strokes. Examples of actions to which these gestures could be naturally associated are 'Previous' / 'Left' and 'Next' / 'Right'. Gesture 3 can be associated with the 'Up' command as well as a 90° rotated version of gesture 1. Gestures 6 and 7 have the same number of strokes (4) and the same slopes sequence and are discriminated based on the proportionality information.

The discrimination process is done sequentially, based on the 3 parameters, as shown in Fig. 5.5. As all gestures that reach this processing step were already validated from the number of strokes and angle sequence point of view, these parameters are strict, while the stroke proportionality is allowed to vary within an error window. The first parameter to take into account when discriminating between gestures is the number of strokes. This step removes from further analysis all gestures with a different number of strokes. In the next step, the angle sequence is decoded. In most cases, this step is enough to uniquely identify the gesture and terminate the analysis.

Some 4-stroke gesture codes (e.g. square/rectangle codes) may require further classification based on strokes proportionality. The strokes' proportions are allowed to vary within an error window. The error window must be chosen large enough to accommodate the imperfections of the hand drawn trajectory, but small enough to allow correct discrimination between different gestures.

Table 5.2. Gestures examples with corresponding symbolic representation

No.	Gesture trajectory prototype	Symbolic representation
1		(2), (0, 7)
2		(2), (7, 0)
3		(2), (3, 6)
4		(3), (1, 5, 7)
5		(3), (2, 4, 6)
6		(4), (2, 0, 5, 7), [$\frac{V}{H} = 1$]
7		(4), (2, 0, 5, 7), [$\frac{V}{H} = 0.5$]
<p>  – Start point  – Turning point  – End point </p>		

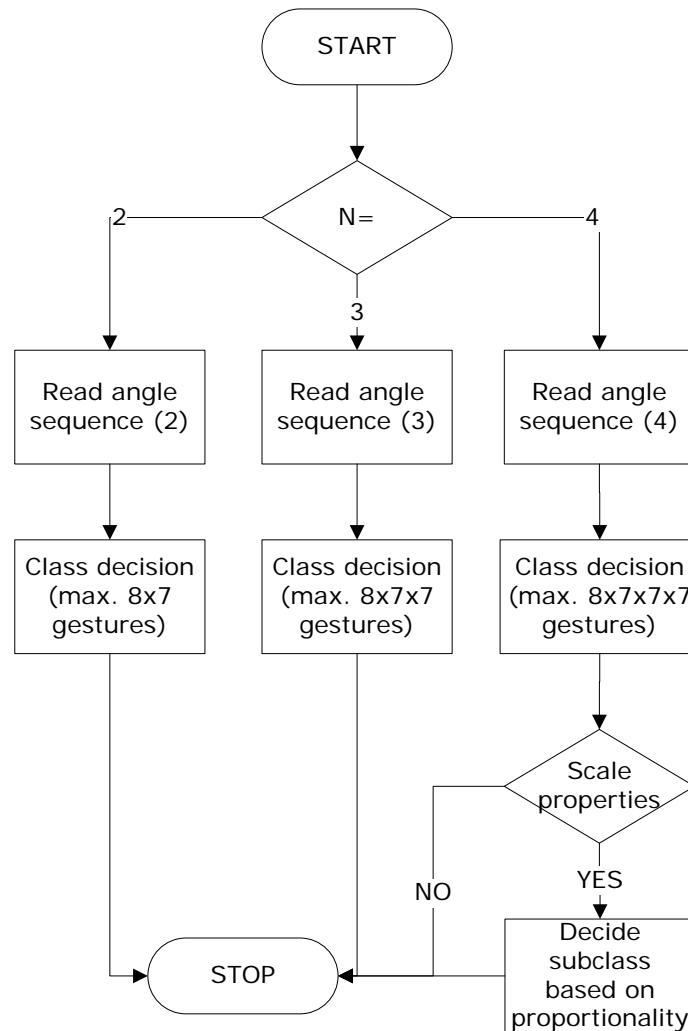


Fig. 5.5. Sequential gesture discrimination.

A guard space must be allowed between gestures that are distinguished based on the proportionality features and if the proportions fall within the guard space, the user is asked to repeat the gesture.

5.4. Results and discussions

The hardware processing system used to implement and test the proposed gesture recognition solution was a PC with a 1.6 GHz AMD Athlon XP processor and 768 MB of RAM. Video acquisition was realized using a commercial USB webcam with 352×288 video resolution. The software application was implemented in Visual C++ and some functions from the OpenCV library were used.

The application uses a tracker based on the CAMSHIFT function of the OpenCV library. This algorithm is able to track a hand based on the hue, assuming saturation and value thresholds are relatively well calibrated and no occlusions occur and no other objects of similar color are present in the neighborhood of the hand. Fig. 5.6 presents the tracker focused on a hand.

The practical tests revealed that a value of 0.4 for the weighting parameter, a , in equation (5.1) produces a relatively smooth trajectory. A threshold distance of 3 pixels was used in the computation of the segments angle sequence, for the 352×288 frame size. A threshold was also imposed on the minimum length of a stroke, in order to avoid detection of false strokes. A reasonable value for this threshold is 1/10 of the image height. The initial bandwidth of the uniform kernel used in the angle domain is 20°, but may be later increased in order to merge clusters that are very close to each other in this domain. The bandwidth used for the Epanechnikov kernel in the position domain is 20.

The use of Epanechnikov kernel in the position domain, even with such a large bandwidth, avoids taking into account angle values from a far stroke, which could be similar to some noisy angles around the current processed position. Even if such angles would fall within the kernel window, their influence would be diminished by the small weights near the kernel window's ends. Without using the kernel in the spatial domain, noisy angles may survive in the filtered angle sequence, if their values are similar to those of another stroke.

The recorded trajectory for a 3-stroke gesture is presented in Fig. 5.7. The displayed trajectory is drawn based on pre-smoothed recorded positions of the hand. The resulting stroke angle sequence for the gesture in Fig. 5.7 is: 45°, -90°, 180° and the resulting coded sequence for this example is "1, 5, 7".



Fig. 5.6. The CamShift tracker.

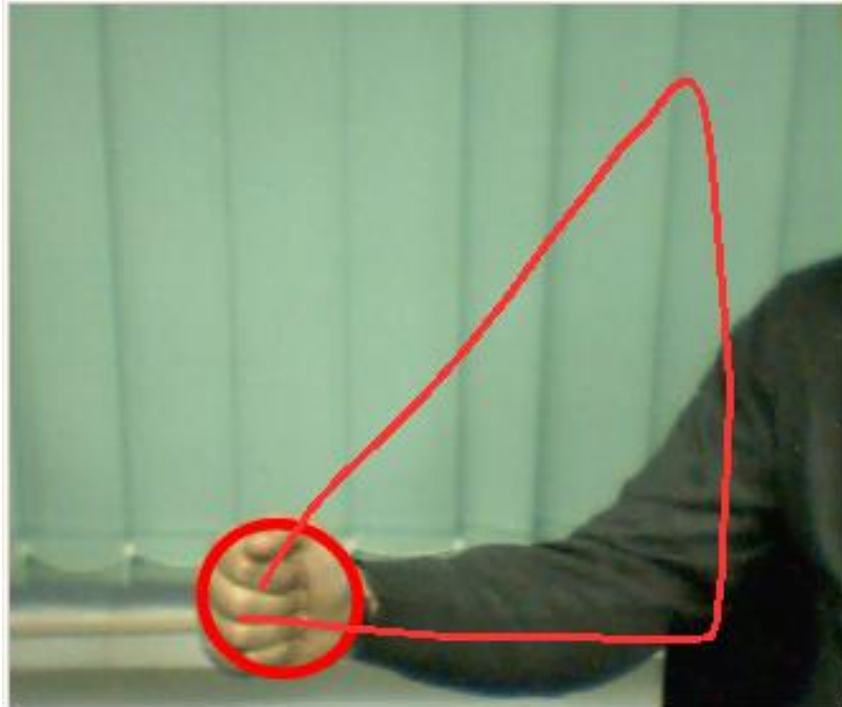


Fig. 5.7. Trajectory of a 3-strokes gesture.

Histograms of the slopes sequences for examples of 2, 3 and 4-strokes gestures are presented in Fig. 5.8, Fig. 5.9 and Fig. 5.10. The 3 histograms of subsection angles indicate that the subsection angles are clustered, allowing easy separation of the strokes.

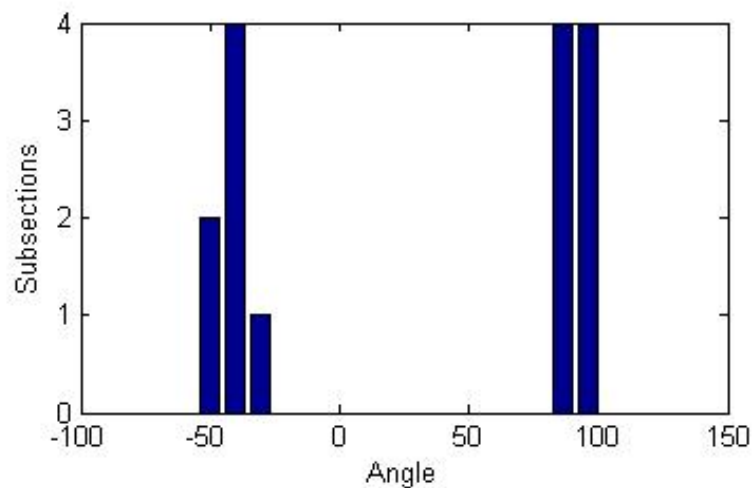


Fig. 5.8. Angle histogram for 2-stroke gesture.

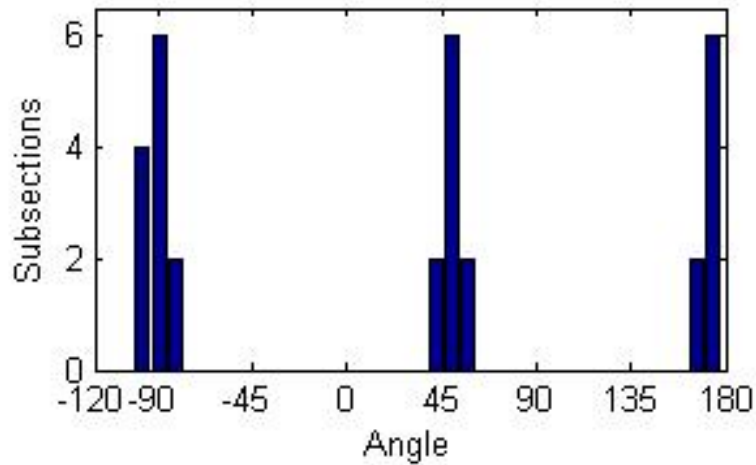


Fig. 5.9. Angle histogram for 3-stroke gesture.

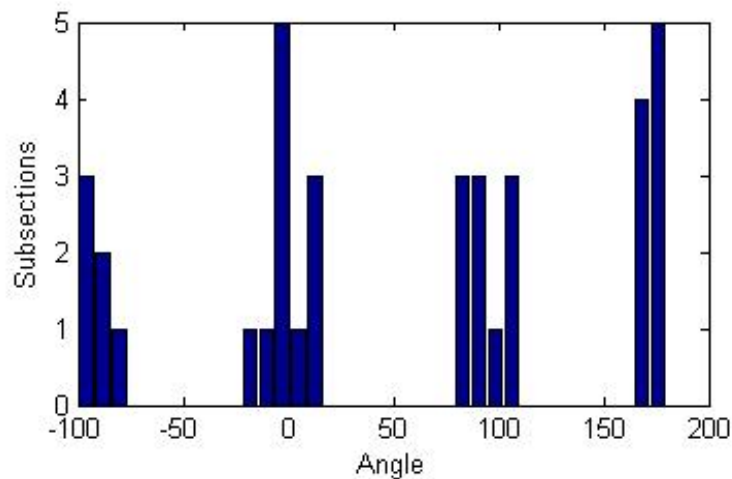


Fig. 5.10. Angle histogram for 4-stroke gesture.

Tests were performed on a total number of 60 gestures, executed by 3 subjects in both daylight and artificial light. The tested gesture sequences were composed of 2, 3 and 4 variable length strokes, with sharp and/or right angles between consecutive strokes.

Fig. 5.11 presents a histogram of standard deviations in raw angle sequences, realized using 70 strokes from 25 randomly chosen gestures. The standard deviation of the angle over a stroke varied between less than 5° for near linear strokes and over 30° for rough ones. After combining median filtering and mean shift filtering, the standard angle deviation over a stroke is reduced to $0^\circ - 3^\circ$.

The mean shift based algorithm was able to correctly identify all gestures, even under challenging conditions, like hand trembling or short term target losses in a cluttered environment.

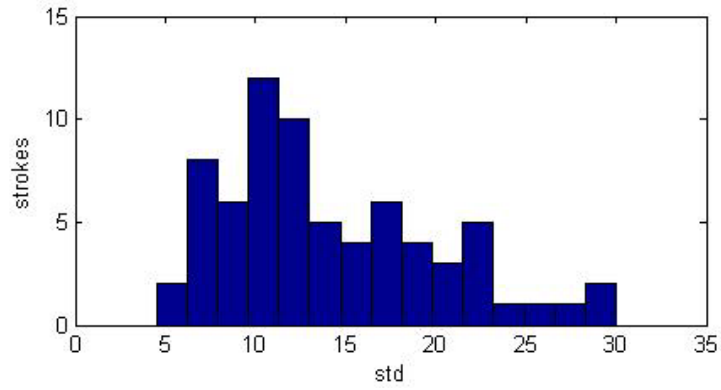


Fig. 5.11. Histogram of the standard deviations over a set of 70 strokes.

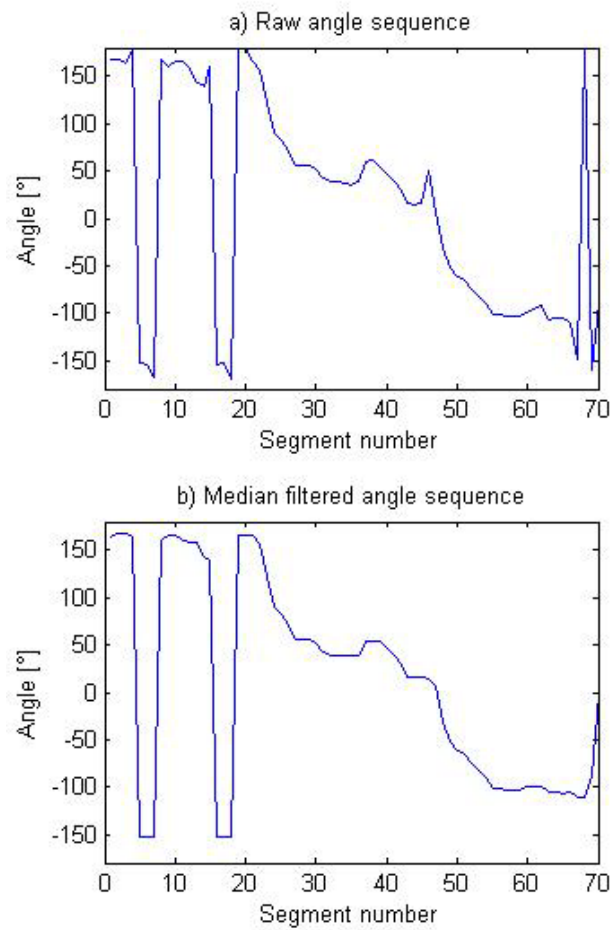


Fig. 5.12. Original and filtered angle sequences plot.

Using the classification based trajectory segmentation, 56 gestures were correctly recognized, 1 gesture was misclassified (4 strokes detected instead of 3) and 3 gestures were not recognized (the number of detected strokes exceeded 4).

The proposed algorithm runs in real-time on the specified hardware system, being able to successfully recognize the tested gestures.

Fig. 5.12.a) presents a high noise angle sequence obtained from a 3 stroke gesture, affected by hand trembling. The median filtered sequence is represented in Fig. 5.12.b).

The classify-and-filter method leads in this case to an erroneous 5 strokes detected gesture (Fig. 5.13.a)). The mean shift based algorithm (Fig. 5.13.b)) is still able to correctly interpret the gesture as the 3 strokes 180° , 45° , -90° .

Other gesture recognition solutions, [116], [117], rely on processing

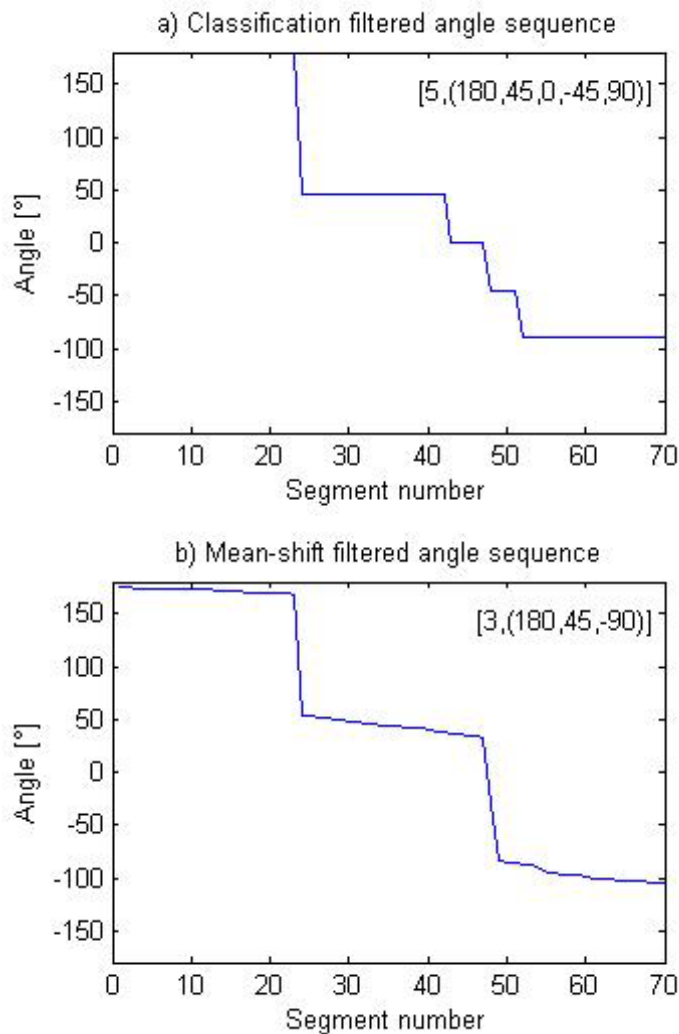


Fig. 5.13. Segmented trajectory and extracted features.

different successions of shape and positions of the hand and achieve relatively good recognition rates, at the price of significantly higher computational complexity.

The proposed solution has a reduced computational complexity, using a mean-shift based tracker, trajectory segmentation and trivial syntactic, trajectory based gesture recognition. The low computational cost allows the use of the algorithm in order to implement HMIs on relatively cheap systems. The user can use the system with minimal training. He is able to decide the gesture-action correspondences.

6. CONCLUSIONS AND FUTURE DEVELOPMENTS

In this thesis, new solutions related to the tracking of fingers in video sequences and the recognition of dynamic hand gestures based on trajectory are proposed. On the tracking related side, new solutions are proposed for tracking initialization and for the tracking process itself. The gesture recognition contributions include the thoughtful choice of the gesture alphabet and solutions for processing the trajectory, as well as for low computational complexity gesture recognition. Given the envisaged application range – HCI – all the newly proposed solutions aim at the real-time running capability, using inexpensive, largely available on the market, hardware systems. The results' evaluations indicate that the proposed solutions successfully achieve the goal assumed, without important sacrifices in terms of naturalness, precision and robustness.

The tracking initialization method presented in section 4.2 proved its reliability during the initial tests and was later successfully used for the initialization of the hand/finger tracking algorithm proposed in [3] and [4] and presented in sections 4.3 – 4.5. The method is easy to use from the user's point of view. While the multiple conditions imposed for initialization need low computational resources, they are able to provide a quick initialization and to prevent false triggering. The multi-cue approach allows the proposed initialization method to operate correctly under very different lighting conditions, with different backgrounds, without the need to readjust the settings of the thresholds. The advantage of a safe start is obtained at the price of a reduced flexibility regarding the initial position of the hand and of restrictions regarding the hand color uniformity (i.e. the user may not wear gloves, have extremely dirty hands etc.). The four detection criteria together with the time constraints imposed provide a user friendly initialization procedure. The time interval when the user must keep the hand in a given pose at a specific location is short enough in order not to be considered a drawback and it is long enough to significantly reduce the chances of false triggering.

The finger detection and tracking algorithm proposed in [3] and [4] is robust, computationally efficient and offers good results in challenging situations like frames affected by motion blur, fast movement, rapid scale changes, occlusions or the presence of other similar objects in the neighborhood of the target. The robustness is granted by the combination of multiple low level features to generate a sparse representation using line strips, followed by clustering and filtering of the new features. The use of line strip features also ensures computational efficiency, by significantly reducing the data to process. The proposed tracking algorithm is suitable for use both in pointing based HCIs (relying only on the fingertip positions) and dynamic gesture based HCIs (the finger angle and length can be used together with the location information). The low computational requirements make the proposed method also suitable for implementation with simpler processors.

The method for gesture definition and symbolic representation, together with the trajectory processing and segmentation algorithms presented in chapter 5, allow for a trivial, yet effective, gesture recognition implementation, based on matching the symbolic representation of the trajectory to the symbolic representation of the gesture prototypes. The low computational cost allows the use of the algorithm in order to implement HMIs on relatively cheap systems. Minimal training is required for the user in order to accommodate with the proposed HCI

framework. The associations between gestures and actions can either be predefined, requiring the user to learn the given correspondences, or the user can be allowed to choose its preferred gesture-action correspondences.

The next section of this chapter presents a summary of my original contributions presented in this thesis, while the last section indicates possible future developments based on the reported results.

6.1. Review of the original contributions

The main original contributions presented in this thesis can be classified, based on the target processing phase within a HCI system, into tracking related contributions and gesture recognition related contributions. Fig. 6.1 presents a hierarchical overview of the original contributions reported in the thesis. The tracking related contributions were presented in Chapter 4.

The **main tracking-related** personal contributions are:

- The design and implementation of a new semi-automatic finger tracking initialization method. The proposed method is presented in section 4.2 of the thesis and was previously published in [1]. Multiple features are used for finger detection. The initialization algorithm is implemented as a state machine. A special state is introduced for ensuring that the finger is detected as a consequence of the user's intention to perform dynamic gestures in front of the camera. Temporal and spatial conditions are tested to avoid triggering the tracker when it is not necessary.
- The design and implementation of a new robust finger tracking algorithm,

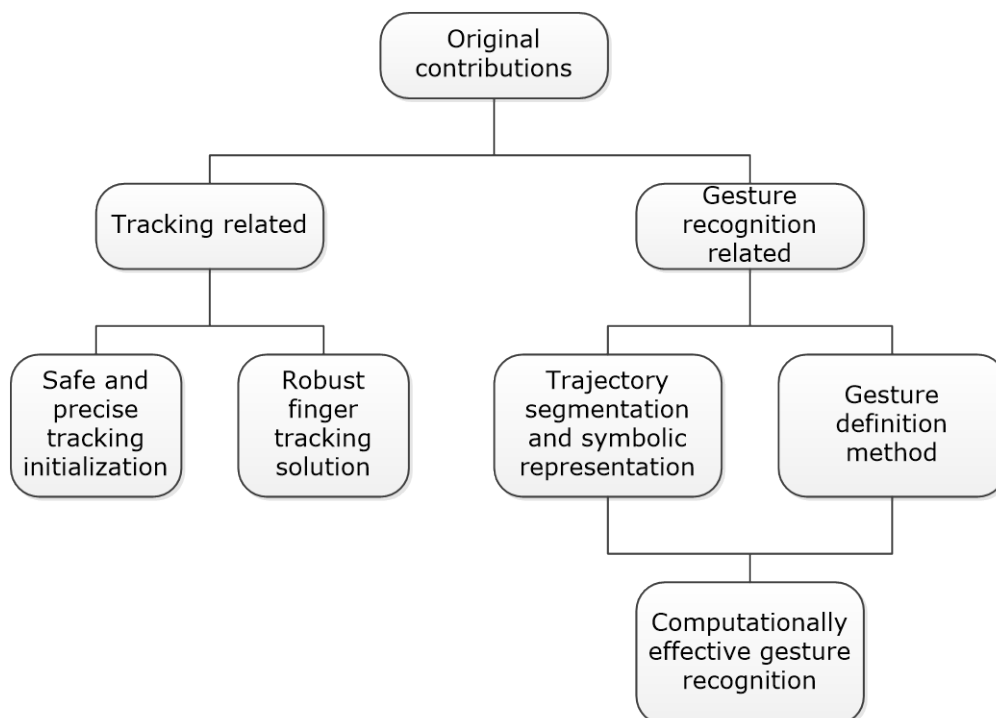


Fig. 6.1. Hierarchical overview of the main original contributions.

using multiple low, medium and high level features. The proposed solution is presented in sections 4.3 – 4.5 and was previously published in [3] and [4]. A multi-layer approach is used for processing the low level features, each processing layer reducing the amount of data to process for the next layer(s). Multiple aspects contribute to ensure the robustness: the use of multiple features, the ability to choose the medium level feature set which maximizes the probability of correct detection, successive filtering steps including weighted averaging and median filter, data clustering using 2D histogram-based analysis, motion prediction and the use of adaptive coefficients for updating the target parameters.

Other tracking-related personal contributions are:

- The experimental testing of performances of the tracking initialization method, with multiple prerecorded and live captured video sequences, under different lighting conditions with various backgrounds. The experimental results are discussed in section 4.6.1.
- The experimental testing of performances of the proposed tracking algorithm, with multiple video sequences acquired in different conditions, including multiple challenges for the tracker: motion blur, rapid scale changes, fast motion, presence of similar objects in the neighborhood, occlusions. The experimental results are presented in section 4.6.2.

The **main contributions** concerning the **gesture recognition** are:

- I proposed a gesture set containing gestures which are composed of quasi-linear consecutive strokes. The syntax used for the gesture representation allows for more than 3000 distinct gestures, using low numbers of consecutive strokes/gesture (2 – 4 strokes). From the huge amount of available gestures, a smaller group (containing the most easy to learn and to execute ones) can be selected, and different commands can be assigned to each of the selected gestures, for use in HCI. The proposed gesture definition method is presented in section 5.3.
- I proposed 2 new methods for hand trajectory processing and segmentation, which lead to symbolic representations of the trajectories and facilitate the gesture recognition: one method uses a statistical approach for filtering classified angle values, while the second method uses median filter and mean-shift clustering for trajectory segmentation. The proposed methods are presented in section 5.2 and were previously published in [5] and [6].
- Relying on the above claimed gesture definition method and the symbolic representation of gestures and trajectories, I proposed a very simple gesture recognition method implemented within a state machine. The gestures are recognized based on symbolic information matching between the trajectory and the gesture prototype. The proposed method is presented in section 5.3 and was previously published in [5] and [6].

Other gesture recognition related contribution:

- The evaluation of performances of the new methods proposed for trajectory segmentation and gesture recognition. The experimental results are presented in section 5.4.

6.2. Future research and development directions

- Integration of the proposed gesture recognition and tracking algorithms into a gesture based interface with a defined gesture alphabet. This also includes the evaluation of performances of gesture recognition using the trajectories

provided by the new tracking algorithm (which outperformed CAMSHIFT in many of the tested challenging situations).

- Further developments include definition of some recommended optimized gesture alphabets. An optimized alphabet involves the selection of gestures such that the general user can easily memorize and execute them. It is therefore desirable to select gestures composed of a reduced number of strokes, representing some significant geometric shapes and minimizing the possibility for the user to exceed the visual area of the camera while executing the gesture.
- An audio feedback to the user would also be useful, in order to allow it to focus its sight on its main activity (e.g. reading, viewing pictures, driving etc.). The audio feedback is straightforward to implement (i.e. play a pre-recorded audio track for each recognized gesture) and allows the user to know whether its gesture was correctly executed and correctly recognized by the system.
- Definition of a procedure for canceling the previous command is also necessary, to allow the user to cancel a command generated by a wrong execution of a gesture.
- The use of additional features (e.g. contours) for tracking, in order to extend the algorithm's robustness to non-fixed background environments.
- The extraction of additional semantical information for use in the tracking algorithm in order to handle long term occlusions.

BIBLIOGRAPHY

- [1] D. Popa, V. Gui, and M. Ottesteanu, "Semi-Automatic Hand/Finger Tracker Initialization for Gesture-Based Human Computer Interaction," in *Digital Information and Communication Technology and Its Applications*. vol. 166, H. Cherifi, J. Zain, and E. El-Qawasmeh, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 417-430.
- [2] V. Gui, G. Popa, P. Nisula, and V. Korhonen, "Finger Detection in Video Sequences Using a New Sparse Representation," *Acta Techn. Napoc.*, vol. 52, pp. 1-6, 2011.
- [3] D. Popa, V. Gui, and M. Ottesteanu, "Real-Time Finger Tracking with Improved Performance in Video Sequences with Motion Blur," in *International Conference on Telecommunications and Signal Processing - TSP 2014*, Berlin, 2014, pp. 715-720.
- [4] D. Popa, V. Gui, and M. Ottesteanu, "Real-Time Multi-Cue Finger Tracking for Human Computer Interaction," in *International Conference on Telecommunications and Signal Processing - TSP 2014*, Berlin, 2014, pp. 721-727.
- [5] D. Popa, G. Simion, V. Gui, and M. Ottesteanu, "Trajectory based hand gesture recognition," presented at the Proceedings of the 6th WSEAS international conference on Computational intelligence, man-machine systems and cybernetics, Tenerife, Canary Islands, Spain, 2007.
- [6] D. Popa, G. Simion, V. Gui, and M. Ottesteanu, "Real time trajectory based hand gesture recognition," *WSEAS Transactions on Information Science and Applications*, vol. 5, pp. 532-546, 2008.
- [7] A. C. Rencher and W. F. Christensen, *Methods of multivariate analysis* vol. 709: John Wiley & Sons, 2012.
- [8] D. W. Scott, *Multivariate Density Estimation*: Wiley & Sons, New York, 1992.
- [9] E. DiBenedetto, *Real Analysis*: Birkhäuser Boston, 2002.
- [10] B. Turlach, "Bandwidth Selection in Kernel Density Estimation: A Review," in *CORE and Institut de Statistique*, Louvain-la-Neuve, 1993, pp. 23-493.
- [11] A. Z. Zambom and R. Dias, "A review of kernel density estimation with applications to econometrics," *arXiv preprint arXiv:1212.2812*, 2012.
- [12] D. W. Scott and G. R. Terrell, "Biased and unbiased cross-validation in density estimation," *Journal of the American Statistical Association*, vol. 82, pp. 1131-1146, 1987.
- [13] K. Ziegler, "On local bootstrap bandwidth choice in kernel density estimation," *Statistics and Decisions-International Journal Stochastic Methods and Models*, vol. 24, pp. 291-302, 2006.
- [14] W. Härdle, M. Müller, S. Sperlich, and A. Werwatz, *Nonparametric and semiparametric models*: Springer, 2004.
- [15] R. van den Boomgaard and J. van de Weijer, "On the equivalence of local-mode finding, robust estimation and mean-shift analysis as used in early vision tasks," in *Pattern Recognition, International Conference on*, 2002, pp. 30927-30927.
- [16] H. Chen and P. Meer, "Robust regression with projection based m-estimators," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, pp. 878-885.

- [17] N. Fieller, "Statistical Modelling and Computing," *Course of the University of Sheffield, Department of Probability and Statistics*, 2004.
- [18] J. Astola, V. Katkovnik, and K. Egiazarian, *Local Approximation Techniques in Signal and Image Processing (SPIE Press Monograph Vol. PM157)*: SPIE-International Society for Optical Engineering, 2006.
- [19] J. Renno, J. Orwell, and G. A. Jones, "Learning surveillance tracking models for the self-calibrated ground plane," in *British Machine Vision Conference (BMVC)*, Cardiff, UK, 2002, pp. 1-10.
- [20] V. Kettner and R. Zabih, "Bayesian multi-camera surveillance," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1999, pp. 253-259.
- [21] B. Zhang, W. Tian, and Z. Jin, "Joint tracking algorithm using particle filter and mean shift with target model updating," *Chinese Optics Letters*, vol. 4, pp. 569-572, 2006/10/01 2006.
- [22] G. Bradski. (1998). *Computer Vision Face Tracking For Use in a Perceptual User Interface*. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.7673>
- [23] J. Newman, M. Wagner, M. Bauer, A. MacWilliams, T. Pintaric, D. Beyer, et al., "Ubiquitous tracking for augmented reality," in *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, 2004, pp. 192-201.
- [24] E. Murphy-Chutorian and M. M. Trivedi, "Head Pose Estimation and Augmented Reality Tracking: An Integrated System and Evaluation for Monitoring Driver Awareness," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, pp. 300-311, 2010.
- [25] W. Vieux, K. Schwerdt, and J. Crowley, "Face-Tracking and Coding for Video Compression," in *Computer Vision Systems*. vol. 1542, ed: Springer Berlin Heidelberg, 1999, pp. 151-161.
- [26] L. Nyugen, Y. Xu, and A. Roy-Chowdhury, "An Illumination Invariant 3D Model Based Tracking Algorithm, with Application in Video Compression," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 1213-1216.
- [27] M. Kobilarov, G. Sukhatme, J. Hyams, and P. Batavia, "People tracking and following with mobile robot using an omnidirectional camera and a laser," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2006, pp. 557-562.
- [28] S. Y. Chen, "Kalman Filter for Robot Vision: A Survey," *Industrial Electronics, IEEE Transactions on*, vol. 59, pp. 4409-4420, 2012.
- [29] G. Junfeng, L. Yupin, and T. Gyomei, "Real-Time Pedestrian Detection and Tracking at Nighttime for Driver-Assistance Systems," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, pp. 283-298, 2009.
- [30] A. DasGupta, A. George, S. L. Happy, and A. Routray, "A Vision-Based System for Monitoring the Loss of Attention in Automotive Drivers," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, pp. 1825-1838, 2013.
- [31] O. V. Olesen, R. R. Paulsen, L. Hojgaard, B. Roed, and R. Larsen, "Motion Tracking for Medical Imaging: A Nonvisible Structured Light Tracking Approach," *Medical Imaging, IEEE Transactions on*, vol. 31, pp. 79-87, 2012.
- [32] Y. Wang, B. Georgescu, T. Chen, W. Wu, P. Wang, X. Lu, et al., "Learning-Based Detection and Tracking in Medical Imaging: A Probabilistic Approach," in *Deformation Models*. vol. 7, M. González Hidalgo, A. Mir Torres, and J. Varona Gómez, Eds., ed: Springer Netherlands, 2013, pp. 209-235.

- [33] A. Cavallaro, O. Steiger, and T. Ebrahimi, "Tracking video objects in cluttered background," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, pp. 575-584, 2005.
- [34] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, pp. 564-577, 2003.
- [35] D. DeMenthon and R. Megret, "Spatio-temporal segmentation of video by hierarchical mean shift analysis," Computer Vision Laboratory, Center for Automation Research, University of Maryland 2002.
- [36] R. T. Collins, "Mean-shift blob tracking through scale space," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003, pp. II-234-40 vol.2.
- [37] P. Viola and W. M. Wells III, "Alignment by Maximization of Mutual Information," *International Journal of Computer Vision*, vol. 24, pp. 137-154, 1997/09/01 1997.
- [38] F. Dellaert and C. E. Thorpe, "Robust car tracking using Kalman filtering and Bayesian templates," in *Intelligent Transportation Systems*, 1998, pp. 72-83.
- [39] M. Isard and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, pp. 5-28, 1998/08/01 1998.
- [40] E. Maggio and A. Cavallaro, *Video Tracking: Theory and Practice*: Wiley, 2011.
- [41] S. Caifeng, W. Yucheng, T. Tieniu, and F. Ojardias, "Real time hand tracking by combining particle filtering and mean shift," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, 2004, pp. 669-674.
- [42] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 603-619, 2002.
- [43] Y. Rubner, J. Puzicha, C. Tomasi, and J. M. Buhmann, "Empirical Evaluation of Dissimilarity Measures for Color and Texture," *Computer Vision and Image Understanding*, vol. 84, pp. 25-43, 10// 2001.
- [44] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, pp. 174-188, 2002.
- [45] B. Stenger, P. R. Mendonça, and R. Cipolla, "Model-Based Hand Tracking Using an Unscented Kalman Filter," in *Proceedings of British Machine Vision Conference (BMVC)*, Manchester, UK, 2001, pp. 63-72.
- [46] M. Matilainen, J. Hannuksela, and L. Fan, "Finger Tracking for Gestural Interaction in Mobile Devices," in *Image Analysis*. vol. 7944, J.-K. Kämäräinen and M. Koskela, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 329-338.
- [47] X. Chai, K. Wang, L. Wei, and H. Wang, *Hand gesture tracking for wearable computing systems*, 2008.
- [48] W. Du and J. Piater, "Hand Modeling and Tracking for Video-Based Sign Language Recognition by Robust Principal Component Analysis," in *Trends and Topics in Computer Vision*. vol. 6553, K. Kutulakos, Ed., ed: Springer Berlin Heidelberg, 2012, pp. 273-285.
- [49] P. Song, S. Winkler, S. Gilani, and Z. Zhou, "Vision-Based Projected Tabletop Interface for Finger Interactions," in *Human-Computer Interaction*. vol. 4796, M. Lew, N. Sebe, T. Huang, and E. Bakker, Eds., ed: Springer Berlin Heidelberg, 2007, pp. 49-58.

-
- [50] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla, "Model-based hand tracking using a hierarchical Bayesian filter," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1372-1384, Sep 2006.
- [51] M. de La Gorce, D. J. Fleet, and N. Paragios, "Model-Based 3D Hand Pose Estimation from Monocular Video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1793-1805, Sep 2011.
- [52] C. Shan, T. Tan, and Y. Wei, "Real-time hand tracking using a mean shift embedded particle filter," *Pattern Recognition*, vol. 40, pp. 1958-1970, Jul 2007.
- [53] M. Kolsch and M. Turk, "Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on, 2004*, pp. 158-158.
- [54] V. Frati and D. Prattichizzo, "Using Kinect for hand tracking and rendering in wearable haptics," in *World Haptics Conference (WHC), 2011 IEEE, 2011*, pp. 317-321.
- [55] J. Molina, M. Escudero-Vinolo, A. Signoriello, M. Pardas, C. Ferran, J. Bescos, *et al.*, "Real-time user independent hand gesture recognition from time-of-flight camera video using static and dynamic models," *Machine Vision and Applications*, vol. 24, pp. 187-204, Jan 2013.
- [56] J.-H. An, J.-H. Min, and K.-S. Hong, "Finger Gesture Estimation for Mobile Device User Interface Using a Rear-Facing Camera," in *Future Information Technology*. vol. 185, J. Park, L. Yang, and C. Lee, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 230-237.
- [57] P. Zhigeng, L. Yang, Z. Mingmin, S. Chao, G. Kangde, T. Xing, *et al.*, "A real-time multi-cue hand tracking algorithm based on computer vision," in *Virtual Reality Conference (VR), 2010 IEEE, 2010*, pp. 219-222.
- [58] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern Recognition*, vol. 40, pp. 1106-1122, 2007.
- [59] S. Dai, M. Yang, Y. Wu, and A. K. Katsaggelos, "Tracking Motion-Blurred Targets in Video," in *Image Processing, 2006 IEEE International Conference on, 2006*, pp. 2389-2392.
- [60] W. Yi, L. Haibin, Y. Jingyi, L. Feng, M. Xue, and C. Erkang, "Blurred target tracking by Blur-driven Tracker," in *Computer Vision (ICCV), 2011 IEEE International Conference on, 2011*, pp. 1100-1107.
- [61] J. Lichtenauer, M. J. Reinders, and E. A. Hendriks, "A self-calibrating chrominance model applied to skin color detection," in *VISAPP (1), 2007*, pp. 115-120.
- [62] F. Tomaz, T. Candeias, and H. Shahbazkia, "Fast and accurate skin segmentation in color images," in *Computer and Robot Vision, 2004. Proceedings. First Canadian Conference on, 2004*, pp. 180-187.
- [63] M. Piccardi, "Background subtraction techniques: a review," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on, 2004*, pp. 3099-3104 vol.4.
- [64] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4-21, 5// 2014.
- [65] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: real-time tracking of the human body," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, pp. 780-785, 1997.

-
- [66] B. P. L. Lo and S. A. Velastin, "Automatic congestion detection system for underground platforms," in *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, 2001, pp. 158-161.
- [67] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, pp. 1337-1342, 2003.
- [68] S. Calderara, R. Melli, A. Prati, and R. Cucchiara, "Reliable background suppression for complex scenes," presented at the Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks, Santa Barbara, California, USA, 2006.
- [69] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*, 1999, p. 252 Vol. 2.
- [70] E. Hayman and J. O. Eklundh, "Statistical background subtraction for a mobile observer," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, pp. 67-74 vol.1.
- [71] D. S. Lee, J. J. Hull, and B. Erol, "A Bayesian framework for Gaussian mixture background modeling," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, 2003, pp. III-973-6 vol.2.
- [72] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, pp. 1151-1163, 2002.
- [73] B. Han, D. Comaniciu, Y. Zhu, and L. S. Davis, "Sequential kernel density approximation and its application to real-time visual tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, pp. 1186-1197, 2008.
- [74] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The Pixel-Based Adaptive Segmenter," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, 2012, pp. 38-43.
- [75] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-time imaging*, vol. 11, pp. 172-185, 2005.
- [76] M. Seki, T. Wada, H. Fujiwara, and K. Sumi, "Background subtraction based on cooccurrence of image variations," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003, pp. II-65-II-72 vol.2.
- [77] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 831-843, 2000.
- [78] C. Shengyong, Z. Jianhua, L. Youfu, and Z. Jianwei, "A Hierarchical Model Incorporating Segmented Regions and Pixel Descriptors for Video Background Subtraction," *Industrial Informatics, IEEE Transactions on*, vol. 8, pp. 118-127, 2012.
- [79] Z. Bineng, Y. Hongxun, S. Shiguang, C. Xilin, and G. Wen, "Hierarchical background subtraction using local pixel clustering," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 2008, pp. 1-4.
- [80] H. Sidenbladh, M. Black, and D. Fleet, "Stochastic Tracking of 3D Human Figures Using 2D Image Motion," in *Computer Vision — ECCV 2000*. vol. 1843, D. Vernon, Ed., ed: Springer Berlin Heidelberg, 2000, pp. 702-718.

-
- [81] A. Dargazany, A. Soleimani, and A. Ahmadyfard, "Multibandwidth Kernel-Based Object Tracking," *Advances in Artificial Intelligence*, vol. 2010, p. 15, 2010.
- [82] H. S. M. Shell, V. Arora, A. Dutta, and L. Behera, "Face feature tracking with automatic initialization and failure recovery," in *Cybernetics and Intelligent Systems (CIS), 2010 IEEE Conference on*, 2010, pp. 96-101.
- [83] J. Schmidt, "Automatic Initialization for Body Tracking: Using Appearance to Learn a Model for Tracking Human Upper Body Motions," in *3rd International Conference on Computer Vision Theory and Applications (VISAPP)*, Funchal, Madeira -Portugal, 2008, pp. 535-542.
- [84] X. Jiang, W. Ying, and A. Katsaggelos, "Part-based initialization for hand tracking," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, 2010, pp. 3257-3260.
- [85] T.-S. Wang, H.-Y. Shum, Y.-Q. Xu, and N.-N. Zheng, "Unsupervised Analysis of Human Gestures," in *Advances in Multimedia Information Processing — PCM 2001*. vol. 2195, H.-Y. Shum, M. Liao, and S.-F. Chang, Eds., ed: Springer Berlin Heidelberg, 2001, pp. 174-181.
- [86] Y. Wu and T. Huang, "Vision-Based Gesture Recognition: A Review," in *Gesture-Based Communication in Human-Computer Interaction*. vol. 1739, A. Braffort, R. Gherbi, S. Gibet, D. Teil, and J. Richardson, Eds., ed: Springer Berlin Heidelberg, 1999, pp. 103-115.
- [87] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, pp. 311-324, 2007.
- [88] P. Garg, N. Aggarwal, and S. Sofat, "Vision based hand gesture recognition," *World Academy of Science, Engineering & Technology - International Science Index*, vol. 3, pp. 821-826, 2009.
- [89] G. Murthy and R. Jadon, "A review of vision based hand gestures recognition," *International Journal of Information Technology and Knowledge Management*, vol. 2, pp. 405-410, 2009.
- [90] L. W. Campbell and A. F. Bobick, "Recognition of human body motion using phase space constraints," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*, 1995, pp. 624-630.
- [91] L. Doe-Hyung and H. Kwang-Seok, "Game interface using hand gesture recognition," in *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, 2010, pp. 1092-1097.
- [92] S. Lenman, L. Bretzner, and B. Thuresson, "Computer Vision Based Recognition of Hand Gestures for Human-Computer Interaction," Stockholm University ISSN 1403-0721, 2002.
- [93] G. Simion, V. Gui, M. Ottesteanu, D. Popa, and C. David, "Hand Edge Detection for Gesture Analysis in a Sparse Framework," *Scientific Bulletin of the "Politehnica" University of Timișoara, Romania, Transactions on Electronics and Communications*, vol. 53(67), pp. 155-160, 2008.
- [94] T. Moeslund and L. Nrgaard, "A Brief Overview of Hand Gestures used in Wearable Human Computer Interfaces," Aalborg University, Denmark ISSN 1601-3646, 2003.
- [95] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 677-695, 1997.

- [96] L. Hyeon-Kyu and J. H. Kim, "An HMM-based threshold model approach for gesture recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, pp. 961-973, 1999.
- [97] T. Starner and A. Pentland, "Real-time American Sign Language recognition from video using hidden Markov models," in *Computer Vision, 1995. Proceedings., International Symposium on*, 1995, pp. 265-270.
- [98] V. Pavlovic and J. M. Rehg, "Impact of Dynamic Model Learning on Classification of Human Motion," in *CVPR, 2000*, pp. 1788-1795.
- [99] J. Yamato, O. Jun, and K. Ishii, "Recognizing human action in time-sequential images using hidden Markov model," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, 1992, pp. 379-385.
- [100] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257-286, 1989.
- [101] M. Elmezain, A. Al-Hamadi, and B. Michaelis, "Hand trajectory-based gesture spotting and recognition using HMM," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, 2009, pp. 3577-3580.
- [102] R. Shrivastava, "A hidden Markov model based dynamic hand gesture recognition system using OpenCV," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, 2013, pp. 947-950.
- [103] Y. Zhaojun, A. Metallinou, E. Erzin, and S. Narayanan, "Analysis of interaction attitudes using data-driven hand gesture phrases," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 2014, pp. 699-703.
- [104] H.-I. Suk, B.-K. Sin, and S.-W. Lee, "Hand gesture recognition based on dynamic Bayesian network framework," *Pattern Recognition*, vol. 43, pp. 3059-3072, 9// 2010.
- [105] X. Shen, G. Hua, L. Williams, and Y. Wu, "Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields," *Image and Vision Computing*, vol. 30, pp. 227-235, 3// 2012.
- [106] P. Hong, M. Turk, and T. S. Huang, "Gesture modeling and recognition using finite state machines," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, 2000, pp. 410-415.
- [107] R. Verma and A. Dev, "Vision based hand gesture recognition using finite state machines and fuzzy logic," in *Ultra Modern Telecommunications & Workshops, 2009. ICUMT '09. International Conference on*, 2009, pp. 1-6.
- [108] L. Shiguo, H. Wei, and W. Kai, "Automatic user state recognition for hand gesture based low-cost television control system," *Consumer Electronics, IEEE Transactions on*, vol. 60, pp. 107-115, 2014.
- [109] M. Yeasin and S. Chaudhuri, "Visual understanding of dynamic hand gestures," *Pattern Recognition*, vol. 33, pp. 1805-1817, 2000.
- [110] S. Mu-Chun, "A fuzzy rule-based approach to spatio-temporal hand gesture recognition," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 30, pp. 276-281, 2000.
- [111] W. Tan, C. Wu, S. Zhao, and L. Jiang, "Dynamic hand gesture recognition using motion trajectories and key frames," in *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, 2010, pp. 163-167.
- [112] Y. Araga, M. Shirabayashi, K. Kaida, and H. Hikawa, "Real time gesture recognition system using posture classifier and Jordan recurrent neural network," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012, pp. 1-8.

-
- [113] J. Wang, B. Thiesson, Y. Xu, and M. Cohen, "Image and Video Segmentation by Anisotropic Kernel Mean Shift," in *Computer Vision - ECCV 2004*, vol. 3022, T. Pajdla and J. Matas, Eds., ed: Springer Berlin Heidelberg, 2004, pp. 238-249.
- [114] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*, 1998, pp. 839-846.
- [115] D. Barash, "Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 844-847, 2002.
- [116] 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*, 2002, pp. 423-428.
- [117] J.-H. Shin, J.-S. Lee, S.-K. Kil, D.-F. Shen, J.-G. Ryu, E.-H. Lee, *et al.*, "Hand region extraction and gesture recognition using entropy analysis," *IJCSNS International Journal of Computer Science and Network Security*, vol. 6, pp. 216-222, 2006.