Universitat
Autònoma
de Barcelona

Model–Based Visual Localisation of Contours and Vehicles

A dissertation submitted by **Daniel Ponsa Mussarra** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor en Informática**.

Bellaterra, 14th May 2007

Director: **Dr. Antonio López Peña**
Universitat Autònoma de Barcelona
Dept. de Ciències de la Computació & Computer Vision Center
Co-director: **Dr. Xavier Roca i Marvà**
Universitat Autònoma de Barcelona
Dept. de Ciències de la Computació & Computer Vision Center

Centre de Visió
per Computador

This document was typeset by the author using LaTeX 2$_\varepsilon$.

The research described in this book was carried out at the Computer Vision Center, Universitat Autònoma de Barcelona.

ISBN 978-84-935251-3-2

Printed by Ediciones Gráficas Rey, S.L.

a la meva família

You can’t always get what you want,
But if you try sometime, yeah,
You just might find you get what you need!

Jagger & Richards

Aquest treball ha estat finançat parcialment pel Ministeri espanyol d'Educació i Ciència amb els projectes TRA2004-06702/AUT i TRA2007-62526/AUT.

Agraïments

Aquesta secció de la tesi representa per a mi el punt d'arribada d'un trajecte que vaig decidir emprendre fa vora 10 anys. És obvi que això és massa temps per escriure una tesi, i com canta el filòsof de guàrdia Portet "creix la sospita de que hauria d'existir un camí més curt per arribar aquí", però així han anat les coses. En ocasions m'ha costat trobar el camí, en d'altres m'ha convingut fer alguna parada, i en alguns moments he arribat a creuaments on no sempre he escollit la millor ruta. Malgrat tot això, i això ja em ve de ben petit, encara que tard al final he enfilat el camí que m'ha dut fins a escriure aquestes ratlles.

Probablement mai m'hagués plantejat començar una tesi doctoral si no hagués gaudit tant en el projecte de final de carrera. El meu agraïment doncs al departament d'enginyeria electrònica que tan bé em va acollir en la realització del projecte, en especial al Dr. Francesc Pérez Murano per tot el temps que em va dedicar i pel molt que em va ajudar, i als companys de laboratori Gabriel Abadal, Xevi Borrisé i Joan Massó.

El meu reconeixement a en Jordi Vitrià i Xavier Roca, per fer possible que aquesta tesi fes les primeres passes, i el meu agraïment superlatiu a l'Antonio López per la seva tutela, sense la qual aquesta tesi no hauria arribat a bon port. Mai t'estaré prou agraït de tot el que m'has ajudat. Tot un luxe poder treballar al teu costat. Un agraïment especial també a en Joan Serrat, pels valuosos comentaris i suggeriments aportats en la revisió final d'aquesta tesi.

Vull agrair també a en Juanjo Villanueva la tasca desenvolupada al capdavant del Centre de Visió per Computador, així com el tracte que m'ha dispensat al llarg de tots aquest anys. Un record especial també als companys amb qui vaig començar la tesi: l'Albert Pujol, en Xavi Varona, i molt especialment a l'Andrés Solé, científic de grandíssim talent i gran amic, que la dissort va apartar-lo del nostre costat. Em vas ensenyar moltíssim. Agraeixo també a en Robert Benavente l'amistat de tots aquest anys, i l'ajuda aportada en aquests darrers mesos, en els que ambdós hem coincidit en la tramitació final de les nostres respectives tesis.

I would like to thank Steve Maybank and the rest of the Computational Vision Group for their warm reception and the work we shared during my stay at Reading University.

Un agraïment especial als companys del grup de recerca en sistemes avançats d'assistència a la conducció, amb qui compartir recerca és tot un plaer: Antonio López, Joan Serrat, Ángel Sappa, David Gerónimo, Carme Julià, Felipe Lumbreras, José Manuel Álvarez i Ferran Diego.

Ich möchte Thorsten Graf und seinen Mitarbeitern in der Volkswagen AG für das Vertrauen danken, das sie in unseren ADAS Arbeitskreis gesetzt haben. Vielen Dank für die gute Zusammenarbeit!

El meu agraïment als companys del despatx, del grup de recerca, i del laboratori que generosament han posat els seus ordinadors a la meva disposició en el tram final d'aquesta tesi, permetent-me així finalitzar a temps el treball experimental d'aquesta tesi: Robert Benavente, Bogdan Raducanu, Fadi Dornaika, Carme Julià, David Gerónimo, Antonio López, Ferran Diego, Felipe Lumbreras, Cristina Cañero, Eva Costa, Anton Cervantes i Enric Sala. Així mateix, el meu agraïment a tots els integrants del Centre que he tingut la sort de conèixer al llarg de tots aquest anys. Gràcies per les vostres mostres d'afecte, i per l'ajuda desinteressada que tantíssimes vegades m'heu ofert, ja sigueu companys de recerca, de docència, de projecte industrial, del laboratori, d'administració, del servei de tractament d'imatges, de la biblioteca, de manteniment,... Disculpeu si no us veieu citats en aquest apartat, però la llista de noms seria llarguíssima, tot un repte per la meva limitada memòria.

No em vull oblidar de donar les gràcies als meus amics: als de la colla de Monistrol de Calders, de qui sempre m'he pogut refiar des que tinc ús de raó, i als que he tingut la sort d'anar coneixent al llarg dels anys, molts d'ells trobats pels estranys viaranys on m'ha portat aquesta tesi. Espero ara tenir més temps per poder gaudir de la vostra amistat.

El meu més sincer agraïment als meus pares Pius i Anna, als meus germans Fina, Pere, Jaume, Fèlix, Jordi i Eva, així com a tots els meus tiets, nebots, cunyats i cosins. Gràcies pel vostre suport i ànims, i per la confiança i paciència que heu tingut amb mi en tots aquests anys de tesi. A la meva família de Viladecavalls: la María, el Rafael i la Silvia, per tot el vostre afecte i recolzament. Finalment, un agraïment molt especial a la Cristina, la persona que ha viscut aquesta tesi més de prop, i a qui li han tocat tots el inconvenients de ser una *doctoranda passiva*. Gràcies pel teu suport incondicional, i per tots els anys que m'has regalat estant al meu costat.

Daniel Ponsa Mussarra

Abstract

This thesis focuses the analysis of video sequences, applying model-based techniques for extracting quantitative information. In particular, we make several proposals in two application areas: shape tracking based on contour models, and detection and tracking of vehicles in images acquired by a camera installed on a mobile platform.

The work devoted to shape tracking follows the paradigm of active contours, from which we present a review of the existent approaches. First, we measure the performance of the most common algorithms (Kalman based filters and particle filters), and then we evaluate its implementation aspects trough an extensive experimental study, where several synthetic sequences are considered, distorted with different degrees of noise. Thus, we determine the best way to implement in practice these classical tracking algorithms, and we identify its benefits and drawbacks.

Next, the work is oriented towards the improvement of contour tracking algorithms based on particle filters. These algorithms reach good results provided that the number of particles is high enough, but unfortunately the required number of particles grows exponentially with the number of parameters to be estimated. Therefore, and in the context of contour tracking, we present three variants of the classical particle filter, corresponding to three new strategies to deal with this problem. First, we propose to improve the contour tracking by propagating more accurately the particles from one image to the next one. This is done by using a linear approximation of the optimal propagation function. The second proposed strategy is based in estimating part of the parameters analytically. Thus, we aim to do a more productive use of the particles, reducing the amount of model parameters that must be estimated through them. The third proposed method aims to exploit the fact that, in contour tracking applications, the parameters related to the rigid transform can be estimated accurately enough independently from the local deformation presented by the contour. This is used to perform a better propagation of the particles, concentrating them more densely in the zone where the tracked contour is located. These three proposals are validated extensively in sequences with different noise levels, on which the reached improvement is evaluated.

After this study, we propose to deal directly with the origin of the previous problem by reducing the number of parameters to be estimated in order to follow a given shape of interest. To reach that, we propose to model the shape using multiple models, where each one requires a lower quantity of parameters than when using a unique model. We propose a new method to learn these models from a training set, and a new algorithm to use the obtained models for tracking the contours. The experimental results certify the validity of this proposal.

Finally, the thesis focuses on the development of a system for the detection and tracking of vehicles. The proposals include: a vehicle detection module, a module devoted to the determination of the three-dimensional position and velocity of the detected vehicles, and a tracking module for updating the location of vehicles on the road in a precise and efficient manner. Several original contributions are done in these three subjects, and their performance is empirically evaluated.

Resum

El treball d'aquesta tesi es centra en l'anàlisi de seqüències de vídeo, aplicant tècniques basades en models per extreure'n informació quantitativa. En concret, es realitzen diferents propostes en dues àrees d'aplicació: el seguiment de formes basat en models de contorns, i la detecció i seguiment de vehicles en imatges proveïdes per una camera installada en una plataforma mòbil.

El treball dedicat al seguiment de formes s'enquadra en el paradigma de contorns actius, del qual presentem una revisió de les diferents propostes existents. En primer lloc, mesurem el rendiment obtingut pels algorismes de seguiment més comuns (filtres basats en Kalman i filtres de partícules), i en segon lloc avaluem diferents aspectes de la seva implementació en un extens treball experimental on es consideren múltiples seqüències sintètiques, distorsionades amb diferents graus de soroll. Així, mitjançant aquest estudi determinem la millor manera d'implementar a la pràctica els algorismes de seguiment clàssics, i identifiquem els seus pros i contres.

Seguidament, el treball s'orienta cap a la millora dels algoritmes de seguiment de contorns basats en filtres de partícules. Aquest algorismes aconsegueixen bons resultats sempre que el número de partícules utilitzades sigui suficient, però malauradament la quantitat de partícules requerides creix exponencialment amb el número de paràmetres a estimar. Per tant, i en el context del seguiment de contorns, presentem tres variants del filtre de partícules clàssic, corresponents a tres noves estratègies per tractar aquest problema. En primer lloc, proposem millorar el seguiment de contorns mirant de propagar més acuradament les partícules emprades per l'algorisme d'una imatge a la següent. Això ho duem a terme utilitzant una aproximació lineal de la funció de propagació òptima. La segona estratègia proposada es basa en estimar part dels paràmetres de manera analítica. Així, es pretén fer un ús més productiu de les partícules emprades, reduint la part dels paràmetres del model que s'han d'estimar amb elles. El tercer mètode proposat té com a objectiu treure profit del fet de que, en aplicacions de seguiment de contorns, sovint els paràmetres relatius a la transformació rígida es poden estimar prou acuradament independentment de la deformació local que el contorn presenti. Això s'utilitza per realitzar una millor propagació de les partícules, concentrant-les més densament en la zona on el contorn seguit es troba. Aquestes tres propostes es validen de manera extensiva en seqüències amb diferents nivells de soroll, amb les que es mesura la millora aconseguida.

A continuació proposem tractar directament l'origen del problema anterior mitjanҫant la reducció del nombre de paràmetres a estimar per tal de seguir una determinada forma d'interès. Per aconseguir això, proposem modelar aquesta forma usant múltiples models, on cadascun requereix una quantitat de paràmetres inferior a la requerida per un únic model. Es proposa un nou mètode per aprendre aquests models a partir d'un conjunt d'entrenament, així com un nou algorisme per emprar-los en el seguiment dels contorns. Els resultats experimentals certifiquen la validesa d'aquesta proposta.

Finalment, la tesi es centra en el desenvolupament d'un sistema de detecció i seguiment de vehicles. Les propostes realitzades comprenen: un mòdul de detecció de vehicles, un mòdul dedicat a determinar la posició i velocitat 3D dels vehicles detectats, i un mòdul de seguiment per actualitzar la localització dels vehicles a la carretera de manera precisa i eficient. Es realitzen diverses aportacions originals en aquests tres temes, i se n'avalua el rendiment.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

In the last 20 years, the analysis of video sequences has become a topic of increasing interest in the area of computer vision. The rise of computational power experienced by personal computers has allowed researchers to address applications in this field, which just a few years ago required the availability of very expensive hardware, designed many times specifically for a desired task. As a result of that, nowadays one can find low-cost products in the market whose functionality is based on a rough real-time sequence analysis, like camera surveillance systems with improved facilities or electronic devices with camera-based interfaces, amongst others. These pioneer products are just the prelude of many camera-based systems that for sure we will deal with in the near future, as video sequence analysis is expected to play a very important role in human-machine communication. Many other fields require the development of systems able to extract key information from sequences of frames. In medical imaging, the quality of imaging devices has improved spectacularly in the last years, providing very helpful visual representations of the human body. Nowadays physicians not only diagnose by analysing images or volumetric data, but sequences of them too, showing for instance the temporal behaviour of a given 3D body part. Computer vision techniques can help in quantifying the dynamics of observed elements, or in preselecting the information of medical interest contained in long sequences. In robotics the analysis of image sequences is also a main topic of research, as one of the basic sensors of robotic systems dealing in dynamic environments are calibrated multi–camera systems. A sector like the automotive industry promotes also many research efforts onto this topic, in the development of the so–called Advanced Driver Assistance Systems (ADAS). ADAS has the goal of increasing the safety of vehicle occupants, by providing vehicles with situational awareness. Sequences supplied by camera systems provide a rich description of the vehicle surroundings, making possible applications which demand to ascertain the state of the driver and passengers or the characterisation of the external environment (the road, other vehicles, traffic signs, etc.).

This growing interest in sequence analysis from such an heterogeneous set of fields has been our main motivation to pose this thesis. In general, applications may require to qualitatively extract information from frames (for instance, detecting a particular dynamical event), or quantitatively (for instance, the precise characterisation of the attributes of a given object along time). This thesis focuses on this second type of application.

The computation of quantitative information from sequences can be formulated as the estimation of the state $\mathbf{x}$ of an object or attribute of interest (target) at instant t, from the evidence $\mathbf{y}$ obtained from frames. A key point in this process is the procedure to analyse video frames in order to localise and extract the information of interest $\mathbf{y}$ (i.e., the observation process). Classical strategies are based on a deterministic search around the area where the target is expected, looking for a zone matching a given object description (i.e., a target model). In applications where targets are observed uncluttered on images, observations obtained may be sufficiently accurate to solve a given task, leading to *data–driven* solutions. These cases are typically implemented using variational methods, which accurately determine $\mathbf{y}$ by minimising iteratively a cost function measuring the disparity between the image region inspected and the target model. Examples of variational methods are found in the localisation of targets using pattern [54], histogram [22, 29], and contour [66] models. However, in many applications is not possible to obtain accurate information from frames, due to restrictions on the computational resources available for this task, or due to the noise and clutter distorting them. In these cases, the confidence given to $\mathbf{y}$ has to be modulated to properly estimate $\mathbf{x}$, and a way to do that is by taking advantage of a priori information of the problem being solved: the knowledge on the observation process and the noise distorting it (i.e., the observation model), and the knowledge on the expected evolution of $\mathbf{x}$ along frames (i.e., the system model). These different sources of information have to be properly combined, which is the purpose of the methods developed in estimation theory. In this thesis the problem of video sequence analysis is addressed within this theoretical framework. Concerning the estimation of a target state $\mathbf{x}$ at a given instant t (i.e., $\mathbf{x}_t$), three different tasks are defined, depending on the evidence $\mathbf{y}$ available at this instant:

prediction: when $\mathbf{x}_t$ is estimated from evidence $\mathbf{y}$ up to $t-1$,

filtering: when $\mathbf{x}_t$ is estimated from evidence $\mathbf{y}$ up to t,

smoothing: when $\mathbf{x}_t$ is estimated from evidence $\mathbf{y}$ up to T, with $T > t$.

This thesis focuses mainly on filtering, the task to be solved in *tracking* applications, which demands the real-time estimation of a target state along time. The main modules to be solved by a video–based tracking system are sketched in Figure 1.1. Basically, a detection module has to inspect acquired frames in order to ascertain the presence of targets of interest. Then, the state of detected targets has to be initially estimated, to then start a *filtering* process (also known as *tracking algorithm* or *tracker*), which will update this state in successive frames, combining efficiently the information extracted from frames with the a priori knowledge provided by the system and observation models.

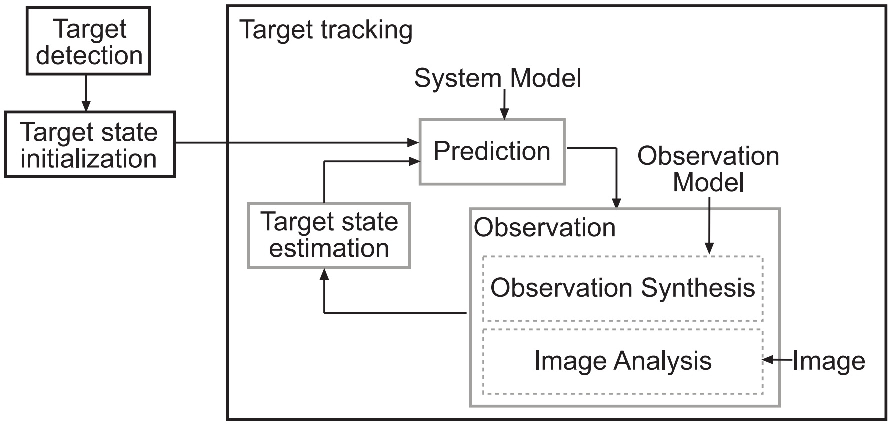


Figure 1.1: Model–Based Tracking Framework.

Taking as reference the model–based tracking framework in Figure 1.1, the main contributions of this thesis are on:

- novel proposals for the target tracking module in the context of contour tracking applications;

- a proposal of a whole tracking system to detect and track vehicles in sequences acquired by a single camera mounted in a mobile platform.

The first part of the thesis has been focused on contour tracking. Contours can be a very informative visual cue of the state of objects, providing sufficient information to solve many practical problems. From a research point of view, this problem is inherently interesting, because contour tracking is very challenging in general. A contour can change non–linearly along frames, and in a very sudden way, being its tracking a difficult task. Hence, the first goal of the thesis has been the proposal of new strategies to track contours, with the aim to overcome the performance of previous approaches.

In the second part of the thesis, the work has been oriented to the development of a *complete* vehicle tracking system. That is, a system able to automatically detect vehicles in images, initialise their 3D state (i.e., their road location and velocity), and update this state efficiently along time. Such a system is of special interest for the automotive industry, because it can help to develop safer vehicles[1]. This thesis contributes with a new proposal for vehicle detection and tracking using a single camera.

Following sections describe in major detail the work developed concerning these two objectives.

[1] Usually, the major threads for a driver are the other vehicles on the road ahead.

1.1 Goals in Contour Tracking

Contour tracking has been studied in this thesis adopting the *active contours* approach, mainly developed by British research groups at the universities of Oxford, Leeds and Manchester. The interested reader may wish to consult [18, 31, 11] for a detailed and complete introduction on this topic. Essentially, the active contours approach is based on defining a generative model of the outline of a given object of interest, and then using an estimation algorithm to determine which model parameters better fit the observations in the frames of a video sequence. Since its formulation, different active contour implementations have been proposed. A first goal of this thesis has been to review and evaluate the main contributions on this technique, analysing the effect on tracking performance of:

- the methods proposed to extract contour observations from frames;
- the estimation methods used to process them.

A comparative study of the different implementation options is presented, evaluating their performance in sequences with different levels of noise. Results obtained reflect the pros and cons of the different alternatives, giving relevant clues about which options to considered given a particular tracking problem.

Next topic in this part of the thesis has faced the following problem. In many contour tracking applications, when the target to be tracked can simultaneously display rigid and non-rigid transformations along time, the high dimensionality of the corresponding contour state can be a handicap. If a particle–based estimation method is used, it is observed that to achieve a desired tracking performance, the number of required particles increases exponentially with the state dimensionality. In some practical systems, this amount of particles will be prohibitive. In this thesis we propose different strategies to balance the effect of state dimension in particle–based contour trackers, which at the time of being developed were novel in this application context:

- the adaptation of the so–called Unscented Kalman Particle Filter to this problem;
- the estimation of part of the contour state analytically, by applying the Rao–Blackwellization technique;
- the estimation of the contour state using the Partitioned Sampling technique, taking advantage of a novel proposed method to estimate the rigid transformation of a contour decoupled from the non–rigid ones.

The performance of all these techniques is evaluated, quantifying their capacity to overcome *traditional* algorithms.

Next, a solution to this state dimensionality problem is proposed from a different perspective: reducing the dimensionality of the estimated state by reducing the number of parameters required to model the non–rigid transformations of contours. Our proposal is based on replacing the single shape model commonly used to represent this transformations, by a collection of multiple shape models. The idea is that, as each one of these model will account just for a subspace of the feasible contour non–rigid transformations, each one will require less parameters than the initial single shape model. A novel method is proposed to generate such a collection of models from training data, as well as an algorithm to manage all of them for contour tracking purposes. The validity of the proposed method is experimentally evaluated, showing its better performance with respect to the single model approach.

1.2 Goals in Vehicle Tracking

In this part of the thesis, the goal is the design and implementation of a complete visual tracking system, aimed at the detection and 3D tracking of vehicles in images acquired from a single camera mounted in a mobile platform. Topics obviated in the previous work concerning contour tracking now have to be explicitly considered: automatic target detection, tracking state initialisation, and tracking activity control (i.e., the identification of misstrack situations, the control of the occlusions between targets, etc.). The development of such a tracking system is a very challenging task.

Focusing just on the target detection task, difficulties arises due to the heterogeneity in the appearance of the objects of interest (with the term vehicle we refer to cars, vans, trucks, buses, etc., excluding, however, motorbikes and similar) and the wide variability of possible illumination conditions of the environment (due to weather conditions, time of the day, shadows, etc.). To fulfil this task, we propose the use of an state of the art machine learning technique (boosting) to generate from training data, a classifier devoted to judge the presence of vehicles in image subregions. The advantage of this approach is that, as long as the training data includes sufficient significant examples, the generated classifier is robust to the high variety of target appearances. To speed up vehicle detection, in practice a common approach is learning several classifiers, evaluating them on images in cascade. In that way, image regions are classified faster, because most regions in an image do not contain vehicles, and are identified without evaluating the whole cascade of classifiers. To further increase the efficiency in vehicle detection, the thesis proposes two novel strategies: the a posteriori *optimisation* of the cascade of classifiers, and its lazy evaluation on frames. A quantitative evaluation of the performance of the proposed detector is also provided.

Concerning the initial estimation of the 3D state of detected vehicles (i.e., its 3D location and velocity), a method combining projective geometry with multiple hypothesis tracking is proposed. First, the output of the vehicle detector is processed to estimate the 3D road region where detected vehicles may be present. Extracting 3D information from just 2D images is an ill–posed problem, and assumptions have to be imposed on the observed scene to fulfil this task. We propose to assume that the observed road conforms to a flat surface (a realistic assumptions in most cases), and that the width and orientation of the observed vehicle is known. Although the hypothesis on the vehicle will be wrong in some cases, the violation of these assumptions will provoke a systematic error in the vehicle localisations, which assures spatio–temporal coherence between the 3D road coordinates estimated for a vehicle in successive frames. This is very interesting because makes easier the construction of vehicle trajectories from them, which is useful for estimating vehicle dynamics, as well as for filtering out false vehicle detections, since they usually are spurious and no trajectories can be build from them. From the constructed trajectories, the initial state of a vehicle tracker is ascertained. This state can be inaccurate, due to wrong assumptions of the vehicle width and orientation. However, if at some point there is additional information to correct these assumptions, we provide with expressions to properly amend the initial state computed.

Finally, we propose a multiple vehicle 3D tracking algorithm. The 3D location and velocity of all detected vehicles is maintained in a single state vector, which includes also parameters concerning the movement of the vehicle holding the camera. This is done in that way to model more precisely the changes observed in successive frames. From the state parameters and their assumed dynamics, the image regions where vehicles should be observed are predicted, localising them more accurately than the vehicle detection module can do, and therefore, estimating more reliably their state along time. After each iteration cycle, a control process checks the estimated state, in order to readjust it if a conflictive situation is observed (i.e., eliminates targets which are occluded, or no longer detected, etc.).

1.3 Thesis Outline

The organisation of the thesis is as follows. Chapter 2 briefly describes the active contours approach to contour tracking. First, the usual methods applied to generate a 2D shape model are detailed, as well as the expressions to describe their dynamic behaviour. Then the Bayesian formulation of the contour tracking problem is presented, and the main proposals to solve it reviewed (i.e., Kalman–based and particle–based estimation methods). Next, different proposals are presented to process frames, in order to extract observations of the contour to be tracked. Then, the chapter focuses on evaluating the proposals reviewed in a case–study application, in order to discern the best way to extract observations from images, and the accuracy achieved by different estimation algorithms (Kalman Filters, Extended Kalman Filters, Unscented Kalman Filters, and Particle Filters). Results provided show the performance achieved by the different proposals on sequences distorted by different levels of noise.

Chapter 3 starts describing a non–linear shape model to represent simultaneously the rigid and non–rigid transformations that a contour may present along time. Then, the use of this model by the classical algorithms presented in the previous chapter is discussed, remarking the performance degradation observed in Particle Filters due to the higher dimensionality of the contour model. We propose then the novel application of three different estimation methods, with the aim of improving the performance of Particle Filters in general contour tracking applications: the Unscented Kalman

Particle Filter, the Rao–Blackwellized Particle Filter, and the Partitioned Sampling algorithm. The last one of these methods is based on a novel approach that we propose to estimate the rigid transformation of a contour, decoupled from its non–rigid one. The performance of each method under different noisy situations is quantified, finally comparing their achieved results to identify their strong and weak points.

Chapter 4 focuses on improving the performance of Particle filters by modelling the outline of a target using multiple shape models. First, the problem of poor specificity of single shape models is introduced, justifying the need for a constraint model to delimit the parameterisations that can be applied on them, in order to synthesise only *valid* shapes. Then, we propose a novel method to learn this constraint model from training data, based on the unsupervised parameterisation of Gaussian mixture models. Using this constraint model, we present an original methodology to replace the initial shape model by an ensemble of simple linear shape models. This multiple model approach to shape representation is shown in a case–study to overcome the single model approach, in terms of the Bayesian Information Criterion. Finally, the use of multiple models in tracking applications is detailed. The interaction along time between the different models is modelled as a Jump Markov System, and a novel multiple model contour tracking algorithm is presented which, in a case–study, is shown to overcome the performance of an equivalent single model tracking algorithm.

Chapter 5 focuses on the detection and 3D tracking of vehicles from images taken by a monocular vision system mounted in a mobile platform. First the detection of vehicles on images is studied, proposing a classifier–based approach to perform this task. The Adaboost technique is applied to learn a classifier from a training set, where examples of the appearance of vehicles and non–vehicles under different illumination conditions are provided. To achieve a desired detection accuracy, different classifiers are learnt and arranged in a cascade. We propose then two strategies to reduce the computations required to apply the generated cascade of classifiers on images. Then, the chapter focuses on the problem of estimating the 3D state (3D location and velocity) of detected vehicles. We propose a method based on combining a model of camera projection, assumptions on the road and observed vehicles, and the Unscented Transform mechanism, devoted to estimate the road location corresponding to vehicle detections. The spatio–temporal coherence of these locations is then exploited, in order to discard spurious false detections generated by the detector, and estimate the 3D velocity of vehicles. This task is solved using a Multiple Hypothesis Tracking Algorithm, whose performance is evaluated experimentally using synthetic detections, studying the effect of common sources of error. Finally, the chapter describes a multiple target tracking algorithm to efficiently update the state of detected vehicles along time. The Unscented Kalman Filter has been proposed to carry out this task.

Chapter 6 draws the conclusions of this thesis, and suggests new directions for further research.

Several appendices complement the information given in the former chapters, providing details on some topics that for the sake of readability were omitted. Appendix A provides insight in the modelisation of state dynamics using auto–regressive models, and describes two novel methods that we propose to parameterise respectively first and second order dynamic models, in order to achieve a desired dynamical behaviour. Appendix B derives the Kalman Filter equations from a Bayesian point of view, complementing in that way the one usually available in estimation theory literature. From this perspective, the relation between Kalman and particle filters is clearer, which can be helpful for readers unfamiliar with these topics. Appendix C presents some basic concepts concerning Monte Carlo methods, which are the basis of the application of sampling techniques in the resolution of state estimation problems. Finally, Appendix D detail the methodology followed to quantify objectively the tracking performance of the different methods studied in the thesis.

Chapter 2

Model–Based Visual Contour Tracking

This chapter focuses on work developed in the last two decades on the use of contour models for object tracking. The proposals reviewed are usually denoted as Active Contours or Active Shape Models, and their basic building blocks concern the following topics:

- the modelling of a parametric contour and the space of its feasible variations;
- the modelling of the dynamic behaviour of a contour, describing how it is expected to change along time;
- the Bayesian approach to combine both shape and dynamic models to estimate the contour state from observations in video frames;
- the measurement process to efficiently analyse video sequences and extract the information required for contour estimation.

All these topics are reviewed in this chapter, focusing on the concrete proposals analysed in this thesis. Taking the general model–based tracking framework as a kind of map, Figure 2.1 sketches the topics analysed in the following sections. Those readers who are already familiar with these topics might wish to skip from section 2.1 to 2.4, and proceed directly to section 2.5, in which we contribute with an in-depth evaluation of the reviewed proposals in a case–study application.

2.1 2D Shape Modelling

Any attempt of robustly modelling an object implies finding a representation that deals in some way with its feasible variations. From a pattern recognition point of view, the generation of a model can be viewed as determining an object representation that generalises the whole spectra of object appearances, taking care of still remaining discriminant (that is, recognise only the desired object). The objective could be

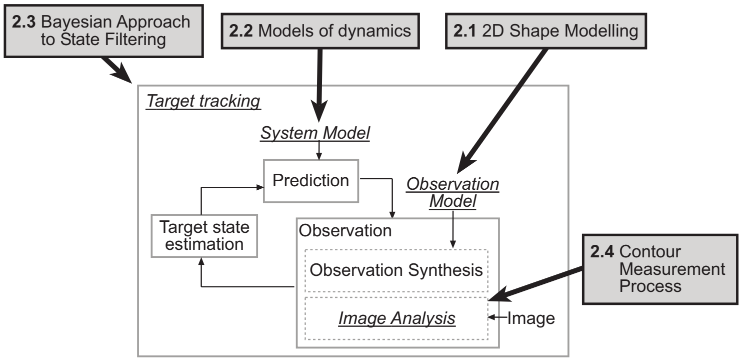


Figure 2.1: Topics analysed in the chapter.

stated as generating models insensible to the object variability, which is particularly interesting to detect specific objects of interest, or to distinguish between different object types. Using machine learning terms, the objective is a classification model, and a concrete example of these models in the context of vehicle detection can be found in Chapter 5.

However, many problems not only require recognising an specific object, but also identifying its state from a set of possible configurations. That is to say, represent object variability instead of being insensible to it, what in machine learning terms is known as a regression model. With this objective in mind, an issue of research is the design of parametric models of object variation. These models are generative, in the sense that given appropriate parameter values, they generate a concrete configuration of the object being represented. Recognising an object using these models means recognising an object in one of its concrete configurations, and this is the information of interest in many applications. These are the models of interest in tracking applications. In these applications, the variation of the object is usually due to some dynamic process, which becomes apparent in changes in the corresponding model parameters.

Many authors have worked on developing representations of shape variability in many different ways. The references [18, 31] review the major contributions on this field, and then focus on the description of a model-based approach to solve this problem. There exist different possibilities to represent parametrically the outline of an object. One of the most popular approaches is using B-spline curves. B-splines construct expressions of a 2D contour as a weighted sum of N_B basis functions. The contour points coordinates $\mathbf{r}(s) = [x(s), y(s)]^T$ of 2D shapes are obtained by an expression with the form

$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} \mathbf{B}(s)^T & \mathbf{0}_{N_B} \\ \mathbf{0}_{N_B} & \mathbf{B}(s)^T \end{bmatrix} \begin{bmatrix} \mathbf{q}^x \\ \mathbf{q}^y \end{bmatrix} , \tag{2.1}$$

or more compactly,

$$\mathbf{r}(s) = \mathbf{U}(s)\mathbf{q} .$$

Matrices $\mathbf{B}(s)^T$ in $\mathbf{U}(s)$ maintain N_B basis functions, which are curves composed of polynomials of degree d with finite support. They are C^{d-1} continuous, which means that the contour derivatives up to the $(d-1)$-th are smooth. $\mathbf{q} = [\mathbf{q}^x, \mathbf{q}^y]^T$ is the vector of the N_{cp} *control points* that weights properly the basis functions to generate a desired curve. Thus, its dimensionality corresponds to $N_q = 2N_{cp}$. The parameter s evaluates the linear combination of polynomia at concrete points.

In this thesis, contours are represented using regular periodic B-spline curves. These curves are defined in terms of a finite spline basis over a closed interval $0 \leq s \leq N_B$, where $N_B = N_{cp}$. The matrix $\mathbf{B}(s)$ corresponds to

$$\mathbf{B}(s) = [B_{0,o}(s)\ B_{1,o}(s) \ldots B_{N_B-1,o}(s)]^T \ ,$$

where $B_{n,o}(s)$ denotes a B-spline basis function, with its subindex o indicating its *order* (the degree d of its polynomial expression plus 1). $B_{n,o}(s)$ is defined in terms of piecewise polynomials obtained from the following recursive rule

Ground Instance

$$B_{n,1}(s) \quad = \quad \begin{cases} 1 & \text{if } n \leq s < n+1 \\ 0 & \text{otherwise} \end{cases}$$

Inductive step

$$B_{n,o}(s) \quad = \quad \frac{(s-n)B_{n,o-1}(s) + (n+o-s)B_{n+1,o-1}(s)}{o-1}$$

For periodic splines,

$$B_{n,o}(s) = B_{0,o}(s-n) \ .$$

Figure 2.2 exemplifies the construction of a 2D contour using a B-spline curve.

In practice, contours are not commonly synthesised from any value s, but at predefined discrete values. By fixing a sampling ratio N_{sr} between control points, a discrete contour is synthesised, consisting of N_s (x, y) coordinates, with $N_s = N_{cp}N_{sr}$. Curve sampling points s_i are regularly placed along the support of the spline parametric space, given by $\{s_i = \frac{i}{N_{sr}}\}_{i=0}^{N_s-1}$. Thus, a sampled contour representation is obtained with

$$\begin{bmatrix} x(s_0) \\ \vdots \\ x(s_{N_s}) \\ y(s_0) \\ \vdots \\ y(s_{N_s}) \end{bmatrix} = \begin{bmatrix} \mathbf{B}(s_o)^T & \mathbf{0}_{N_B} \\ \vdots & \vdots \\ \mathbf{B}(s_{N_s-1})^T & \mathbf{0}_{N_B} \\ \mathbf{0}_{N_B} & \mathbf{B}(s_o)^T \\ \vdots & \vdots \\ \mathbf{0}_{N_B} & \mathbf{B}(s_{N_s-1})^T \end{bmatrix} \begin{bmatrix} \mathbf{q}^x \\ \mathbf{q}^y \end{bmatrix} \ ,$$

or in compact form

$$\mathbf{r} \quad = \quad \mathbf{U}\mathbf{q} \ . \tag{2.2}$$

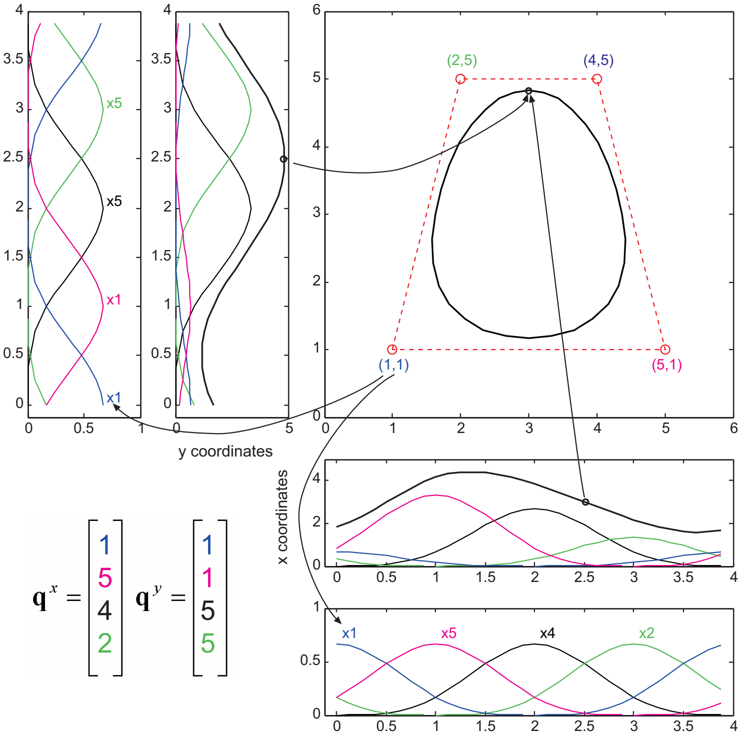


Figure 2.2: B-Splines contour construction in a nutshell. A contour is interpolated using cubic B-splines, and 4 control points. The control point values multiply their corresponding B-function, and their summation determine the x and y contour coordinates.

Using B-spline curves, a silhouette is represented by a point in an *spline space* $\mathbb{R}^{N_q}$. For complex silhouettes, the dimensionality of this space can be considerably big. This is unconvenient for tracking purposes. Estimating a 2D shape will require managing high dimension state vectors, which will be computationally expensive. Besides, the estimation obtained will be prone to be erroneous: the bigger the number of parameters to be estimated, the higher the probability of wrongly estimating some of them. For this reason, models requiring less parameters are preferred. The most popular approach is based on defining a *Shape Space*, which details a mapping of shape space vectors $\mathbf{c} \in \mathbb{R}^{N_c}$ to a spline space vectors $\mathbf{q} \in \mathbb{R}^{N_q}$, where $N_c \ll N_q$. In that way, the feasible values of $\mathbf{q}$ are constrained in a given subspace of $\mathbb{R}^{N_q}$. Next section focuses on two common methods to generate desired shape spaces, to describe respectively local and global contour transformations.

2.1.1 Local Contour Transformations

A simple way to construct a generative model to account for local variations of a mean shape $\bar{\mathbf{q}}$ is by means of defining a linear *Shape Space* $\mathcal{L}(\mathbf{W}, \bar{\mathbf{q}})$, with the form

$$\mathbf{q} = \mathbf{W}\mathbf{c} + \bar{\mathbf{q}} \ . \tag{2.3}$$

$\mathbf{W}$ is a $N_q \times N_c$ *shape matrix*, $\mathbf{c}$ the shape space vector and $\bar{\mathbf{q}}$ the spline control points of the average contour of the modelled shape. The vector $\mathbf{q}$ corresponds to a linear combination of the columns in $\mathbf{W}$ added to $\bar{\mathbf{q}}$. Thus, the family of shape variations represented by $\mathcal{L}(\mathbf{W}, \bar{\mathbf{q}})$ depends on the columns of $\mathbf{W}$, which are the *basis* of the shape space. To account for the specific variability of a shape of interest, the usual approach is establishing them from training data. Given a set $\{\mathbf{q}_i\}_{i=1}^{N}$ corresponding to the spline control points of the shapes in a training sequence, its mean and covariance $(\bar{\mathbf{q}}, \mathbf{\Sigma})$ are computed by

$$\begin{aligned} \bar{\mathbf{q}} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{q}_i \ , \\ \mathbf{\Sigma} &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{q}_i - \bar{\mathbf{q}})\mathcal{M}(\mathbf{q}_i - \bar{\mathbf{q}})^T \ , \end{aligned}$$

where $\mathcal{M}$ is a metric matrix which allows to measure the distance between B-spline curves from only their control points (see Chapter 3 in [18]). Performing a Principal Component Analysis (PCA) on $\mathbf{\Sigma}$, the principal modes of variation of the examples in the training data are obtained in the eigenvectors of $\mathbf{\Sigma}$. The N_c most significative eigenvectors (i.e., the N_c with biggest eigenvalues) are used to conform the basis in $\mathbf{W}$. N_c is usually established as the minimum number of eigenvectors whose sum of eigenvalues exceeds a given percentage of the total eigenvalues sum. The eigenvectors discarded are considered as accounting for noise in training examples. The interesting point is that $N_c \ll N_q$, and thus a more compact shape model is obtained.

2.1.2 Global Contour Transformations

A possibility to construct a shape space that accounts for global transformations of a given shape $\bar{\mathbf{q}}$ is designing a matrix $\mathbf{W}$ parameterised to account for a given space of similarities, like Euclidean or Affine transformations. For example, for the affine case, a 5 dimensional shape space is defined, maintaining

$$\mathbf{c} \quad = \quad [t_x \; t_y \; s_x \; s_y \; \theta]^T, \tag{2.4}$$

where (t_x, t_y) accounts for a 2D translation, (s_x, s_y) for x and y scale factors, and θ for a given rotation. Given a shape $\bar{\mathbf{q}}$, its global affine transformation is obtained from the expression

$$\mathbf{q} \quad = \quad \mathbf{W}_{(\mathbf{c})}\bar{\mathbf{q}} + \mathbf{T}\mathbf{c} \ , \tag{2.5}$$

where $\mathbf{W}_{()}$ is a function matrix on $\mathbf{c}$ given by

$$\mathbf{W}_{(\mathbf{c})} = \begin{pmatrix} s_x\mathbf{I}_{Nq} & \mathbf{0}_{Nq} \\ \mathbf{0}_{Nq} & s_y\mathbf{I}_{Nq} \end{pmatrix} \begin{pmatrix} \cos_{(\theta)} \mathbf{I}_{Nq} & -\sin_{(\theta)} \mathbf{I}_{Nq} \\ \sin_{(\theta)} \mathbf{I}_{Nq} & \cos_{(\theta)} \mathbf{I}_{Nq} \end{pmatrix} \quad , \tag{2.6}$$

and $\mathbf{T}$ the matrix

$$\mathbf{T} = \begin{pmatrix} \mathbf{1}_{Nq} & \mathbf{0}_{Nq} & \mathbf{0}_{Nq} & \mathbf{0}_{Nq} & \mathbf{0}_{Nq} \\ \mathbf{0}_{Nq} & \mathbf{1}_{Nq} & \mathbf{0}_{Nq} & \mathbf{0}_{Nq} & \mathbf{0}_{Nq} \end{pmatrix} \quad . \tag{2.7}$$

Notice that this proposal of affine shape space is non–linear. If the transformed base shape $\bar{\mathbf{q}}$ is constant along a sequence (i.e., the contour to be tracked is rigid), a 6 dimensional linear shape space $\mathcal{L}(\mathbf{W}, \bar{\mathbf{q}})$ can be defined, with

$$\mathbf{W} = \begin{pmatrix} \mathbf{1}_{Nq/2} & \mathbf{0}_{Nq/2} & \bar{\mathbf{q}}^x & \mathbf{0}_{Nq/2} & \mathbf{0}_{Nq/2} & \bar{\mathbf{q}}^y \\ \mathbf{0}_{Nq/2} & \mathbf{1}_{Nq/2} & \mathbf{0}_{Nq/2} & \bar{\mathbf{q}}^y & \bar{\mathbf{q}}^x & \mathbf{0}_{Nq/2} \end{pmatrix} \quad ,$$

where $\bar{\mathbf{q}} = [\bar{\mathbf{q}}^x \bar{\mathbf{q}}^y]^T$. Now an spline–vector $\mathbf{q}$ is synthesised with the expression

$$\mathbf{q} \quad = \quad \mathbf{W}\mathbf{c}' + \bar{\mathbf{q}} \ .$$

The vector $\mathbf{c}'$ relates with the affine parameters in (2.4) as

$$\mathbf{c}' \quad = \quad [t_x \; t_y \; (s_x \cos(\theta) - 1) \; (s_y \cos(\theta) - 1) \; (s_y \sin(\theta)) \; (-s_x \sin(\theta))]^T \ .$$

2.1.3 Shape Space Representability

Besides their low dimension, shape spaces still have a bigger representability than the one desired. That is, they still can synthesise shapes completely different to those in the training data. Thus, in order to improve the model specificity, a constraint model can be attached to the shape space, delimiting the subspace where parameters within generate desired *valid* shapes. Figure 2.3 shows an example of a simple constraint model attached to a shape model representing a hand with a pointing finger. In this case, constraining the model parameters just inside a Gaussian envelope is sufficient to assure that valid shapes are generated. In other problems where the modeled shape can display a wide range of different configurations, a more complex model is required to constraint the space of valid parameters (this is object of study in chapter 4).

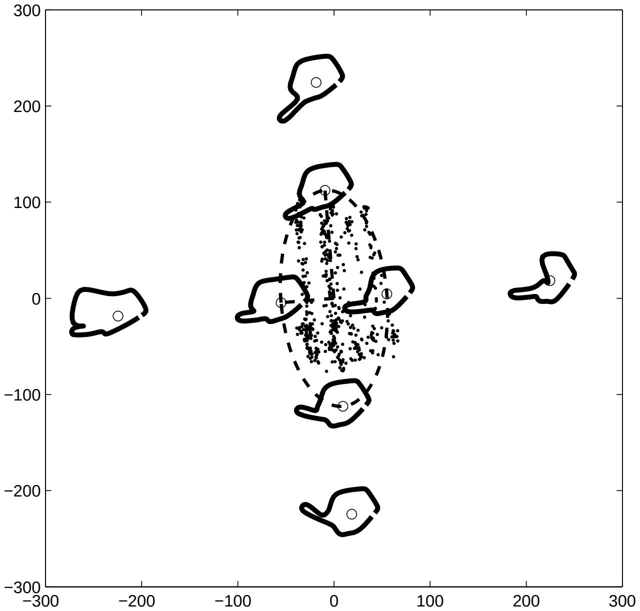


Figure 2.3: 2D shape space corresponding to a hand with a pointing finger. The orthogonal directions of the shape space model variations in the orientation of the pointing finger (vertical direction), and the degree of folding of the rest of hand fingers (horizontal direction). The plotted ellipse (*dashed*) delimit a Gaussian envelope where silhouettes in the training set project (*dots*). The shape corresponding to specific points in the shape space (*small circles*) is reconstructed. Notice that locations outside the gaussian envelope correspond to invalid outlines.

2.2 Models of Dynamics

In the context of contour tracking applications, in addition to a generative model of the target shape, it is required to model how this shape is expected to vary along time. Using parametric shape models as the ones previously presented, shape variation between successive time steps can be described by means of a model of the variation of their parameters. Thus, the objective is establishing a dynamic model which fits an expected shape parameter dynamics.

Concerning visual tracking applications, dynamics are usually modelled by means of discrete time series. The value of a parameter x at a given instant t (denoted as x_t) is determined using an expression of the form

$$x_t \quad = \quad v + \sum_{k=1}^{n} \alpha_k x_{t-k} + \xi_t \ , \tag{2.8}$$

where v and α_k are real constants, with $\alpha_k \neq 0$. ξ_t is an stochastic error term that accounts for the inaccuracies of the deterministic part of (2.8). When ξ_t corresponds to a Gaussian white noise process, then (2.8) is denoted as Markov process or Auto–Regressive process of order n (AR(n)) . For notation convenience, Equation (2.8) is commonly expressed in vector–matrix form. For instance, for models where $v = 0$ and $\xi_t = b_0 w_t$ corresponds to a Gaussian white noise process with parameters $\mathcal{N}(0, b_0 b_0^T)$, then Equation (2.8) is expressed as

$$\begin{bmatrix} x_t \\ x_{t-1} \\ x_{t-2} \\ \vdots \\ x_{t-(n-1)} \end{bmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} & \alpha_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ x_{t-2} \\ x_{t-3} \\ \vdots \\ x_{t-n} \end{bmatrix} + \begin{bmatrix} b_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} w_t \tag{2.9}$$

This is called the companion form of Equation (2.8), and it describes the n-th order dynamics of x_t as a first–order Markov model in state space. Usually it is expressed more compactly as follows

$$\mathbf{x}_t \quad = \quad \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{w}_t \ , \tag{2.10}$$

where terms equals one to one with vectors and matrices in Equation (2.9), except for the part of the stochastic error term ξ_t. Here it is alternatively expressed using a $n \times n$ $\mathbf{B}$ matrix given by

$$\mathbf{B} \quad = \quad \begin{bmatrix} b_0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \ ,$$

and a $n \times 1$ Gaussian white noise vector $\mathbf{w}_t$.

In some cases, a third parameter $\bar{\mathbf{x}}$ is used to fix a desired average value of $\mathbf{x}_t$ along time, leading to the following final expression of dynamics

$$\mathbf{x}_t - \bar{\mathbf{x}} \quad = \quad \mathbf{A}(\mathbf{x}_{t-1} - \bar{\mathbf{x}}) + \mathbf{B}\mathbf{w}_t \ , \tag{2.11}$$

which, in fact, only adds just a constant term in the right–side part of expression (2.10), namely $(\mathbf{A}-\mathbf{I})\bar{\mathbf{x}}$.

This formalisation is the one adopted for the different problems analysed in this thesis. To enhance the readability of the chapter, a more detailed explanation on the use of AR processes to model dynamics has been placed in Appendix A. In this appendix two novel methods are proposed to establish the parameters of AR(1) and AR(2) models, with the aim of constraining the evolution of a parameter of interest x inside a desired Gaussian subspace. This kind of *constrained* dynamical models are very appealing for tracking purposes, since they provide some degree of control on the *validity* (see Section 2.1.3) of predicted object states.

2.3 Bayesian Approach to State Filtering

Previous sections have described the models used in this thesis to represent a shape and its dynamic behaviour. Now the focus is on describing how these models can be used to extract reliable information from sequences. Determining the outline of an object in a video sequence is a task that can only be done with some uncertainty. Observations extracted from frames are subject to several noise sources (effects of digitalisation, acquisition conditions, clutter, etc.), providing just inaccurate evidence of the target of interest. Models can help to make a better interpretation of observations, but as long as they are just an approximation of reality, they add also uncertainty in the estimated information. Thus, methods in estimation theory face the problem of determining the most likely state of a given target, providing at the same time some measure of the certainty in this guess.

Classically estimation has been formulated from a computational point of view, as the problem of finding the target state which minimises a given defined error. Kalman–based estimation algorithms were developed with the aim to provide a recursive solution to the least–square estimation method. We refer the reader to [47] for an excellent review of classical estimation theory. A more general way to pose the estimation problem is from a probabilistic point of view, using a Bayesian formulation. Bayesian modelling provides a principled way to represent the available prior knowledge about a phenomenon being modelled. Uncertainty is explicitly represented by means of probability density functions, which may be of arbitrary form. Inference on unknown quantities is obtained from the application of Bayes' theorem. What has made this new problem statement important in estimation theory is that it has provided a new perspective to solve the estimation problem through the use of simulation–based methods. Let's detail which is the estimation problem to be solved.

Solving a sequential estimation problem consist in estimating the sequence of a state[1] $\mathbf{x}_{0:t}=[\mathbf{x}_i]_{i=0}^{t}$, from observations $\mathbf{y}_{1:t}$ (i.e., observations up to instant t). In Bayesian terms, this corresponds to estimate the posterior density

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \ . \tag{2.12}$$

The estimation of (2.12) can be expressed recursively by applying Bayes' theorem,

[1] In contour tracking, this state is constituted by parameters of the defined shape model.

obtaining

$$\begin{aligned} p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) &= \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}\mathbf{y}_{1:t-1})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \\ &= \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}\mathbf{y}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}) \ . \end{aligned} \tag{2.13}$$

Notice the recursive nature of this estimation problem, where the posterior at a given instant is defined in terms of its estimation at the previous time instant. In tracking applications the objective is usually posed as estimating the *filtering density* $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, which is a marginal of $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ corresponding to

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \int p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d_{\mathbf{x}_{0:t-1}} \ ,$$

that can be obtained by just discarding the state part corresponding to $\mathbf{x}_{0:t-1}$ from $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. In practice, the filtering density is computed recursively, without the need to keep track of the complete history of states.

Expression (2.13) is usually simplified by making two assumptions, which generally hold in tracking applications:

- $\mathbf{x}_t$ conditionally depends only on its state at the previous instant $\mathbf{x}_{t-1}$.
- observations $\mathbf{y}_{1:t}$ are conditionally independent given the state of the object at each instant.

These assumptions lead to

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}) \ . \tag{2.14}$$

To solve this expression it is required to define two important terms:

- $p(\mathbf{x}_t|\mathbf{x}_{t-1})$: the dynamics of $\mathbf{x}_t$ expressed by a first order Markov model[2], i.e., the probabilistic model of the state evolution, commonly referred as *transition prior*, or *state transition model*.
- $p(\mathbf{y}_t|\mathbf{x}_t)$: the observation likelihood, or how the state is reflected on observations. It is commonly referred as *state measurement model*.

Term $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ factorises as the integration of the rest of right–side terms in (2.14). That is

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_t \ .$$

When $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_t)$ correspond to Gaussian distributions, it is possible to solve (2.14) analytically. Next section focuses on this particular case.

[2] A first order Markov model in *state space* does not means being limited to just first order dynamics in *shape space*(see Section 2.2).

2.3.1 Kalman–based Filters

The Kalman filter (KF) provides a solution to the estimation problem for cases with

- a Normal initial state distribution.

$$p(\mathbf{x}_0) \quad = \quad \mathcal{N}(\bar{\mathbf{x}}_0, \mathbf{\Sigma}_0)$$

- linear state dynamics, perturbed by zero–mean normal white noise. That is, a Markov model given by

$$\mathbf{x}_t \quad = \quad \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{w}_t \qquad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{1})\ ,\ \mathbf{B}_t\mathbf{B}_t^T = \mathbf{Q}_t. \tag{2.15}$$

- linear observation model, perturbed by zero–mean normal white noise. That is

$$\mathbf{y}_t \quad = \quad \mathbf{H}_t\mathbf{x}_t + \mathbf{V}_t\mathbf{v}_t \qquad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{1})\ ,\ \mathbf{V}_t\mathbf{V}_t^T = \mathbf{R}_t. \tag{2.16}$$

Matrices $\mathbf{A}_t$ and $\mathbf{H}_t$ are respectively referred as system and observation matrices, and both are usually constant in most of practical tracking applications. Thus, in the following their subscript is eliminated. $\mathbf{Q}_t$ and $\mathbf{R}_t$ model the noise assumed at each time step. The action of a known control input $\mathbf{u}_t$ or a constant term $\mathbf{D}$[3] could be also considered in (2.15) and (2.16). For the sake of clarity, these terms are assumed zero in this formulation. The expression of these models in Bayesian terms correspond to the following Gaussian distributions

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad = \quad \mathcal{N}(\mathbf{A}\mathbf{x}_{t-1}, \mathbf{Q}_t)\ , \tag{2.17}$$

$$p(\mathbf{y}_t|\mathbf{x}_t) \quad = \quad \mathcal{N}(\mathbf{H}\mathbf{x}_t, \mathbf{R}_t)\ . \tag{2.18}$$

Being all the right–side terms in (2.14) Normal distributions, the posterior distribution turns out to be also Normal. It can be shown (see Appendix B for a detailed explanation) that parameters of $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ can be computed from a linear combination of the parameters of

- the system state prediction distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$,
- the expected observation distribution $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$,
- the cross-correlation between the predicted state and observations $\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}$.

For the linear Gaussian models assumed in the Kalman filter, these parameters are determined analytically. Before detailing their expressions, some notation has to be introduced. In the following, a subscript is added to parameters to indicate concisely at which instant they are estimated, and which information up to a given instant has been used. Let's see some examples

- $\hat{\mathbf{x}}_{t|t-1}$: denotes the estimation of $\mathbf{x}$ at instant t, using the observations available up to instant $t-1$. The symbol ˆ indicates that its value corresponds to just an estimation of the *real* value $\mathbf{x}_{t|t-1}$.

[3] For instance, the term $(\mathbf{A} - \mathbf{I})\bar{\mathbf{x}}$ derived from the model of dynamics in (2.11).

- $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t}$: denotes the covariance of state $\mathbf{x}$ at instant t, from observations up to instant t.

Once the notation has been presented, let's express how the Kalman filter estimates $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, from the parameters of this distribution at the previous instant $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, given by

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \quad \simeq \quad p_{\mathcal{N}}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_{t-1|t-1}, \mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1}) \ .$$

From the assumed system and the observation models, the parameters of the densities required by the Kalman filter are determined as

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) &\sim \mathcal{N}(\hat{\mathbf{x}}_{t|t-1}, \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}) = \mathcal{N}(\mathbf{A}\hat{\mathbf{x}}_{t-1|t-1},\, \mathbf{A}\mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\mathbf{A}^T + \mathbf{Q}_t) \ , \\ p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) &\sim \mathcal{N}(\hat{\mathbf{y}}_{t|t-1}, \mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1}) = \mathcal{N}(\mathbf{H}\hat{\mathbf{x}}_{t|t-1},\, \mathbf{H}\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T + \mathbf{R}_t) \ , \\ \Sigma^{\mathbf{xy}}_{t|t-1} &= \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T \ . \end{aligned}$$

Given these parameters, the final posterior distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is characterised using the well–known Kalman expressions, usually organised in the following two steps:

- Kalman Prediction Step

$$\begin{aligned} \hat{\mathbf{x}}_{t|t-1} &= \mathbf{A}\hat{\mathbf{x}}_{t-1|t-1} \ , \\ \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} &= \mathbf{A}\mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\mathbf{A}^T + \mathbf{Q}_t \ . \end{aligned}$$

- Kalman Updating Step

$$\begin{aligned} \mathbf{K} &= \frac{\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T}{\mathbf{H}\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T + \mathbf{R}_t} \ , \\ \nu_t &= \mathbf{y}_t - \mathbf{H}\hat{\mathbf{x}}_{t|t-1} \ , \\ \hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}\nu_t \ , \\ \mathbf{\Sigma}^{\mathbf{xx}}_{t|t} &= (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} \ . \end{aligned}$$

The term ν_t, commonly denoted as *innovation*, measures the disparity between observed and predicted measurements (i.e., the difference between $\mathbf{y}_t$ and $\hat{\mathbf{y}}_{t|t-1}$). Figure 2.4 relates the KF equations within the modules of the tracking framework presented at the beginning of the chapter.

Extended Kalman Filter

Many real problems can not be properly described using only linear expressions, and require the use of non–linear models as

$$\begin{aligned} \mathbf{x}_t &= a(\mathbf{x}_{t-1}, \mathbf{w}_t) \qquad \mathbf{w}_t \sim \mathcal{N}(\bar{\mathbf{w}}_t, \mathbf{Q}_t) \ , && (2.19) \\ \mathbf{y}_t &= h(\mathbf{x}_t, \mathbf{v}_t) \qquad \mathbf{v}_t \sim \mathcal{N}(\bar{\mathbf{v}}_t, \mathbf{R}_t) \ , && (2.20) \end{aligned}$$

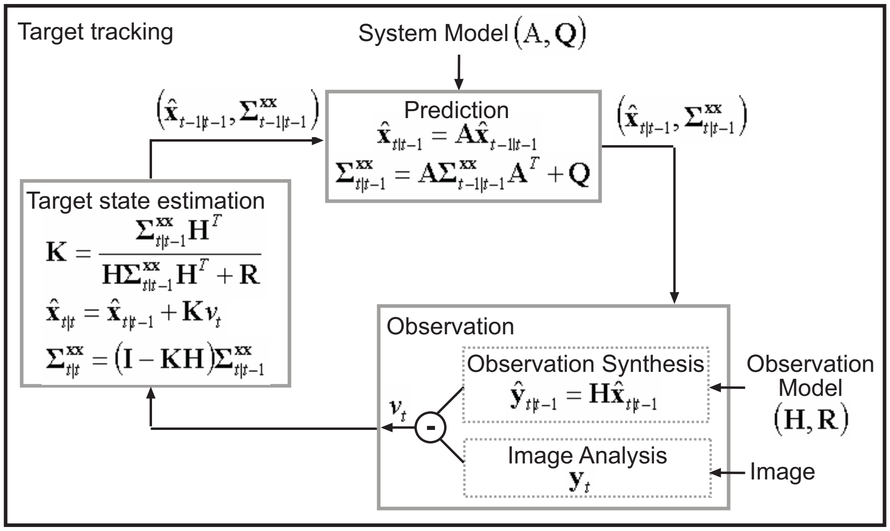

Target tracking
System Model $(\mathbf{A},\mathbf{Q})$
$(\hat{\mathbf{x}}_{t-1|t-1},\boldsymbol{\Sigma}^{\mathbf{xx}}_{t-1|t-1})$
Prediction
$\hat{\mathbf{x}}_{t|t-1}=\mathbf{A}\hat{\mathbf{x}}_{t-1|t-1}$
$\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}=\mathbf{A}\boldsymbol{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\mathbf{A}^T+\mathbf{Q}$
$(\hat{\mathbf{x}}_{t|t-1},\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1})$
Target state estimation
$\mathbf{K}=\frac{\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T}{\mathbf{H}\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T+\mathbf{R}}$
$\hat{\mathbf{x}}_{t|t}=\hat{\mathbf{x}}_{t|t-1}+\mathbf{K}\boldsymbol{v}_t$
$\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t}=(\mathbf{I}-\mathbf{K}\mathbf{H})\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}$
Observation
Observation Synthesis
$\hat{\mathbf{y}}_{t|t-1}=\mathbf{H}\hat{\mathbf{x}}_{t|t-1}$
Observation Model $(\mathbf{H},\mathbf{R})$
$\boldsymbol{v}_t$
-
Image Analysis
$\mathbf{y}_t$
Image

Figure 2.4: The KF within the model–based tracking framework.

where $a()$ and $h()$ model the non–linear system and observation processes, and $\bar{\mathbf{w}}_t$ and $\bar{\mathbf{v}}_t$ are usually assumed to equal to zero. Again, control inputs and constants are not considered in the formulation presented for the sake of clarity. Notice that although the disturbing noise is considered Gaussian too, it is no longer necessaryly additive. The system non–linearities prevent finding a close–form solution to $p(\mathbf{x}_t|\mathbf{y}_{0:t})$ but a Gaussian approximation of it can be obtained by linearising the system, and then applying the previously described Kalman equations. This is the approach followed by the Extended Kalman Filter (EKF) , based on the Taylor series expansion of $a()$ and $h()$. The linear approximations are defined in terms of the Jacobians of the system and observation processes, given by

$$\begin{aligned}
\mathbf{A}_{\mathbf{lx}} &\triangleq \frac{\partial a(\mathbf{x}_{t-1},\mathbf{w}_t)}{\partial \mathbf{x}_{t-1}}|(\mathbf{x}_{t-1}=\hat{\mathbf{x}}_{t-1|t-1},\mathbf{w}_t=\bar{\mathbf{w}}_t) \ , \\
\mathbf{A}_{\mathbf{lw}} &\triangleq \frac{\partial a(\mathbf{x}_{t-1},\mathbf{w}_t)}{\partial \mathbf{w}_t}|(\mathbf{x}_{t-1}=\hat{\mathbf{x}}_{t-1|t-1},\mathbf{w}_t=\bar{\mathbf{w}}_t) \ , \\
\mathbf{H}_{\mathbf{lx}} &\triangleq \frac{\partial h(\mathbf{x}_t,\mathbf{v}_t)}{\partial \mathbf{x}_t}|(\mathbf{x}_t=\hat{\mathbf{x}}_{t|t-1},\mathbf{v}_t=\bar{\mathbf{v}}_t) \ , \\
\mathbf{H}_{\mathbf{lv}} &\triangleq \frac{\partial h(\mathbf{x}_t,\mathbf{v}_t)}{\partial \mathbf{v}_t}|(\mathbf{x}_t=\hat{\mathbf{x}}_{t|t-1},\mathbf{v}_t=\bar{\mathbf{v}}_t) \ .
\end{aligned}$$

Using these linear expansion terms, it is straightforward to derive the following filtering equations using the same reasoning that for the standard Kalman filter.

- Extended Kalman Prediction Step

$$\begin{aligned}
\hat{\mathbf{x}}_{t|t-1} &= a(\hat{\mathbf{x}}_{t-1|t-1},\bar{\mathbf{w}}_t) \ , \\
\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1} &= \mathbf{A}_{\mathbf{lx}}\boldsymbol{\Sigma}^{\mathbf{xx}}_{t-1|t-1}{\mathbf{A}_{\mathbf{lx}}}^T+\mathbf{A}_{\mathbf{lw}}\mathbf{Q}_t{\mathbf{A}_{\mathbf{lw}}}^T \ .
\end{aligned}$$

- Extended Kalman Updating Step

$$
\begin{aligned}
\mathbf{K} &= \frac{\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H_{lx}}^T}{\mathbf{H_{lx}}\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H_{lx}}^T + \mathbf{H_{lv}}\mathbf{R}_t\mathbf{H_{lv}}^T} \ , \\
\nu_t &= \mathbf{y}_t - h\left(\hat{\mathbf{x}}_{t|t-1}, \bar{\mathbf{v}}_t\right) \ , \\
\hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}\nu_t \ , \\
\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t} &= (\mathbf{I} - \mathbf{K}\mathbf{H_{lx}})\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1} \ .
\end{aligned}
$$

Unscented Kalman Filter

The Unscented Kalman Filter (UKF) [114, 119, 63] is a quite recent approach that deals with non–linearities in the Kalman scheme from a novel point of view. In essence, the Kalman approach is based on the propagation along time of a Gaussian distribution, through the linear processes describing a given system. When these processes are non–linear, the propagation of this Gaussian does not remain Gaussian anymore, and some mechanism has to used to obtain a Gaussian approximation of it. The solution provided by the EKF is based on approximating the non–linear processes by linear ones, forcing the propagated Gaussian to remain Gaussian. The UKF gives a solution from another perspective: instead of approximating non–linear functions, the idea is propagate the Gaussian non–linearly, and then approximate the resultant distribution with a Gaussian. This is the task carried out by the so called *Unscented Transform.* Given a Gaussian distribution and a non–linear function to transform it, a set of weighted samples (denoted as *σ–points*) are deterministically chosen from the distribution, in a way that their sample mean and covariance matches the statistics of the original distribution. There exist different proposals to determine these points, being popular the methods described in [62, 64]. Each σ-point has two weights w_m and w_c associated, which are required to posteriorly compute the Gaussian parameters approximation. These σ–points are propagated using the non–linear system, generating a cloud of transformed points. Computing the weighted sample mean and covariance of these points (using, respectively, w_m and w_c) the transformed Gaussian distribution is characterised. Figure 2.5 exemplifies this procedure.

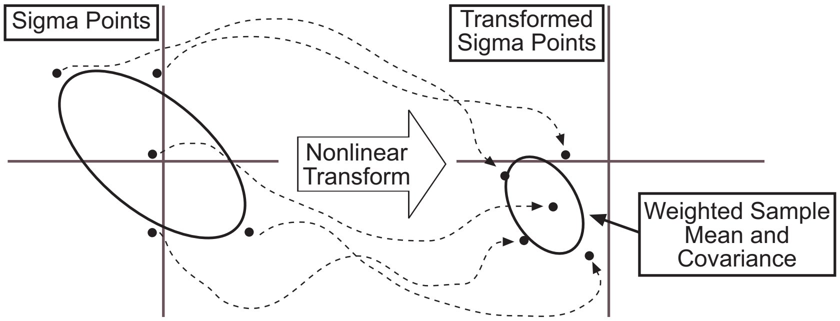


Figure 2.5: The Unscented Transform mechanism. Procedure to estimate the mean and covariance of a Gaussian distribution, when is non–linearly propagated.

The UKF takes profits of this transformation to compute the terms required by Kalman update equations, namely

- the distribution of the system state prediction $\mathcal{N}(\hat{\mathbf{x}}_{t|t-1}, \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1})$.
- the distribution of the expected observations $\mathcal{N}(\hat{\mathbf{y}}_{t|t-1}, \mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})$.
- the cross-correlation between the predicted state and observations $\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}$.

The UKF procedure is as follows. First, a Gaussian distribution is stated from the concatenation of the variables of the system (the original state and noise variables), with parameters

$$\mathbf{x}^a_t = \left[\begin{array}{ccc} \hat{\mathbf{x}}_{t-1|t-1} & \bar{\mathbf{w}}_t & \bar{\mathbf{v}}_t \end{array}\right]^T , \tag{2.21}$$

$$\mathbf{\Sigma}^{\mathbf{x}^a\mathbf{x}^a}_t = \left[\begin{array}{ccc} \mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_t \end{array}\right] . \tag{2.22}$$

From this distribution, *σ–points* are generated and used to estimate the distributions required by the Kalman filter. Once having these terms, the classical Kalman update expressions can be used. Algorithm 1 summarises an iteration of the UKF

The UT mechanism allows to obtain the statistics required by Kalman equations in a very simple an efficient way. With respect to EKF, the UKF algorithm is easier to implement, and it can be shown that higher accuracy is obtained [63]. The first order system linearisations in a EKF introduces large errors in the estimated statistics of the posterior state distribution, specially when the models are highly non–linear, and the local linearity assumption breaks down (i.e., the effects of the higher order term of the Taylor expression become significant). On the other hand, the UKF captures the posterior statistics with an accuracy equivalent to a second order Taylor expansion of any non–linear function.

2.3.2 Particle Filters

Methods described up to this point are based on approximating the $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ probability density function (PDF) with a Gaussian. In general, $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ can have any form, so the performance of these methods will be conditioned on the validity of this approximation. To overcome this problem, Particle Filters (PFs) are simulation–based methods that approximate this PDF by means of population of samples (i.e., particles), being thus able to represent any arbitrarily complex density (see [8] for a tutorial). Estimating a sample population describing accurately $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is not a trivial task, as there is no possibility to generate samples from this distribution (otherwise, it would be already known and no estimation needed). The methods commonly applied to solve this problem are popularly known as Monte Carlo methods, and they are distinguished from other simulation methods by being stochastic (i.e., nondeterministic in some manner), using random numbers or (in practice) pseudo-random numbers. The theory of these methods was developed in the 50s for early work related to the development of the hydrogen bomb, and since then, their practical application

Algorithm 1 Unscented Kalman Filter Iteration

```
$\left[\left(\hat{\mathbf{x}}_{t|t}, \mathbf{\Sigma}^{\mathbf{xx}}_{t|t}\right)\right] = \text{UKF}\left[\left(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\right), \mathbf{y}_{1:t}\right]$

  Establish $\mathbf{x}^a_t$ and $\mathbf{\Sigma}^{\mathbf{x}^a\mathbf{x}^a}_t$
  $\left[\left\{\mathbf{x}^{a(i)}_t, w^{(i)}_m, w^{(i)}_c\right\}_{i=1}^{N}\right] = \text{COMPUTE\_}\sigma\text{\_POINTS}\left[\left(\mathbf{x}^a_t, \mathbf{\Sigma}^{\mathbf{x}^a\mathbf{x}^a}_t\right)\right]$
  { where $\mathbf{x}^{a(i)}_t = [\mathbf{x}^{(i)}_{t-1|t-1} \quad \mathbf{w}^{(i)}_t \quad \mathbf{v}^{(i)}_t]$, and $w^{(i)}_m$ and $w^{(i)}_c$ are the weights associated to each $\sigma$-point }

  Time update equations:

    $\mathbf{x}^{(i)}_{t|t-1} = a(\mathbf{x}^{(i)}_{t-1|t-1}, \mathbf{w}^{(i)}_t) \quad \forall i = 1, \ldots, N$
    $\bar{\mathbf{x}}_{t|t-1} = \sum_{i=1}^{N} w^{(i)}_m \mathbf{x}^{(i)}_{t|t-1}$
    $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} = \sum_{i=1}^{N} w^{(i)}_c (\mathbf{x}^{(i)}_{t|t-1} - \bar{\mathbf{x}}_{t|t-1})(\mathbf{x}^{(i)}_{t|t-1} - \bar{\mathbf{x}}_{t|t-1})^T$

  Measurement update equations:

    $\mathbf{y}^{(i)}_{t|t-1} = h(\mathbf{x}^{(i)}_{t|t-1}, \mathbf{v}^{(i)}_t) \quad \forall i = 1, \ldots, N$
    $\bar{\mathbf{y}}_{t|t-1} = \sum_{i=1}^{N} w^{(i)}_m \mathbf{y}^{(i)}_{t|t-1}$
    $\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1} = \sum_{i=1}^{N} w^{(i)}_c (\mathbf{y}^{(i)}_{t|t-1} - \bar{\mathbf{y}}_{t|t-1})(\mathbf{y}^{(i)}_{t|t-1} - \bar{\mathbf{y}}_{t|t-1})^T$
    $\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1} = \sum_{i=1}^{N} w^{(i)}_c (\mathbf{x}^{(i)}_{t|t-1} - \bar{\mathbf{x}}_{t|t-1})(\mathbf{y}^{(i)}_{t|t-1} - \bar{\mathbf{y}}_{t|t-1})^T$
    $\mathbf{K} = \mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}(\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})^{-1}$
    $\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}(\mathbf{y}_t - \bar{\mathbf{y}}_{t|t-1})$
    $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t} = \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} - \mathbf{K}\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1}\mathbf{K}^T$
```

in a great variety of problems has grown parallelly to the increase of computational power of generic purpose processors. Their use is very popular in finding solutions to mathematical problems involving many variables, which can not be efficiently solved using deterministic numerical methods. The most outstanding characteristic of Monte Carlo methods is that their efficiency increases when the dimension of the problem increases. For a didactic introduction to this methods the reader may refer to [105, 79]. A more thorough review can be found in [73]. Classical problems where Monte Carlo methods are applied are:

- expectation estimation: estimate the expectation of functions given a variable distribution $p(x)$

$$E[f(x)] \quad = \quad \int f(x)p(x)dx \ \ ;$$

- PDF sampling: generate a collection of samples distributed accordingly to a given PDF $p(x)$.

In fact, PDF sampling is implicitly required to solve the expectation estimation problem, so methods developed with this first aim also allow to provide a population of samples describing $p(x)$. Common methods directed to PDF sampling are the classical Rejection Sampling method, or Markov Chain Monte Carlo methods like the Metropolis–Hasting method, or the Gibbs Sampler [118], but they require too much computations to be used in real–time applications. The common approach used in tracking application to approximate $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ by a population of samples is the *Importance Sampling* technique. This approach is the basis of a collection of algorithms that in estimation theory have been denoted as Particle Filters. Next section describes briefly how the importance sampling technique is applied to solve the tracking problem. For readers not familiar with this technique, a more detailed explanation is provided in Appendix C.

Sequential Importance Sampling

The task of particle filters is to provide a particle set representing properly the distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. As generating samples from this distribution is not possible, the mechanism of importance sampling is used to approximate it. Very roughly, the task carried out is the following: A sample set $\{\mathbf{x}_{0:t}^{(i)}\}_{i=1}^{N}$ is first generated from a defined importance function $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Then, a weight $\tilde{w}_t^{(i)}$ is associated to each sample $\mathbf{x}_{0:t}^{(i)}$. This weight evaluates if $\mathbf{x}_{0:t}^{(i)}$ may be considered also a real sample of $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Thus, the desired posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ is approximated by

$$\widetilde{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \quad = \quad \sum_{i=1}^{N} \delta_{\mathbf{x}_{0:t}^{(i)}} \tilde{w}_t^{(i)} \ \ , \tag{2.23}$$

where $\mathbf{x}_{0:t}^{(i)}$ is a random sample generated from $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, in the following expressed as

$$\mathbf{x}_{0:t}^{(i)} \quad \sim \quad q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \ \ ,$$

and

$$\begin{aligned}\tilde{w}_t^{(i)} &= \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}} \ , \\ w_t^{(i)} &= \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})} \ .\end{aligned} \tag{2.24}$$

The weights $\tilde{w}_t$ and w_t are denoted respectively as normalised and unnormalised importance weights. If the importance function is chosen of the form

$$q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \quad = \quad q(\mathbf{x}_0)\prod_{k=1}^{t} q(\mathbf{x}_k|\mathbf{x}_{0:k-1}\mathbf{y}_{1:k}) \ ,$$

the weighted sample set in (2.23) can be estimated efficiently along time by a recursive expression. Using Bayes' theorem, the importance weight w_t turns out to be

$$\begin{aligned}w_t &= \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}\mathbf{y}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}, \\ &= \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}\mathbf{y}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})}\frac{p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})}{q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})}, \\ &= \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}\mathbf{y}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})} w_{t-1} \ .\end{aligned} \tag{2.25}$$

Thus, the importance weight of a particle at instant t can be computed in terms of its preceding value. In many formulations, the term $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ is omitted from (2.25), as acts as a normalisation factor which cancels when $\tilde{w}_t$ is computed. In most practical cases, as first order state dynamics are assumed as well, and observations are independently conditioned on $\mathbf{x}_t$, the weight w_t is finally computed as

$$w_t \quad \propto \quad \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})} w_{t-1} \ .$$

Thus, the Sequential Importance Sampling (SIS) algorithm consists in the recursive propagation of weights and samples of $\widetilde{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ as each new measurement is received. The initial particle set $\{\mathbf{x}_0^{(i)}, w_0^{(i)}\}_{i=1}^N$ propagated by this algorithm is determined from the system priors, so that

$$\begin{aligned}\mathbf{x}_0^{(i)} &\sim q(\mathbf{x}_0) = p(\mathbf{x}_0) \ , \\ w_0^{(i)} &= 1 \ .\end{aligned}$$

An iteration of this algorithm is detailed in the pseudocode in Algorithm 2.

Using this basic procedure, different algorithms have been proposed which differ in the importance function used to generate samples (some proposals will be presented latter). However, whichever the importance function used, the SIS algorithm can not avoid suffering what has been called the *Degeneracy* phenomenon. It can be proved [41, 69] that at each iteration the variance of weights can only increase over time,

Algorithm 2 Sequential Importance Sampling Iteration

```
$[\{\mathbf{x}_{0:t}^{(i)}, w_t^{(i)}\}_{i=1}^N] = \mathrm{SIS}[\{\mathbf{x}_{0:t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^N, \mathbf{y}_{1:t}]$

  for $i = 1$ to $N$ do
    Draw $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t})$
    $\mathbf{x}_{0:t}^{(i)} = [\mathbf{x}_{0:t-1}^{(i)}\ \mathbf{x}_t^{(i)}]$
    Update weight $w_t^{(i)} = \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}^{(i)}\mathbf{y}_{1:t-1})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t-1})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t})} w_{t-1}^{(i)}$
  end for
```

which in practical terms means that after several iterations, all but one particle will have negligible weight. This provokes that a large computation effort is devoted to updating particles whose contribution to $\widetilde{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ is almost zero, which consequently represents a poor approximation of $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. The brute force approach to mitigate this effect is having a very large amount of particles N, but then this precludes the use of this method in real–time applications. An operative solution to this problem did not appear until the last decade, when [50] proposed the addition of a *Resampling* step at the end of the SIS iteration. The intuitive idea is to get rid of particles with small weights, focusing on particles of bigger *importance*. This is achieved by resampling with replacement the point mass distribution $\widetilde{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, generating a theorically equivalent point distribution density $\widehat{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ given by

$$\widehat{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \quad = \quad \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{x}_{0:t}^{(i)}} \quad .$$

With this process, the importance of each particle now gets reflected in the number of copies generated from it, having each copy a weight equal to $1/N$. This obviously resets the particle weight variance to the feasible minimum. To implement the resampling step, several algorithms have been proposed, being popular multinomial [50], stratified [68], residual [74] and systematic [26] resampling schemes. A comparison between them can be found in [39].

This resampling step is applied by some authors at each SIS iteration. However, since its finality is addressing the degeneracy problem, it is reasonable to apply it only when this degeneracy effectively is present on the particle set. A suitable measure of degeneracy is the *effective sample size* N_{eff} introduced in [69], a term commonly estimated (see [41, 78, 76]) using the approximation

$$N_{eff} \quad \sim \quad \frac{1}{\sum_{i=1}^{N}(\tilde{w}_t^{(i)})^2} \quad . \tag{2.26}$$

In the ideal non–degenerated case, all particles have identical weight $1/N$, resulting that $N_{eff} = N$. As the weight variance increases, N_{eff} diminishes. Thus, a common approach is applying the resampling step only when N_{eff} is below a defined threshold N_{deg}. Algorithm 3 specifies the general framework of Sequential Importance Sampling with Resampling (SISR) algorithms, popularly denoted as Particle Filters.

Algorithm 3 Sequential Importance Sampling Resampling Iteration

```
$[\{\mathbf{x}_{0:t}^{(i)}, w_t^{(i)}\}_{i=1}^{N}] = \mathrm{SISR}[\{\mathbf{x}_{0:t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}, \mathbf{y}_{1:t}]$

  for $i = 1$ to $N$ do
    Draw $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t})$
    $\mathbf{x}_{0:t}^{(i)} = [\mathbf{x}_{0:t-1}^{(i)}\ \mathbf{x}_t^{(i)}]$
    Update weight $w_t^{(i)} = \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}^{(i)}\mathbf{y}_{1:t-1})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t-1})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t})} w_{t-1}^{(i)}$
  end for
  Normalise weights $\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}} \quad \forall i = 1, \dots, N$
  Compute $N_{eff} = \frac{1}{\sum_{i=1}^{N}(\tilde{w}_t^{(i)})^2}$
  if $N_{eff} < N_{deg}$ then
    $[\{\mathbf{x}_{0:t}^{(i)}, w_t^{(i)}\}_{i=1}^{N}] = \mathrm{RESAMPLE}[\{\mathbf{x}_{0:t}^{(i)}, \tilde{w}_t^{(i)}\}_{i=1}^{N}]$
  end if
```

Although the resampling step solves effectively the degeneracy problem, it does so by reducing the diversity among particles (see Figure 2.6). This undesirable side effect is commonly denoted as *sample impoverishment*. The resampled particle set will contain multiple repeated particles, being far from the ideal set of independent identical distributed random samples from $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. To avoid that, different methods have been proposed, as for example the use Markov Chain Monte Carlo methods to implement the resampling step [74, 6], or the regularisation of the particle set before resampling [88]. However, in most of the particle filter implementations, the sample impoverishment problem is overlooked. The relevance of this problem attenuates with the number of particles, and for the sake of simplicity, in practice is commonly preferred the use of a less complex algorithm, whose lower computational complexity allows the use of more particles. This has been also the consideration done in the experiments carried out in this thesis.

Choice of the Importance Functions

This section summarises the basic concepts proposed to define $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})$, which is the essence of any particle filter implementation. The following cases can be distinguished:

Prior Distribution $p(\mathbf{x}_t|\mathbf{x}_{0:t-1})$. Is the most common choice of importance function, and defines what is commonly referred as Bootstrap filter or Condensation algorithm. It is based on propagating particles in the previous instants using the expected system dynamics.

Optimal Distribution $p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})$. It can be shown [41] that this importance function minimises optimally (but not avoids) the rise of weight variance at each SISR iteration. The major problem is that, except for some specific situations, it is not straightforward to define this distribution.

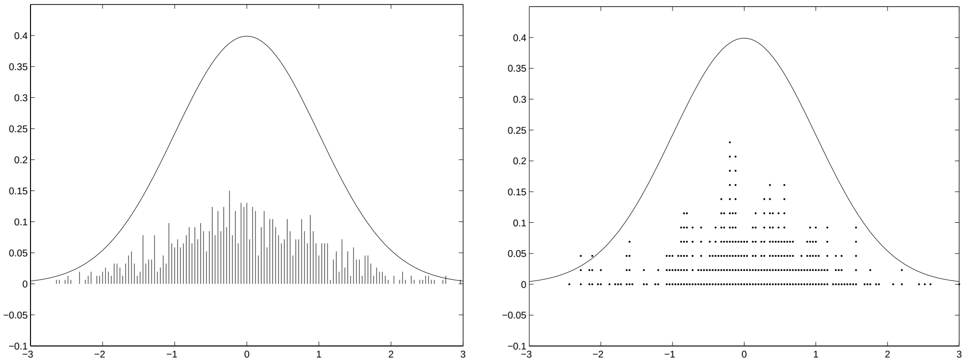


Figure 2.6: Sample–based density approximation. Left: approximation of a 1D Gaussian distribution by means of a weighted sample set $\widetilde{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Each particle is represented by a bar, whose height denotes its weight. Right: result of resampling the weighted sample set, where the importance of particles (now represented by dots) is reflected in the number of their copies that conform $\widehat{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$.

Local linearised Distribution $\mathcal{N}(\mathbf{x}_t, \mathbf{\Sigma}_t^{\mathbf{xx}})$. This approach is based on defining the importance function by means of a Gaussian approximation of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. The idea is that at each iteration, for each i-th particle, is estimated a Gaussian approximation of $p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t}) \simeq \mathcal{N}(\hat{\mathbf{x}}_{t|t}^{(i)}, \mathbf{\Sigma}_{t|t}^{\mathbf{xx}(i)})$ to generate an importance sample. This approximation is obtained using a Kalman–based estimator applied on the state of each particle. This requires maintaining a covariance matrix for each particle, which evolves according to the Kalman equations. In general, algorithms based on this technique are commonly denoted as Kalman Particle Filters [41, 116].

Measurement–based Distribution $q(\mathbf{x}_t|\mathbf{z}_{1:t})$. In this case, importance samples are generated using information $\mathbf{z}_{1:t}$ extracted from the current image, which is related but not necessarily equal to $\mathbf{y}_{1:t}$. With this information an importance function is defined, which is sampled usually in combination with the prior distribution $p(\mathbf{x}_t|\mathbf{x}_{0:t-1})$ to generate the importance samples at each iteration. Examples of this way to proceed are given in [61, 96].

Remarks on Importance Sampling

One of the most important results of Monte Carlo theory is that the accuracy of estimations obtained from its methodology is independent of the dimensionality of the space sampled. Thus, for instance, the expectation of a given high dimensional variable $\mathbf{x}$

$$E[\mathbf{x}] \quad = \quad \int \mathbf{x} p(\mathbf{x}) d\mathbf{x} \ .$$

can be approximated satisfactorily just by obtaining a few dozen of independent samples of $p(\mathbf{x})$. That is

$$E[\mathbf{x}] \quad \simeq \quad \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)} \ . \tag{2.27}$$

Unfortunately, this result does not hold in algorithms based on importance sampling. In this cases, samples are generated from the importance function $q()$ instead of $p()$, what cancels out this so appealing property. The basic problem is that with the dimensionality, the variance of the weights associated to particles increases exponentially(see [79] for details). This means that the bigger the state dimension, the smaller the number of effective particles (i.e., with non–negligible weight w_t) representing the estimated distribution. As a consequence, an importance sampling estimate is totally dominated by a few samples with huge weights, what previously has been denoted as *degeneracy* problem. Thus, the availability of a good importance function (i.e., a $q()$ approximating very tightly $p()$) is more crucial the bigger the sampled space is. A consequence of this problem in practice is that to achieve a desired performance, an importance sampling algorithm in a high dimensional space requires a huge number of particles. For this reason it is commonly said that particle filters suffer from *the curse of dimensionality*.

2.4 Contour Measurement Process

Preceding sections have described the basic modules in estimation methods, assuming the availability of some measurements $\mathbf{y}_{1:t}$. It is time to describe how this measurements are extracted from images, that is, to define the *measurement process*. This is a key module in any vision–based tracking system, as usually it has to satisfy two important demands: obtaining a robust and accurate description of the object of interest, and providing this description within an small time interval. The information extracted then has to be correctly linked with the state to be estimated, in order to make a proper use of it. Concerning the tracking of contours, the information of interest to be obtained from the measurement process is the disparity between measured and predicted contours, what has been termed as *innovation* (ν) in Section 2.3.1. This can be obtained from

$$d^2(\mathbf{r}_f(s), \mathbf{r}(s)) = \frac{1}{L}\int(\mathbf{r}_f(s) - \mathbf{r}(s))^T(\mathbf{r}_f(s) - \mathbf{r}(s))ds \ , \tag{2.28}$$

which computes the mean square disparity between the predicted contour curve $\mathbf{r}(s)$ and the observed one $\mathbf{r}_f(s)$ (subindex f indicates that the curve is defined from image features), where s is the sampling parameter of the curve, and L the extent of its support. Unfortunately, there is one problem in the application of expression (2.28): it is sensitive to curve parameterisation. This means that the disparity between curve points sampled at s is valid only if both curves are parameterised according to a common reference frame (see Figure 2.7).

To deal with this problem, a criterion has to be defined to map samples of $\mathbf{r}_f(s)$ with the corresponding ones in $\mathbf{r}(s)$. One reasonable criterion is to select the map-

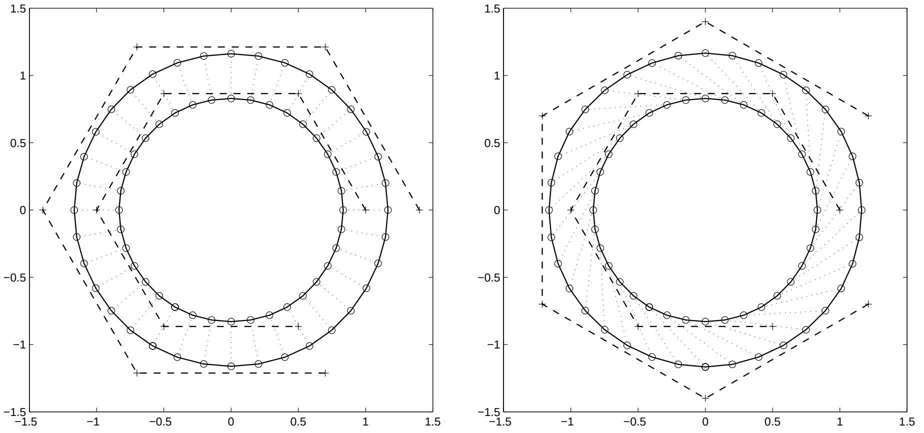


Figure 2.7: B–Splines parameterisation problem. Discrete point–to–point comparison between synthesised curves (*dotted links* between curves) rely on their parameterisation in a common reference frame (crosses between discontinuous lines indicate spline control points). Left and right plots show identical situations, which yield different distance computations (notice the different B-spline control points in the outermost curve).

ping that generates a minimum disparity between curves. That is, finding a value ϵ minimising

$$\frac{1}{L}\int(\mathbf{r}_f(s)-\mathbf{r}(s+\epsilon))^T(\mathbf{r}_f(s)-\mathbf{r}(s+\epsilon))ds \ . \tag{2.29}$$

Solving this problem globally is too complex to be useful for a real–time tracking application. However, this criterion can be applied locally, as long as the following assumptions hold:

- The offset between curve parameterisation is small (i.e., $\epsilon \approx 0$).
- Compared curves are continuous and smooth.

First assumption holds in tracking applications where the tracked contour performs slight changes between frames. The second one holds for the shape model used in this thesis shape, which assumes that real contours are precisely modelled using regular B-spline curves. Hence, given a sample of the measured contour $\mathbf{r}_f(s_i)$, the closest point of $\mathbf{r}(s)$ can be approximately found by the following process. Due to assumptions done, $\mathbf{r}(s)$ can be approximated by its first–order Taylor expansion at the point s_i, giving

$$\mathbf{r}_a(s) \quad \simeq \quad \mathbf{r}(s_i)+\frac{\partial\mathbf{r}(s_i)}{\partial s}(s-s_i) \ . \tag{2.30}$$

Using this approximation, the local disparity between curves at a given point s_i now is expressed as

$$d^2(\mathbf{r}_f(s_i),\mathbf{r}(s_i+\epsilon)) \simeq (\mathbf{r}_f(s_i)-\mathbf{r}_a(s_i+\epsilon))^T(\mathbf{r}_f(s_i)-\mathbf{r}_a(s_i+\epsilon)) \ ,$$

that is, an expression which depends only on the parameter ϵ. The minimum of this square distance accomplishes that

$$\frac{\partial(\mathbf{r}_f(s_i) - \mathbf{r}_a(s_i+\epsilon))^T(\mathbf{r}_f(s_i) - \mathbf{r}_a(s_i+\epsilon))}{\partial\epsilon} = 0 \ ,$$

that simplifies to

$$(\mathbf{r}_a(s_i+\epsilon) - \mathbf{r}_f(s_i))^T \frac{\partial\mathbf{r}_a(s_i+\epsilon)}{\partial\epsilon} = 0 \ . \tag{2.31}$$

From Equation (2.30) it follows that this expression is equivalent to

$$(\mathbf{r}_a(s_i+\epsilon) - \mathbf{r}_f(s_i))^T \frac{\partial\mathbf{r}(s_i)}{\partial s} = 0 \ . \tag{2.32}$$

Equation (2.32) determines that the vector that connects $\mathbf{r}_f(s_i)$ and $\mathbf{r}_a(s_i+\epsilon)$ with minimum distance must be perpendicular to the tangent of $\mathbf{r}(s)$ at s_i. In other words, this disparity vector has the same direction than the normal of $\mathbf{r}(s)$ at s_i, which is denoted as $\mathbf{n}(s_i)$. This result reflects what has been denoted classically as the *aperture problem*[4] [59]. To measure the disparity between curves, the information of interest is the module of the disparity vector $(\mathbf{r}_a(s_i+\epsilon) - \mathbf{r}_f(s_i))$, which, as Figure 2.8 shows, corresponds to

$$d(\mathbf{r}_f(s_i), \mathbf{r}(s_i)) \quad \simeq \quad \mathbf{n}^T(s_i)(\mathbf{r}_f(s_i) - \mathbf{r}(s_i)) \ ,$$

which is the normal displacement between corresponding points on the two curves.

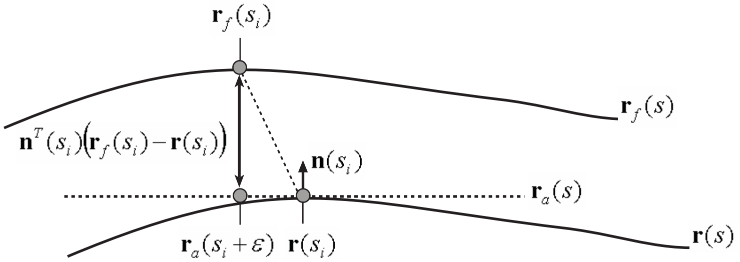


Figure 2.8: Normal displacement between curves. It provides a measure of difference which is approximately invariant to contour reparameterisation.

In practice, this disparity value has to be extracted from images, and an image processing algorithm with this aim is required. From Equation (2.32), it follows that

$$\mathbf{r}_a(s_i+\epsilon) - \mathbf{r}_f(s_i) \quad = \quad \alpha\mathbf{n}(s_i) \ , \tag{2.33}$$

[4] Using just local information, the association between contour points is inherently ambiguous. Under this situations, humans perceive an association perpendicular to the contours orientation.

where α is the required normal disparity value. From the assumption $\epsilon \approx 0$, Equation (2.33) derives to

$$\mathbf{r}(s_i) \quad = \quad \mathbf{r}_f(s_i) + \alpha\mathbf{n}(s_i) \ ,$$

which establishes that $\mathbf{r}(s_i)$ and $\mathbf{r}_f(s_i)$ are connected by a vector normal to $\mathbf{r}(s_i)$. Thus, an image processing method to approximate the normal displacement α just requires to detect edges along 1D measurement lines across $\mathbf{r}(s_i)$, with direction given by $\mathbf{n}(s_i)$.

Using this result, a norm is defined to determine the disparity between two curves, in terms of a third curve $\mathbf{r}_m(s)$ (measurement curve), assumed sufficiently close to both curves $\mathbf{r}_f(s)$ and $\mathbf{r}(s)$. This third curve is introduced to provide a method to establish an association between $\mathbf{r}_f(s)$ and $\mathbf{r}(s)$, independent of both. The procedure is as follows: given an image, the measurement curve $\mathbf{r}_m(s)$ is used to establish measurement lines normal to it, which are used to process the image and estimate the normal displacement between $\mathbf{r}_f(s)$ and $\mathbf{r}_m(s)$ (see Figure 2.9). The normal displacements obtained from the image processing are equivalent to the evaluation of $\mathbf{n}_m(s)^T(\mathbf{r}_f(s) - \mathbf{r}_m(s))$. Equivalently, this same measurement curve $\mathbf{r}_m(s)$ can be used to compute its normal displacement to the predicted curve $\mathbf{r}(s)$. If this normal displacement is identical to the obtained with $\mathbf{r}_f(s)$, this means that both curves are identical, and their norm–difference must be zero. Otherwise, the disparity between both vector reflects the disparity between $\mathbf{r}_f(s)$ and $\mathbf{r}(s)$ (see Figure 2.10).

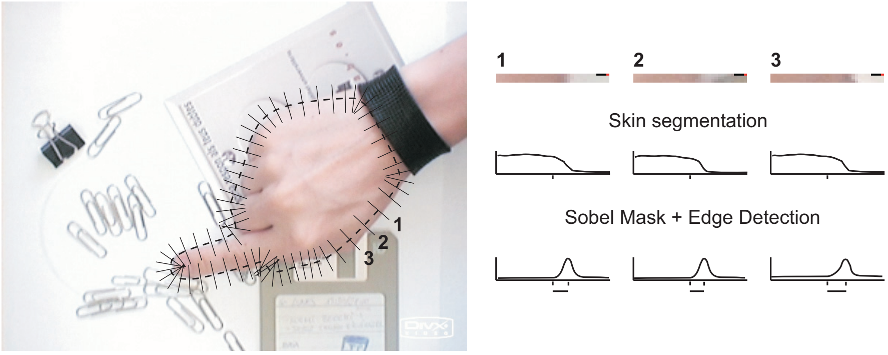


Figure 2.9: Contour measurement process. Image processing procedure to extract contour observations for real–time applications.

Thus, the norm–difference between curves is defined in terms of $\mathbf{r}_m(s)$ as:

$$\begin{aligned} d(\mathbf{r} - \mathbf{r}_f)_{\mathbf{n}_m} &= \frac{1}{L}\int \mathbf{n}_m^T(s)(\mathbf{r}_f(s) - \mathbf{r}_m(s)) - \mathbf{n}_m^T(s)(\mathbf{r}(s) - \mathbf{r}_m(s))ds \quad (2.34) \\ &= \frac{1}{L}\int \mathbf{n}_m^T(s)(\mathbf{r}_f(s) - \mathbf{r}(s))ds \quad (2.35) \end{aligned}$$

In practice, normal displacements between $\mathbf{r}_f(s)$ and $\mathbf{r}_m(s)$ are computed at dis-

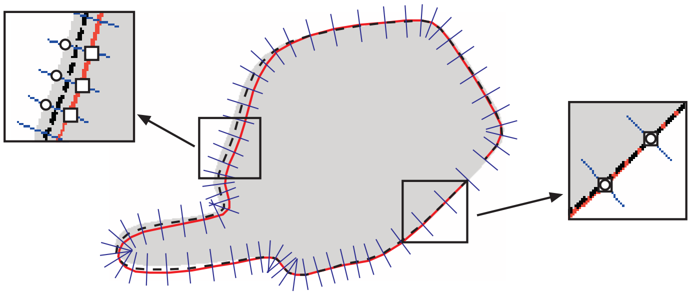


Figure 2.10: The use of the measurement contour (*dashed*). It evaluates the disparity between image contours (*circles*) and the contours of the predicted curve (*squares*).

crete sampling points $\{s_i\}_{i=1}^{N_s}$, so in this case (2.35) is approximated by the summation

$$d(\mathbf{r}-\mathbf{r}_f)_{\mathbf{n}_m} \quad \simeq \quad \frac{1}{N}\sum_{i=1}^{N}\mathbf{n}_m^T(s_i)(\mathbf{r}_f(s_i)-\mathbf{r}(s_i)) \ .$$

Once derived the expressions to compute the mean normal disparity between two curves, let's see how this translates into the expression of the *measurement process* of a contour tracking application. Observations $\mathbf{y}_t = \{y_i\}_{i=1}^{N_s}$ reflect the disparity between $\mathbf{r}_m(s)$ and the hypothetical curve $\mathbf{r}_f(s_i)$ determined by the image features, at each sampling point s_i. Thus,

$$y_i \quad = \quad \mathbf{n}_m^T(s_i)(\mathbf{r}_f(s_i)-\mathbf{r}_m(s_i)) \ . \qquad (2.36)$$

Equivalently, the observations expected to be measured (i.e., $\hat{\mathbf{y}}_{t|t-1}$) correspond to the disparity between $\mathbf{r}_{t|t-1}(s)$ and $\mathbf{r}_m(s)$ along $\mathbf{n}_m(s_i)$, where $\mathbf{r}_{t|t-1}(s)$ denotes the contour corresponding to the predicted state $\hat{\mathbf{x}}_{t|t-1}$. If $\mathbf{r}_m(s)$ is synthesised from parameters $\mathbf{x}_m$, then $\mathbf{y}_{t|t-1}$ can be expressed in terms of state space parameters as

$$\hat{\mathbf{y}}_{t|t-1} \quad = \quad \mathbf{N}_m^T\mathbf{U}(f(\hat{\mathbf{x}}_{t|t-1})-f(\mathbf{x}_m)) \ , \qquad (2.37)$$

where $f()$ is a function defined in terms of the shape space used, that transform an state vector $\mathbf{x}$ into spline control points $\mathbf{q}$. $\mathbf{U}$ is the matrix defined in (2.2) that translates $\mathbf{q}$ onto contour samples, and $\mathbf{N}_m$ is a $N_s \times N_s$ matrix maintaining the normal vectors $[n_x(s_i)\ n_y(s_i)]^T$ at contour sampling points as

$$\mathbf{N}_m = \begin{bmatrix} n_x(s_0) & 0 & \dots & 0 \\ 0 & n_x(s_1) & \dots & 0 \\ \vdots & \vdots & \dots & n_x(s_{N_s-1}) \\ n_y(s_0) & 0 & \dots & 0 \\ 0 & n_y(s_1) & \dots & 0 \\ \vdots & \vdots & \dots & n_y(s_{N_s-1}) \end{bmatrix} \ .$$

Thus, from (2.37) and using a linear shape space as the one in (2.3), the *noiseless* expression of the observation model corresponds to

$$\begin{aligned}\mathbf{y}_t &= \mathbf{N}_m^T\mathbf{U}((\mathbf{W}\mathbf{x}_t+\bar{\mathbf{q}})-(\mathbf{W}\mathbf{x}_m+\bar{\mathbf{q}}))\\ &= \mathbf{N}_m^T\mathbf{U}\mathbf{W}(\mathbf{x}_t-\mathbf{x}_m)\ .\end{aligned}$$

For a non–linear shape space as the one in (2.5) the following non–linear expression is obtained

$$\begin{aligned}\mathbf{y}_t &= \mathbf{N}_m^T\mathbf{U}((\mathbf{W}_{(\mathbf{x}_t)}\bar{\mathbf{q}}+\mathbf{T}\mathbf{x}_t)-(\mathbf{W}_{(\mathbf{x}_m)}\bar{\mathbf{q}}+\mathbf{T}\mathbf{x}_m))\\ &= \mathbf{N}_m^T\mathbf{U}((\mathbf{W}_{(\mathbf{x}_t)}-\mathbf{W}_{(\mathbf{x}_m)})\bar{\mathbf{q}}+\mathbf{T}(\mathbf{x}_t-\mathbf{x}_m))\ .\end{aligned}$$

It lacks to define how $\mathbf{r}_m(s)$ is established in each measurement process. The usual approach is to synthesise it from the prediction of the currently tracked contour at that instant (i.e., $\hat{\mathbf{x}}_{t|t-1}$).

2.4.1 Kalman Filter Measurement Process

The measurement process to obtain the normal displacement between a contour prediction and the contour in the image, is based on the presented method of processing 1D measurement lines on the image. In many practical implementations, the length of this measurement line is fixed to a given value, expecting to account for the error in the contour prediction. However, in estimation methods like the Kalman filter, jointly with the prediction of the most probable contour shape $\mathbf{x}_{t|t-1}$, it is available the uncertainty in this prediction (i.e., the predicted state covariance $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}$) . This uncertainty delimits an image region where the real contour should be, according to the modelisation of the problem. This information can be used to automatically control the spatial scale for searching image contours, as proposed in [17]. The objective is constraining the search region to the minimum reasonable size, in a way that correct contours can still be detected, while edges from image clutter are less likely to be included in extracted observations. This is done in the following way. Let's assume an state vector $\mathbf{x}$ maintaining the parameters $\mathbf{c}$ of a shape model. The Kalman state prediction given by $\mathcal{N}(\hat{\mathbf{x}}_{t|t-1},\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1})$ can be used to estimate the predicted contour in the image, as well as the covariance at any location s along this contour given by

$$\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)=\mathbf{U}(s)\mathbf{W}\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{W}^T\mathbf{U}(s)^T\ . \tag{2.38}$$

This covariance defines a validation gate in a spatially distributed fashion. For each contour sample, a 2D covariance delimits the most likely region where its real value may lie. The proposal in [17] exploits this covariance to establish the length of the measurement line used to detect the contour. Since what is measured is the normal component of the disparity between the predicted and the real contour, the length of this measurement line should be constrained to the maximum value that this normal component is expected to take. This is computed by

$$\sigma_n^2(s) \quad = \quad \mathbf{n}_m^T(s)\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)\mathbf{n}_m(s)\ ,$$

where $\mathbf{n}_m(s)$ specifies the direction of the normal measurement line. Using the rule of 3σ, this variance is used to specify the extremes of the measurement line as

$$\mathbf{r}_m(s) \pm 3\sigma_n(s)\mathbf{n}_m(s) \ , \tag{2.39}$$

which encloses the region where the normal displacement of the contour should be measured (at a level of confidence around 98%).

The proposed mechanism of taking advantage of $\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)$ to control search scale suggested the study in this thesis of a method to design a more proper measurement extraction process. Effectively, the contour covariance at a given contour point provides very significant information about the more likely region where the searched contour may lie. Thus, it seems reasonable to use this information to stablish *better* measurement directions along which to look for contours, instead of the *blind* normal direction presented previously. Unfortunatelly for us, in the early stages of the development of this idea, a reviewer informed us that a similar proposal had already been done in [10]. This generally overlooked reference provides an elegant way to take this concept into account. The inverse of the positional covariance at a given contour point defines a Mahalanobis distance metric, which can be used to redefine the measure of the local disparity between curves

$$d^2(\mathbf{r}_f(s), \mathbf{r}(s)) = \frac{1}{L}\int (\mathbf{r}_f(s) - \mathbf{r}(s))^T (\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s))^{-1}(\mathbf{r}_f(s) - \mathbf{r}(s))ds \ .$$

Making the same assumptions previously exposed, the local disparity between curves is approximated as

$$d^2(\mathbf{r}_f(s_i), \mathbf{r}(s_i)) \simeq (\mathbf{r}_f(s_i) - \mathbf{r}_a(s_i+\epsilon))^T (\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s_i))^{-1}(\mathbf{r}_f(s_i) - \mathbf{r}_a(s_i+\epsilon)) \ , \tag{2.40}$$

whose minimum requires that

$$(\mathbf{r}_f(s_i) - \mathbf{r}_a(s_i+\epsilon))^T (\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s_i))^{-1}\frac{\partial \mathbf{r}(s_i)}{\partial s} \quad = \quad 0 \ .$$

From previous equation, the direction of vector $(\mathbf{r}_f(s_i) - \mathbf{r}_a(s_i+\epsilon))^T (\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s_i))^{-1}$ equals to the normal to the contour, which from the assumption $\epsilon \approx 0$ yields a measurement line connecting $\mathbf{r}_f$ and $\mathbf{r}_a$ given by

$$\mathbf{r}_f(s_i) \quad = \quad \mathbf{r}(s_i) + \alpha\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s_i)\mathbf{n}(s_i) \ ,$$

where α is an arbitrary scalar. The result obtained states that the direction of measurement is computed now from $\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)\mathbf{n}(s)$, given by

$$\mathbf{m}(s) = \frac{\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)\mathbf{n}(s)}{|\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)\mathbf{n}(s)|} \ .$$

Notice that if $\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)$ equals to $a\mathbf{I}_2$ with $a > 0$, it defines a circular validation gate, and the direction of measurement $\mathbf{m}(s)$ corresponds to the one indicated by $\mathbf{n}(s)$, reflecting that in this case the aperture problem situation is not eluded (no additional information is provided by $\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)$).

The direction vector $\mathbf{m}(s)$ is usually close to the principal axis of the uncertainty ellipse provided by the positional covariance (see Figure 2.11). Thus, [10] proposes to determine the length of measurement lines by constraining them within the uncertainty ellipse corresponding to this positional covariance. This is achieved by choosing a length value $\rho(s_i)$ so that points in the measurement line lie within a Mahalanobis distance of three standard deviations from the center of the ellipse, i.e.,

$$\rho(s)(\mathbf{m}^T(s)(\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s))^{-1}\mathbf{m}(s))^{\frac{1}{2}} = 3 \ . \tag{2.41}$$

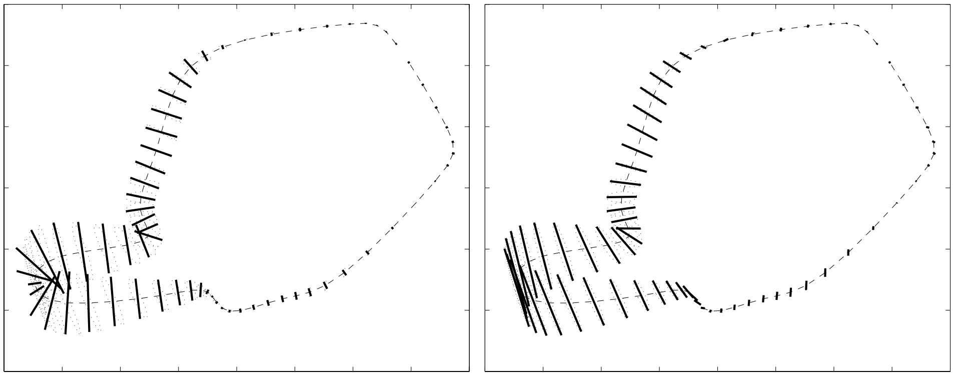


Figure 2.11: Orientation of the measurement lines. Normal(left) and Learned (right) measurement lines, with their length adjusted according to $\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}(s)$.

This search direction proposal is denoted in the following as *Learned* or *Baumberg*[5] direction. In [10] it is said that this direction improves the performance of a KF, compared with the use of the normal search direction. However, this is supported by a very reduced experimental work, where other additional proposals are also evaluated. In order to ascertain this claimed superiority, we contribute in section 2.5.1 with a performance evaluation of both search direction proposals, in an exhaustive experimental work done.

2.4.2 Particle Filter Measurement Process

In practical terms, the way how predicted contours extract measures from images in Particle Filters does not differ from the Kalman filter methodology. Typically, measurement lines are established normal to the contour, selecting the edge closer to the contour prediction. The difference is that while in a Kalman Filter a single measurement vector $\mathbf{y}_t$ is extracted using commonly the mean contour of the predicted Gaussian state distribution as measurement contour, in a particle filter each particle extracts its *own* measurement vector. This fact is of essential relevance, as intrinsically defines which form of $p(\mathbf{y}_t|\mathbf{x}_t)$ is theorically assumed. In a Kalman Filter it is assumed that the real contour is effectively extracted from the image, distorted just by Gaussian

[5] From the surname of the author who first proposed it.

noise. Thus, the contour detected at the N measurement lines define an observation process given by

$$p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{\nu_{t_i}^2}{2\sigma^2} \tag{2.42}$$

where $\nu_{t_i} = y_{t_i} - \hat{y}_{t_i}$ is the disparity along the i-th measurement line between the detected contour and the contour synthesised from $\mathbf{x}_t$. Thus, the more likely an state $\mathbf{x}_t$ is, the closer the ν_{t_i} values are to zero.

In the case of particle filters, the measurement processes realized for the multiple particles implicitly modelise a multimodal density function of $p(\mathbf{y}_t|\mathbf{x}_t)$. As the contours corresponding to particles are synthesised at different image locations, the image contours detected in their corresponding measurement lines may be different, which means that the likelihood of particles is evaluated with a function $p(\mathbf{y}_t|\mathbf{x}_t)$ of multiples modes. This is decidedly a model that better adjusts to the characteristics of the observation process, since the presence of multiple contours in the image is considered. A graphical comparison of the Kalman and the Particle filter observation models is given in Figure 2.12.

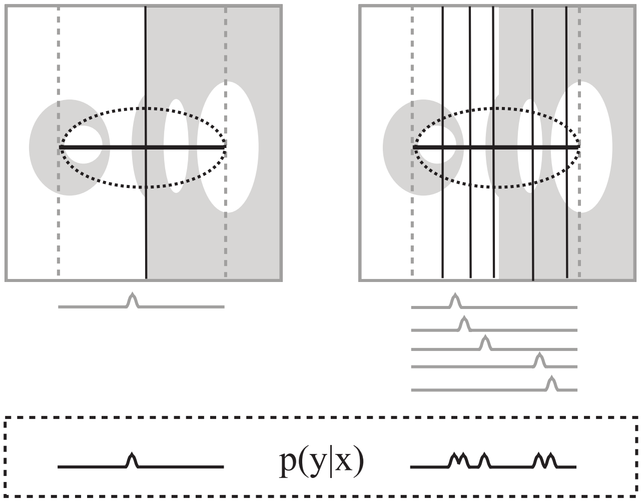


Figure 2.12: Observation model. Differences between the Kalman (*left*) and the Particle Filter (*right*) observation model. For Particle Filters, different measurement contours (vertical black lines) are evaluated, distributed according to the predicted state uncertainty. Taking each time the closer contour as observation, this derives in the construction of a multimodal observation model.

In [77, 76] are proposed analytic expressions for the $p(\mathbf{y}_t|\mathbf{x}_t)$ inherent in PFs, explicitly representing the presence of clutter in the image (i.e., contours in the image that do not correspond to the tracked shape), as well as the possibility that the tracked contour is occasionally missdetected. The most popular formulation used is denoted as the *Poisson likelihood.* This model assumes that clutter features on a 1-dimensional measurement line obey a Poisson law with density λ, and that the probability of detecting and missdetecting the tracked contour are respectively q_{11} and q_{01}. From this assumptions results the likelihood function

$$p(\mathbf{y}_t|\mathbf{x}_t) = \left(\prod_{i=1}^{N} \frac{e^{-\lambda L} \lambda^{n_i}}{n_i!} \right) \prod_{j=1}^{N} \left(q_{01} + \frac{q_{11}}{\lambda} \sum_{k=1}^{n_i} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z_k^{(j)} - \hat{y}_{t_j})^2}{2\sigma^2} \right) \right) \quad , \tag{2.43}$$

where $z_k^{(j)}$ denote n_i edges assumed detectable along a measurement line of length L. Although this likelihood expression is quite complex, in practice it can be computed efficiently. The first factor in (2.43) is constant, and its computation can be avoided. The sum in the second factor is determined mainly by its largest element, since σ is usually small and other terms rarely have a significant contribution. Thus, in practice, this likelihood can be approximated by

$$p(\mathbf{y}_t|\mathbf{x}_t) \propto \prod_{i=1}^{N} \left(q_{01} + \frac{q_{11}}{\sqrt{2\pi}\sigma\lambda} \exp\left(-\frac{\nu_{t_i}^2}{2\sigma^2} \right) \right) \quad , \tag{2.44}$$

where ν_{t_i} is the distance of the contour to the closer edge in the image, for the i-th measurement line. This is the likelihood function proposed in the original Condensation paper [60], and simply corresponds to define an specific $p(\mathbf{y}_t|\mathbf{x}_t)$ function for each particle $\mathbf{x}_t^{(i)}$, from its *own* edge measurements $\mathbf{y}_t^{(i)}$. Thus, for each particle there is a functional very similar to (2.42), except for the terms q_{01}, q_{11} and λ, that explicitly account for occasional missdetections of the tracked contour.

2.5 Contour Tracking in a Case–Study

Up to this point, a fast review of the essential points in a visual contour tracking application has been presented, namely

- models to represent a contour shape and its dynamic behaviour,
- methods that fuse these models with observations to estimate the contour state at each frame of a sequence,
- strategies to extract measurements from images in real–time tracking applications.

Multiple alternatives has been presented for each one of these tracking building blocks. From the description done, one may guess which combination should generate the best tracking performance: the one with the more accurate model, with the estimation method requiring less a priori assumptions to be fulfilled, and with the more robust observation process. However, this kind of decision is based inherently in considering

- the worst case tracking scenario (i.e., performing tracking in very challenging sequences),
- the availability of unlimited computational resources.

Real tracking applications commonly do not correspond to neither of these two assumptions. On the one hand, it is obvious that a real tracking application will impose constraints in the computational resources available. On the other, for a given application it may be possible to condition the tracking environment in order to obtain good quality sequences. Thus, one has to consider these points when the solution to a tracking application is designed. Our experimental work described in this section tries to bring some light to this point. The major implementation issues concerning a contour tracking application are evaluated in multiple experiments, using test sequences with different degrees of noise, and under different computational constraints (when relevant). Our objective has been providing a fair comparison between the different alternatives, showing the pros and cons of them. The experiments realized have been designed in the following way:

First, we have acquired several sequences, showing the top view of a hand with pointing finger displaying a representative set of feasible hand configurations. Then, the outline of the hand at each frame has been annotated, generating in that way examples used to train a shape and a dynamic model (AR(2)). With these models, we have generated different synthetic[6] sequences, where to evaluate next the performance of different estimation methods and observation processes. Generated sequences simulate the ones of a hand with pointing finger, which has been isolated from the rest of elements in the image by means of an ideal skin segmentation process. One important advantage of processing these synthetic sequences, is that tracking algorithms use the same shape model of the sequence generation. In that way, the quality of the shape model is not a factor that can alter the performance of algorithms, as always the *perfect* model is used. Another obvious advantage is that the *ground truth* shape parameters used to generate sequences are available, and they can be used to quantify the tracking performance.

To observe how robust is the performance of algorithms under noisy conditions, sequences generated are distorted with different artifacts prior to its processing. The tracking performance evaluation methodology is the following. Given the sequence to be analysed, it is first distorted with random artifacts, according to a desired signal–to–noise ratio (SNR). This sequence is then processed by a tracking algorithm, obtaining the shape state at each frame. To quantify if this estimation is accurate, an *Image–based* and a *Contour–based* method are simultaneously applied (see Figure 2.13). Essentially, the image–based method quantifies the tracking performance from the degree of overlap between the original sequence (i.e., without distortion) and a sequence generated from the estimated contour parameters. The accuracy achieved is expressed by means of a SNR value, which is higher the better the tracking performance. The contour–based method measures the average disparity between the ground truth contour shown in the sequence, and the one estimated by the tracking algorithm (i.e., the Mean Contour Error (MCE)). A detailed description of both quantification methods is provided in Appendix D, where their complementarity is remarked.

[6] Real sequences have also been used to observe qualitatively the performance of the different proposals. However, the study presented focus just on synthetic sequences, as allow to obtain objective measurements of the achieved performance.

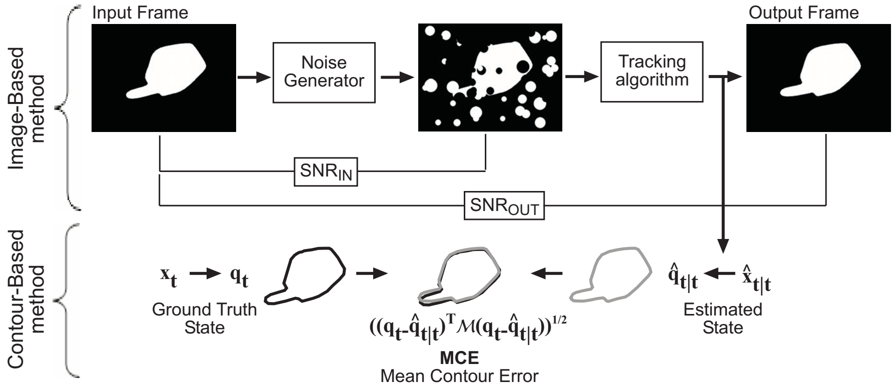
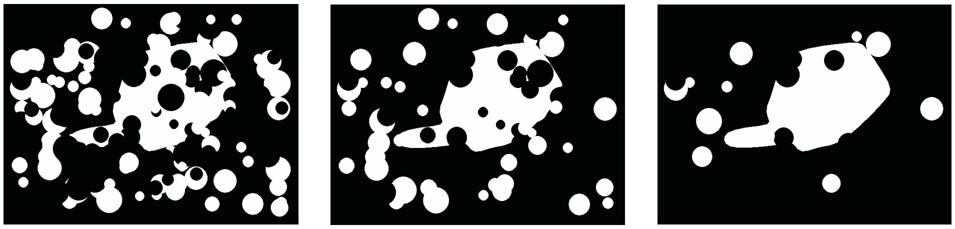


Figure 2.13: Performance quantification procedure. SNR_{IN} measures the degree of distortion of the input to the tracking algorithm. SNR_{OUT} quantifies the disparity between the tracking output, and the ideal output (i.e., the undistorted input sequence). MCE measures the mean contour disparity between the ground truth contour and the estimated one.

To obtain an statistical view of the performance of the analysed tracking methods, a hundred noisy sequences are generated for each level of noise distorting the input sequence considered. Figure 2.14 gives some examples of different SNR_{IN} situations considered in the experiments. All trackers are evaluated on these sequences, sharing the same shape and dynamical models. To model dynamics, a simple AR(1) model has been used, while synthetic sequences have been generated using a second order model. This has been done that way in order to evaluate methods in situations where the dynamical model roughly corresponds with the real object behaviour, which commonly happens in real applications.



$SNR_{IN} = 6dB$ $SNR_{IN} = 8dB$ $SNR_{IN} = 13dB$

Figure 2.14: Examples of noise in frames, for different situations considered in the experiments.

Plots are generated displaying the performance achieved by two different evaluated methods, in a way that their respective performance can be easily compared. For instance, plots are generated comparing different proposals to extract observations from frames, different estimation methods, etc. The performance measured using the image–based method is displayed using a boxplot representation, while the median of the performance measured with the contour–based method is shown in an attached table. Figure 2.15 describes briefly the information provided in the generated performance plots. A more detailed explanation of the graphical elements used in this thesis to display tracking results is provided in Appendix D.

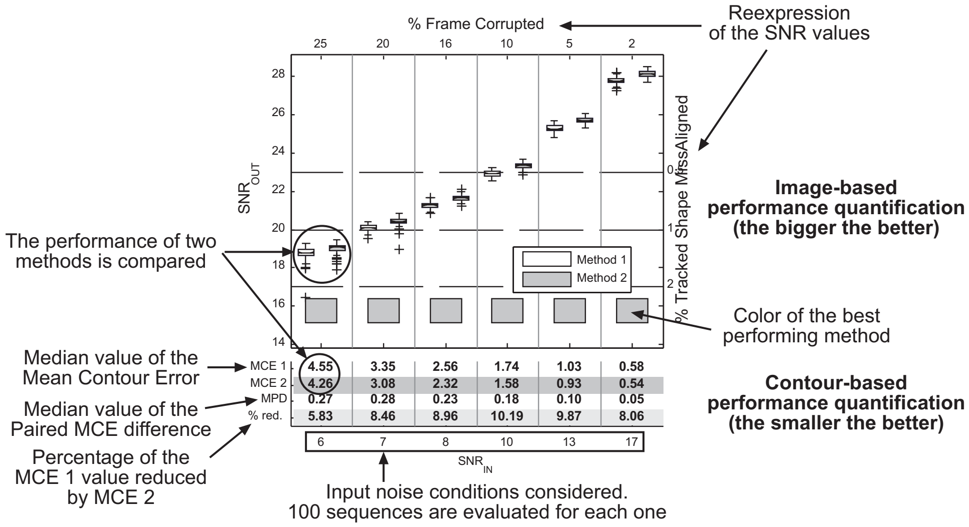


Figure 2.15: Proposed representation of the evaluation results.

The sequences used in the experiments are of two types:

- sequences showing local deformations of the tracked shape;
- sequences showing rigid transformations of the overall shape.

The states estimated for each type of sequence are denoted respectively as $\mathbf{x}_D$ and $\mathbf{x}_R$, with dimensionality 2 and 5. $\mathbf{x}_D$ maintain the parameters of a linear shape space as the presented in Section 2.1.1, while $\mathbf{x}_R$ maintain the global affine transformation parameters in Section 2.1.2, which relate non–linearly with the tracked shape. Thus, estimation algorithms concerning linear and non–linear models will be evaluated. Each one of the analysed sequences is constituted by 750 frames, which correspond to a 30 seconds sequence acquired by a standard video camera. Table 2.1 details the system and the observation processes used in the estimation algorithm. The *noise* term in the observation model corresponds to Gaussian noise $\mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ for Kalman–based algorithms, and non–Gaussian noise for Particle Filters.

2.5.1 Measurement Process Evaluation

Our first experiments have been focused to discern which is the contour measurement process leading to a better tracking performance. Accordingly, in the following it is quantified how the direction of measurement lines and the method used to establish their lengths affects the performance of KFs and PFs. This study has been done in a sequence showing only local shape deformations.

Table 2.1: Models used in each type of processed sequence.

	Local Transformations Estimation
State	$\mathbf{x} = [\mathbf{c}_0 \ \mathbf{c}_1]^T$
System Model	$\mathbf{x}_t - \bar{\mathbf{x}} = \mathbf{A}(\mathbf{x}_{t-1} - \bar{\mathbf{x}}) + \mathbf{B}\mathbf{w}_t$
Observation Model	$\mathbf{y}_t = \mathbf{N}_m^T \mathbf{U}\mathbf{W}(\mathbf{x}_t - \mathbf{x}_m) + \text{noise}$

	Global Affine Transformations Estimation
State	$\mathbf{x} = [t_x \ t_y \ s_x \ s_x \ \theta]^T$
System Model	$\mathbf{x}_t - \bar{\mathbf{x}} = \mathbf{A}(\mathbf{x}_{t-1} - \bar{\mathbf{x}}) + \mathbf{B}\mathbf{w}_t$
Observation Model	$\mathbf{y}_t = \mathbf{N}_m^T \mathbf{U}((\mathbf{W}_{(\mathbf{x}_t)} - \mathbf{W}_{(\mathbf{x}_m)})\bar{\mathbf{q}} + \mathbf{T}(\mathbf{x}_t - \mathbf{x}_m)) + \text{noise}$

The Measurement Process in KFs

In this experiment we evaluate the performance of a KF tracker using the normal and Learned measurement lines. Both approaches are evaluated for the case where the measurement length is fixed a priori, as well as for the case where the length is adjusted according to the covariance of the predicted contour location. Results are shown in Figure 2.16.

For fixed length measurement lines, the Learned direction achieves a slightly better performance. It reduces a $5 - 10\%$ the MCE obtained with the normal direction, slightly improving the average contour alignment in around 0.25 pixels. This better performance may come from the fact that, although both compared directions can extract with the same probability measurements which corresponds to clutter, the information deduced from normal measurement lines can suggest contour displacements more unfeasible than the ones suggested for learned measurement lines (specially in zones where the Baumberg and the normal direction are almost perpendicular. See Figure 2.17).

The use of an adaptive validation gate provides a remarkable improvement on the tracking performance, with respect to the fixed length case. A shorter measurement line means decreasing the probability of extracting features corresponding to clutter, being in that way less distracted from it. Results also show that the use of the Learned measurement direction does not contribute in an improvement of the tracking performance worth to be considered. While in low noise situations a minor improvement in the tracking accuracy is obtained, as long as clutter increases the performance of both approaches is put on the same level, and for extremely noise situations, the normal measurement direction performs even better. This contradicts the assertions previously done in [10], an demands an explanation of why this happens. We found out that the reason lies on the method used in [10] to establish the length of measurement lines (Equation (2.41)), which is very unappropiate for the case of *normal*

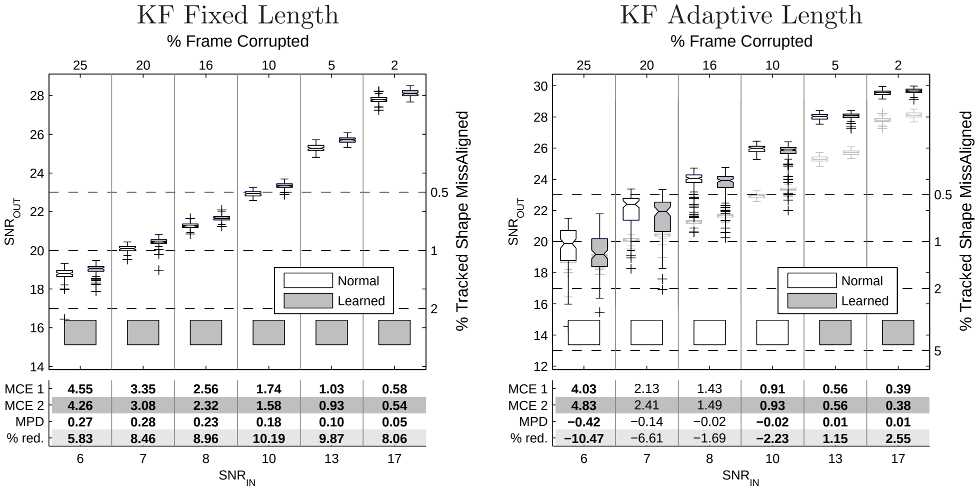


Figure 2.16: KF performance depending on the measurement process. Left: measurement lines of fixed length (20 pixels). Right: measurement lines with a length according to the contour covariance (in gray are shown the fixed length results).

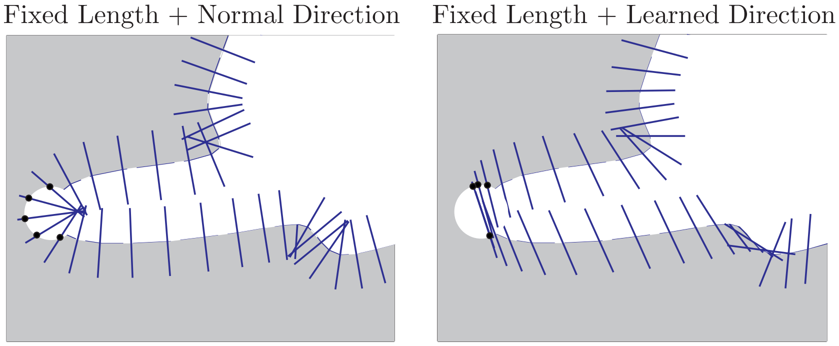

Fixed Length + Normal Direction
Fixed Length + Learned Direction

Figure 2.17: Problems with fixed length measurement lines. The misslocation errors using normal measurement lines are further more critical.

directions. In cases where the direction of $\mathbf{n}(s)$ differs considerably from the principal axes of $\mathbf{\Sigma}^{\mathbf{rr}}_{t|t-1}$, the length of the measurement line can be too short to detect any contour in the image (see Figure 2.18). Thus, normal measurement lines are prone to missdetect edges, achieving a worse tracking behaviour. Hence, the way how both methods were compared in [10] was unfair. In our experiments, since the length of measurement lines for the normal case is determined following Equation (2.39), this edge missdetection problem is eluded. Figure 2.18 shows the tracking performance achieved using normal measurement lines, comparing the two described methods to establish their lengths. The method based on Equation (2.39) is clearly superior.

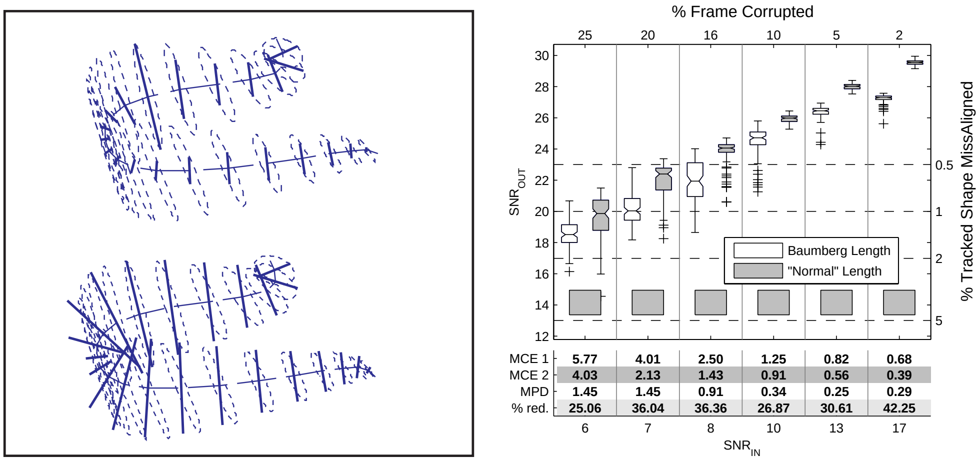


Figure 2.18: Normal measurement lines and their length. Left: length of the normal measurement lines established according to both checked proposals. The use of the method in [10] in normal measurement lines (top example) is clearly unfair. Right: Kalman performance using normal measurement lines, using the two methods to establish the length of measurement lines.

Hence, our results show not only that the normal measurement direction performs as good as the learned one, but also that in high noise scenarios it performs even better. The explanation given to the fact that both methods perform very similarly in most of cases is that, in fact, both measurement direction proposals explicitly take into account that the real displacement between contours is not obtained, but the projection of this real displacement onto a given measurement direction. Therefore, from this point of view, the effect of the direction of measurement in obtaining *more informative* observations is practically irrelevant. On the other hand, the reason why the normal measurement direction performs better in high noise sequences is not so clear. One plausible hypothesis could be that a tracker based on the learned measurement direction takes this wrong information as more confident, because the direction between neighbouring measurement lines is significantly correlated. Thus, it is more likely that a given contour zone is affected by the same clutter artifact, giving more credibility to the wrong information extracted. For the case of the normal measurement direction, each contour zone is misleaded by the closer artifact, so between neighbouring lines the detected clutter elements are more likely to be different, providing a more contradictory information of the contour location, and hence, reducing the reaction of the tracking algorithm to this wrong information. This situation happens rarely in low noise sequences, but as long as more artifacts distort frames, it is more likely that this situation occurs, penalising the performance achieved using learned measurement lines. Figure 2.19 shows an example of the problem described.

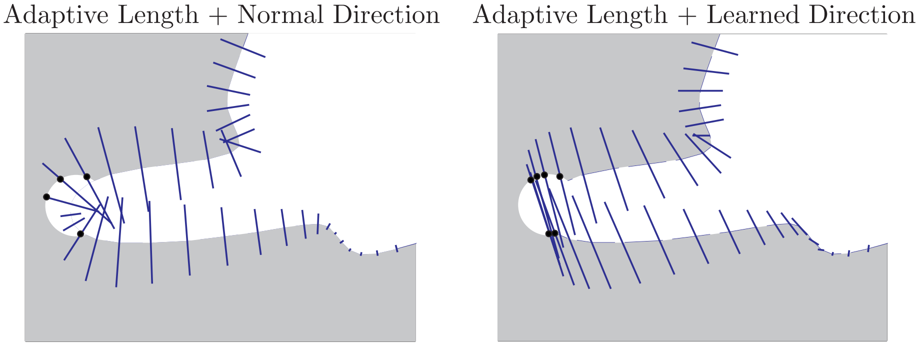
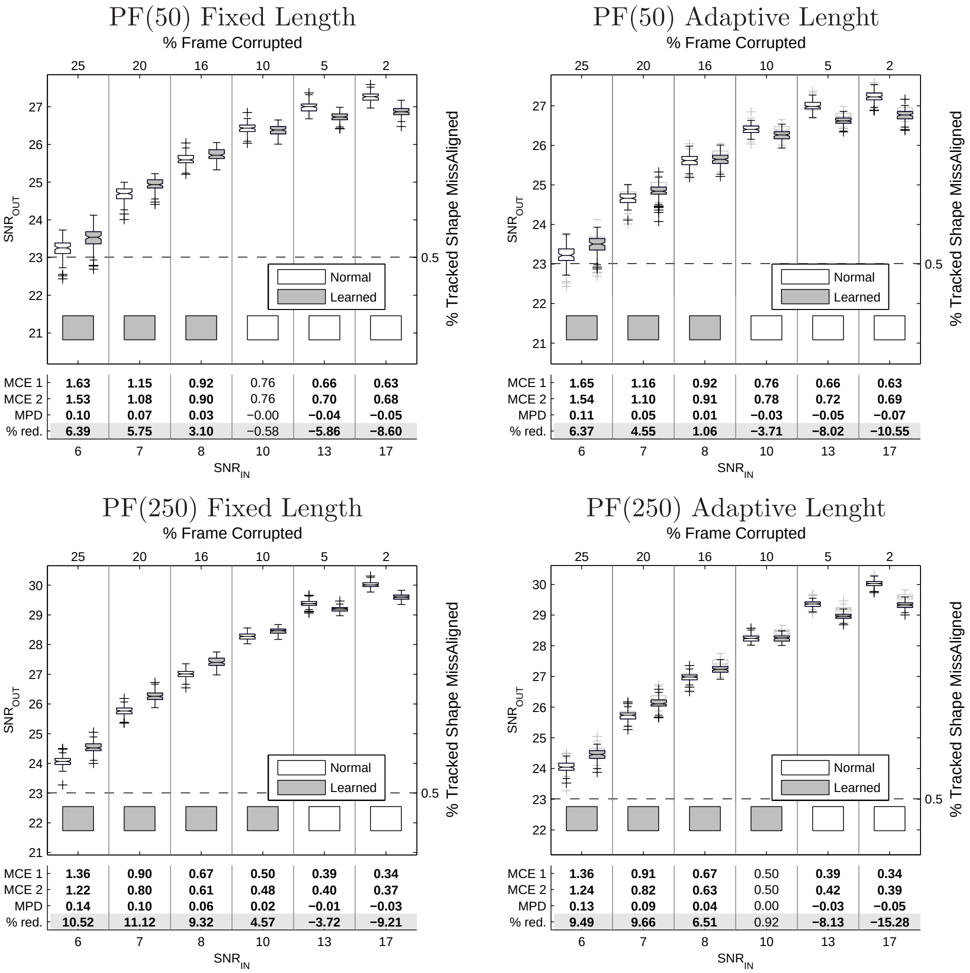

Adaptive Length + Normal Direction
Adaptive Length + Learned Direction

Figure 2.19: Problems with adaptive length measurement lines. The correlation between learned measurement lines leads in some cases to a major amount of edge misslocations. As these misslocations are *consistent*, they disrupt more notably the tracking performance.

The Measurement Process in PFs

Our tests concerning PFs evaluate the performance of the Condensation algorithm, that is, an implementation of the SISR algorithm which uses the prior target dynamics as importance function $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{0:t-1})$. The essence of the measurement process in PFs (see Section 2.4.2) is the computation of the likelihood of each particle, done in terms of the proximity of the contour of each particle to edges in the image. The experiments done quantify how this process is affected by the length and direction of the measurement line. In the case of PFs, the direction of measurement lines is determined from the uncertainty in their dynamical model (i.e., its stochastic part defined by $\mathbf{Q}_t$ in Equation (2.15)). The obtained results are shown in Figure 2.20, for two different amounts of particles considered (50 and 250 respectively).

Results show, differently to KFs, that there is no clear advantage on making an adaptive control of the length of measurement lines. Using the normal direction, the performance achieved with fixed and adapted measurement line lengths is practically identical. This is because just edges very close to the predicted contour has a significant contribution to the likelihood term used to weight particles (Equation (2.44)). Edges from image clutter detected due to processing a too large measurement line contribute nearly the same as if they were undetected, so there is no need to implement a fine control of the measurement line length. Using the learned direction, it is observed that in low noise situations, fixed length measurement lines perform even slightly better. This is due to the fact that measurement lines with a direction



PF(50) Fixed Length

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	1.63	1.15	0.92	0.76	0.66	0.63
MCE 2	1.53	1.08	0.90	0.76	0.70	0.68
MPD	0.10	0.07	0.03	−0.00	−0.04	−0.05
% red.	6.39	5.75	3.10	−0.58	−5.86	−8.60

PF(50) Adaptive Lenght

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	1.65	1.16	0.92	0.76	0.66	0.63
MCE 2	1.54	1.10	0.91	0.78	0.72	0.69
MPD	0.11	0.05	0.01	−0.03	−0.05	−0.07
% red.	6.37	4.55	1.06	−3.71	−8.02	−10.55

PF(250) Fixed Length

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	1.36	0.90	0.67	0.50	0.39	0.34
MCE 2	1.22	0.80	0.61	0.48	0.40	0.37
MPD	0.14	0.10	0.06	0.02	−0.01	−0.03
% red.	10.52	11.12	9.32	4.57	−3.72	−9.21

PF(250) Adaptive Lenght

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	1.36	0.91	0.67	0.50	0.39	0.34
MCE 2	1.24	0.82	0.63	0.50	0.42	0.39
MPD	0.13	0.09	0.04	0.00	−0.03	−0.05
% red.	9.49	9.66	6.51	0.92	−8.13	−15.28

Figure 2.20: PF performance depending on the measurement process. Fixed length (left) and adaptive length (right) measurement lines, for the normal and learned measurement direction, in particle filters with 50 (top) and 250 particles (bottom).

close to the contour tangent are prone to miss-detect contours if their length is very small[7]. This reduces the likelihood of some evaluated particles, with respect to using a fixed measurement line length. In low noise situations, this penalises (although very slightly) the tracking performance more than the few noise artifacts that are measured due to using a fixed measurement line length.

Concerning the measurement line direction, results obtained manifest that the Learned direction performs slightly worse in low noise sequences, while it improves the performance in high noise ones. Quite surprisingly, this is just the opposite behaviour than the one observed in KFs. Insight on this behaviour has been obtained by analysing the frame–by–frame performance of PFs. In low noise situations, it is observed that at the end of each iteration, the number N_{eff} of effective particles (see Equation (2.26)) of the PF is inferior if the Learned measurement direction is used. This is due to the fact that extracting contours along the Learned direction implicitly means evaluating the disparity between predicted and observed contours using a Mahalanobis distance (see Section 2.4.1), and this implies being more selective in giving a high likelihood weight to particles close to the observed contour. As a consequence, the tracking output is determined from an effectively inferior number of particles, and the resulting contour adjusts less precisely around the ground truth contour. If the number of particles in the PF are increased, the median paired difference between the performance achieved using the normal and Learned measurement direction is reduced.

In high noisy situation the Learned measurement direction performs better for the following reason. The more clutter there is in a frame, the higher the likelihood of a *wrong* particle can be. The edges of distracting elements in frames match the ones of misplaced contours of wrong particles, and the probability of misstrack due to this factor increases. This problem is less serious using the learned measurement line, because the region of the video frame effectively used to evaluate the likelihood of particles is much constrained than the one used by the normal measurement direction (see Figure 2.21). Processing a better adjusted region of interest means avoiding more distracting elements in the frame, and in that way decreasing the possibility of misstrack . However, the use of the learned measurement direction has an important drawback that is not reflected in the performance evaluation plot. The computational time needed to process each particle increases notably, because to establish their respective measurement lines a bigger amount of computation is needed. This reduces the number of particles that can be used in a PF, if a given response time constraint has to be fulfilled. In practice, a better performance is achieved inspecting a less constrained image region using a bigger number of particles, so in the case evaluated is more important *quantity* than *quality*, because the cost of quality is too high (see Figure 2.21).

[7] Due to discretisation effects, in these cases the measurement line pixels commonly lay all in foreground pixels, preventing the contour detection. Using the normal measurement direction this is less likely to happen.

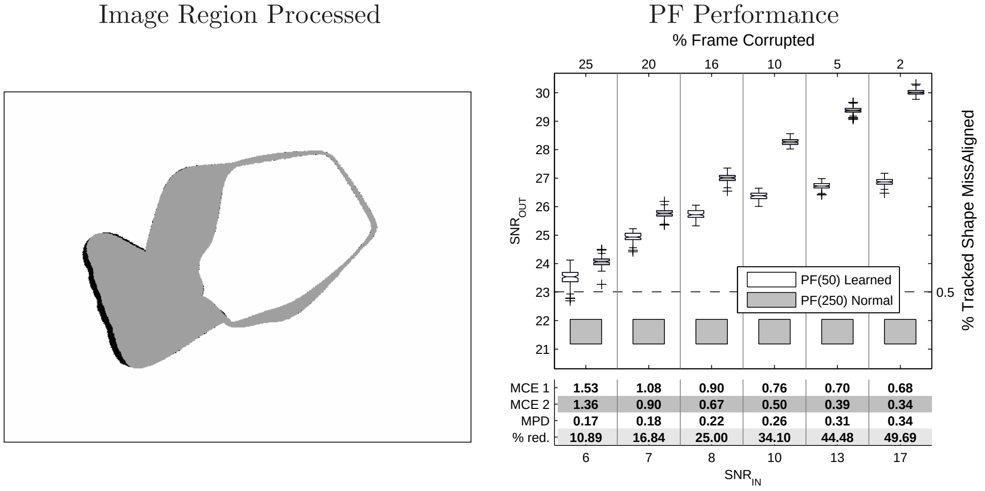


Figure 2.21: PFs and the orientation of measurement lines. Left: image zone used to evaluate the performance of particles. In grey, region common for the learned and normal measurement direction. In black, zone inspected only if the normal measurement direction is considered. Right: PFs with different measurement process and amount of particles. The advantages of using a bigger number of particles overcome the ones of using a learned measurement direction.

Summary

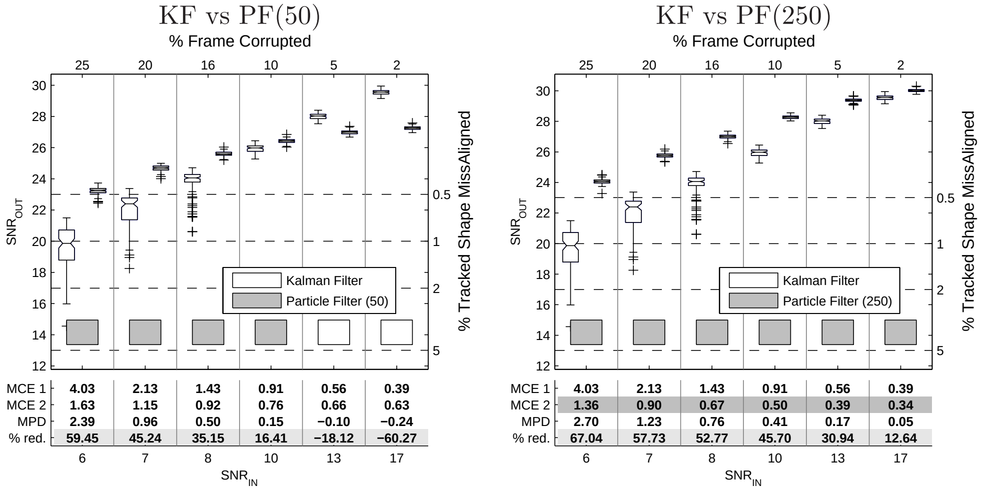
In the experimental work described, we have evaluated the performance of different methods to extract observations from from frames, identifying the more appropriate measurement processes to be used by each estimation algorithm in the following experiments. Concerning Kalman–based algorithms, the normal measurement direction will be used, adjusting the length of measurement lines according to the predicted contour covariance. Concerning PFs, fixed length measurement lines will be used, using also the normal measurement direction. Although the Learned direction improves the tracking accuracy of PFs, it is computationally cheaper achieve the same results by just using a bigger amount of particles.

2.5.2 Evaluation of Estimation Methods

It is commonly assumed that any tracking performance achieved by a Kalman–based filter can be equally obtained (or even improved) using a PF. Of course, this only holds if the number of particles used is *big enough* to represent the distribution to be estimated. In practice, the amount of particles is commonly set in order to fulfil a given time response constraint, so in some situations, using a PF may not be the best choice. Moreover, the bigger the dimension of the state to be estimated, the bigger the number of particles required in a PF to achieve a desired performance. Considering that, we have found interesting to compare the performance of Kalman and particle based filters in different noise conditions. We have performed two types of experiments. In the first one we have compared algorithms in the estimation of local contour deformations, using a linear shape model of dimension 2. In the second the estimation of affine contour transformations has been studied, using a non–linear shape model of dimension 5.

Estimation of Local Hand Deformation

We have carried out multiple experiments to compare the performance of the KF versus a PF with 50 and 250 particles. Results obtained are shown in Figure 2.22



	6	7	8	10	13	17
MCE 1	4.03	2.13	1.43	0.91	0.56	0.39
MCE 2	1.63	1.15	0.92	0.76	0.66	0.63
MPD	2.39	0.96	0.50	0.15	−0.10	−0.24
% red.	59.45	45.24	35.15	16.41	−18.12	−60.27

	6	7	8	10	13	17
MCE 1	4.03	2.13	1.43	0.91	0.56	0.39
MCE 2	1.36	0.90	0.67	0.50	0.39	0.34
MPD	2.70	1.23	0.76	0.41	0.17	0.05
% red.	67.04	57.73	52.77	45.70	30.94	12.64

Figure 2.22: KF vs PF comparison. Left: PF with 50 particles vs best KF. Right: PF with 250 particles vs best KF.

Plots in 2.22 clearly manifest clearly the robustness of PF with respect to noisy situations. This results from its superior observation likelihood distribution, which better accounts for clutter in the observations. However, if the number of particles is too reduced, the KF can have a superior performance in low noise sequences. In general, a KF usually will be usually the choice in low–noise scenarios, as with less computation achieves a performance comparable to PFs. PFs require a big number of particles even in low noise scenarios, because otherwise the mean contour shape estimated from particles jitters around the tracked object, and an accuracy inferior to that of KF is obtained.

Estimation of Global Hand Transformation

Similarly to the previous experiments, a synthetic sequences have been generated showing a rigid contour under a time–varying affine transform. Our objective in the tracking of these sequences is two–fold: analysing the performance of Kalman–based approaches facing non–linear systems (i.e., comparing EKFs with UKFs), and comparing them against PFs.

We propose an original adaptation of the EKF and the UKF to non–linear contour tracking applications, which is significantly different to other approaches in the literature. With respect to the UKF, the few references found concerning this topic are based on a wrong modelisation of the contour measurement process. For instance, [28] uses the UKF to estimate the affine transformation of an ellipsoidal contour, but assuming that the measurement extraction process can determine the real point–to–point contour displacement, while just the normal displacement is really obtained. In [71], the use of the UKF for curve tracking is also proposed, but surprisingly in a tracking application concerning linear models. What they really propose is using the σ–points procedure of the UKF to design an alternative contour measurement process, which again does not take into account that just the normal contour displacement is observed. Once observations are obtained, the contour state could be in fact estimated using a classical KF.

In order to clarify which specific EKF and UKF implementation is proposed in this thesis[8], both are detailed in Algorithms 4 and 5. Notice that implementations solve analytically the estimation parts concerning linear models, applying the Taylor approximation and the Unscented Transform respectively, only when their application can not be eluded.

Three different sequences have been used to evaluate algorithms:

Seq1 : *slow–velocity* rigid transformations

Seq2 : *medium–velocity* rigid transformations with abrupt changes of behaviour.

Seq3 : *fast–velocity* rigid transformations with sudden accelerations.

Figure 2.23 details the affine parameters used to generate the sequences, which synthesise frames approximating the evolution observed in the outline of a hand, when this hand (or equivalently the acquisition camera) changes its location and orientation in the 3D space. Algorithms will track all these sequences using a fixed Constrained Brownian motion model (see Appendix A), which reflects poorly the real dynamics of the parameters to be estimated.

Figure 2.24 compares the performance of the EKF vs the UKF, and the UKF vs a PF with 1000 particles. Notice that with respect to previous experiments, the number of particles has been considerably increased, due to the higher dimensionality of the state to be estimated.

Plots reveal some interesting results. With respect to the Kalman–based algorithms, the EKF and the UKF have a nearly identical performance, being virtually equivalent for the contour tracking problem analysed. On the other hand, the PF clearly outperform the UKF in Seq1 (slow motion dynamics), but as long as the assumed dynamical model worse adjusts to the real dynamics (Seq2 and Seq3), its performance significantly degrades, being overcomed by the UKF in low–noise situations (for the amount of particles considered). Hence, in applications where the behaviour of targets is difficult to predict, the most common PF implementation (i.e., the Condensation algorithm) require a high number of particles to reach in low noise situations the performance of Kalman–based solutions.

[8] We claim that is the proper way to implement them, coherently with the system and observation models described in previous sections.

Algorithm 4 UKF for Contour Tracking

```
$\left[\left(\hat{\mathbf{x}}_{t|t}, \mathbf{\Sigma}^{\mathbf{xx}}_{t|t}\right)\right] = \text{UKF}\left[\left(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\right)\right]$

  Time update equations:

    $\hat{\mathbf{x}}_{t|t-1} = \mathbf{A}\mathbf{x}_{t-1|t-1} + (\mathbf{I} - \mathbf{A})\bar{\mathbf{x}}$
    $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} = \mathbf{A}\mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\mathbf{A}^T + \mathbf{Q}$

  Frame observations extraction

    $\mathbf{x}_m = \hat{\mathbf{x}}_{t|t-1}$
    $(\mathbf{y}_t, \mathbf{R}_t) = \text{FRAME\_PROCESSING}(\mathbf{x}_m)$

  Measurement update equations:

    $\left[\left\{\mathbf{x}^{(i)}_{t|t-1}, w^{(i)}_m, w^{(i)}_c\right\}^N_{i=1}\right] = \text{COMPUTE\_}\sigma\text{\_POINTS}\left[\left(\hat{\mathbf{x}}_{t|t-1}, \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\right)\right]$
    $\mathbf{y}^{(i)}_{t|t-1} = \mathbf{N}^T_m\mathbf{U}((\mathbf{W}_{(\mathbf{x}^{(i)}_{t|t-1})} - \mathbf{W}_{(\mathbf{x}_m)})\bar{\mathbf{q}} + \mathbf{T}(\mathbf{x}^{(i)}_{t|t-1} - \mathbf{x}_m))$
    $\bar{\mathbf{y}}_{t|t-1} = \sum_{i=1}^N w^{(i)}_m \mathbf{y}^{(i)}_{t|t-1}$
    $\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1} = \left(\sum_{i=1}^N w^{(i)}_c (\mathbf{y}^{(i)}_{t|t-1} - \bar{\mathbf{y}}_{t|t-1})(\mathbf{y}^{(i)}_{t|t-1} - \bar{\mathbf{y}}_{t|t-1})^T\right) + \mathbf{R}_t$
    $\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1} = \sum_{i=1}^N w^{(i)}_c (\mathbf{x}^{(i)}_{t|t-1} - \bar{\mathbf{x}}_{t|t-1})(\mathbf{y}^{(i)}_{t|t-1} - \bar{\mathbf{y}}_{t|t-1})^T$
    $\mathbf{K} = \mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}(\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})^{-1}$
    $\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}(\mathbf{y}_t - \bar{\mathbf{y}}_{t|t-1})$
    $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t} = \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} - \mathbf{K}\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1}\mathbf{K}^T$
```

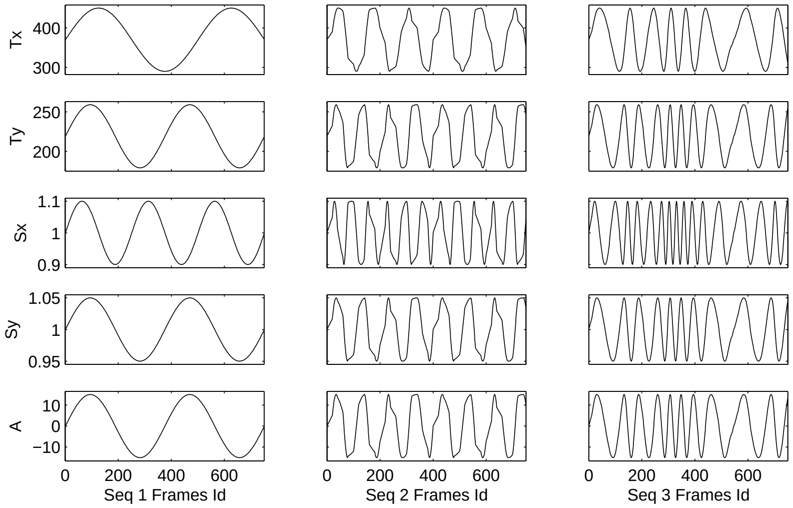


Figure 2.23: Ground truth sequence parameters. $\mathbf{x}^R$ parameters of the sequences used in the experiments.

Algorithm 5 EKF for Contour Tracking

$$\left[\left(\hat{\mathbf{x}}_{t|t}, \boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t}\right)\right] = \mathrm{EKF}\left[\left(\hat{\mathbf{x}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\right)\right]$$

Time update equations:

$$\begin{aligned}\hat{\mathbf{x}}_{t|t-1} &= \mathbf{A}\mathbf{x}_{t-1|t-1} + (\mathbf{I} - \mathbf{A})\bar{\mathbf{x}} \\ \boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1} &= \mathbf{A}\boldsymbol{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\mathbf{A}^T + \mathbf{Q}\end{aligned}$$

Frame observations extraction

$$\begin{aligned}\mathbf{x}_m &= \hat{\mathbf{x}}_{t|t-1} \\ (\mathbf{y}_t, \mathbf{R}_t) &= \text{FRAME\_PROCESSING}(\mathbf{x}_m)\end{aligned}$$

Shape Space Linearization

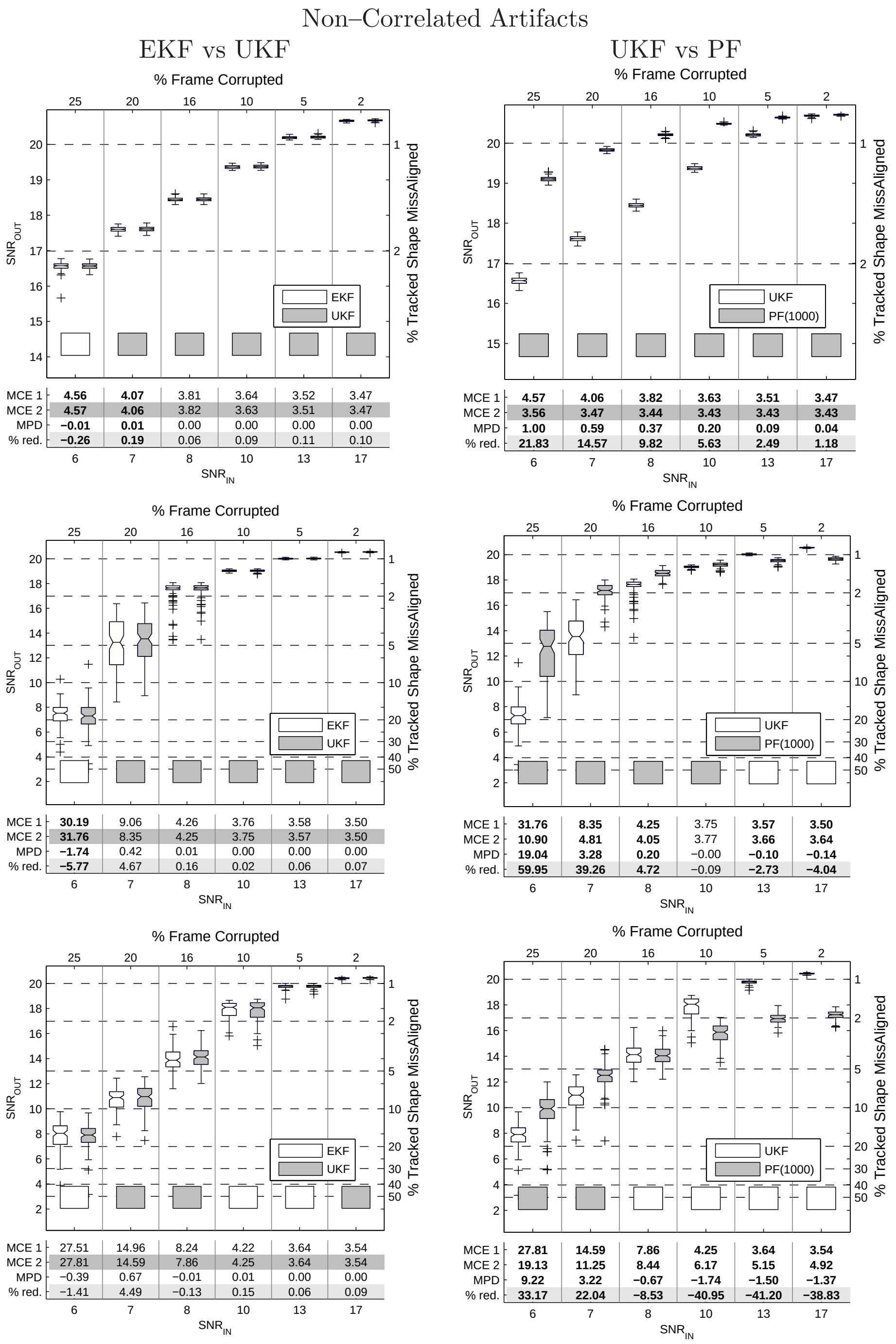
$$\mathbf{W}_{\mathbf{lx}} = \begin{bmatrix} \mathbf{1}_{ncp} & \mathbf{0}_{ncp} \\ \mathbf{0}_{ncp} & \mathbf{1}_{ncp} \\ (\cos(\theta)\bar{\mathbf{q}}^x - \sin(\theta)\bar{\mathbf{q}}^y) & \mathbf{0}_{ncp} \\ \mathbf{0}_{ncp} & (\sin(\theta)\bar{\mathbf{q}}^x + \cos(\theta)\bar{\mathbf{q}}^y) \\ -s_x(\sin(\theta)\bar{\mathbf{q}}^x + \cos(\theta)\bar{\mathbf{q}}^y) & s_y(\cos(\theta)\bar{\mathbf{q}}^x - \sin(\theta)\bar{\mathbf{q}}y) \end{bmatrix}^T$$

where $[s_x, s_y, \theta]$ are taken from $\hat{\mathbf{x}}_{t|t-1}$.

$$\mathbf{H}_{\mathbf{lx}} = \mathbf{N}_m^T\mathbf{U}\mathbf{W}_{\mathbf{lx}}$$

Measurement update equations:

$$\begin{aligned}\mathbf{K} &= \frac{\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}_{\mathbf{lx}}{}^T}{\mathbf{H}_{\mathbf{lx}}\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}_{\mathbf{lx}}{}^T + \mathbf{R}_t} \\ \nu_t &= \mathbf{y}_t - (\mathbf{N}_m^T\mathbf{U}((\mathbf{W}_{(\hat{\mathbf{x}}_{t|t-1})} - \mathbf{W}_{(\mathbf{x}_m)})\bar{\mathbf{q}} + \mathbf{T}(\hat{\mathbf{x}}_{t|t-1} - \mathbf{x}_m))) \\ \hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}\nu_t \\ \boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t} &= (\mathbf{I} - \mathbf{K}\mathbf{H}_{\mathbf{lx}})\boldsymbol{\Sigma}^{\mathbf{xx}}_{t|t-1}\end{aligned}$$



Seq1, EKF vs UKF

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	**4.56**	**4.07**	3.81	3.64	3.52	3.47
MCE 2	**4.57**	**4.06**	3.82	3.63	3.51	3.47
MPD	**−0.01**	**0.01**	0.00	0.00	0.00	0.00
% red.	**−0.26**	**0.19**	0.06	0.09	0.11	0.10

Seq1, UKF vs PF

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	**4.57**	**4.06**	**3.82**	**3.63**	**3.51**	**3.47**
MCE 2	**3.56**	**3.47**	**3.44**	**3.43**	**3.43**	**3.43**
MPD	**1.00**	**0.59**	**0.37**	**0.20**	**0.09**	**0.04**
% red.	**21.83**	**14.57**	**9.82**	**5.63**	**2.49**	**1.18**

Seq2, EKF vs UKF

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	**30.19**	9.06	4.26	3.76	3.58	3.50
MCE 2	**31.76**	8.35	4.25	3.75	3.57	3.50
MPD	**−1.74**	0.42	0.01	0.00	0.00	0.00
% red.	**−5.77**	4.67	0.16	0.02	0.06	0.07

Seq2, UKF vs PF

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	**31.76**	**8.35**	**4.25**	3.75	**3.57**	**3.50**
MCE 2	**10.90**	**4.81**	**4.05**	3.77	**3.66**	**3.64**
MPD	**19.04**	**3.28**	**0.20**	−0.00	**−0.10**	**−0.14**
% red.	**59.95**	**39.26**	**4.72**	−0.09	**−2.73**	**−4.04**

Seq3, EKF vs UKF

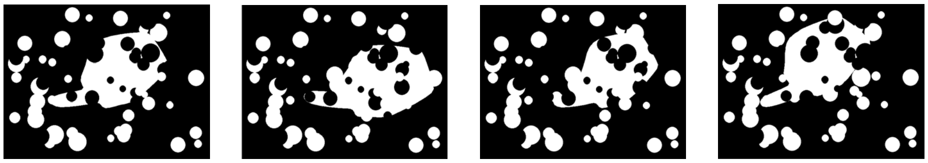
SNR$_{IN}$	6	7	8	10	13	17
MCE 1	27.51	14.96	8.24	4.22	3.64	3.54
MCE 2	27.81	14.59	7.86	4.25	3.64	3.54
MPD	−0.39	0.67	−0.01	0.01	0.00	0.00
% red.	−1.41	4.49	−0.13	0.15	0.06	0.09

Seq3, UKF vs PF

SNR$_{IN}$	6	7	8	10	13	17
MCE 1	**27.81**	**14.59**	**7.86**	**4.25**	**3.64**	**3.54**
MCE 2	**19.13**	**11.25**	**8.44**	**6.17**	**5.15**	**4.92**
MPD	**9.22**	**3.22**	**−0.67**	**−1.74**	**−1.50**	**−1.37**
% red.	**33.17**	**22.04**	**−8.53**	**−40.95**	**−41.20**	**−38.83**

Figure 2.24: EKF vs UKF vs PF comparison. Performance obtained for the three evaluation sequences considered (from top to bottom, Seq1, Seq2, and Seq3).

Experiments have been repeated to study the performance of algorithms in sequences disturbed by elements correlated along time. The objective is to simulate very challenging situations that may occur in real applications: the presence of static elements in the scene that continuously occlude the tracked object, and the presence of elements with characteristics similar to the target, that add consistently clutter along time. Figure 2.25 shows some frames of a sequence with this kind of artifacts.



frame 1 frame 10 frame 20 frame 30

Figure 2.25: Frames of a sequence disturbed by correlated noise.

In the experiments, only frame distortions up to just 10% of their pixels are considered, while previous experiments considered distortions up to the 25%. As correlated artifacts disturbs more severely the tracking performance, extremely high noise situations are unreasonable to be considered, as in just a few frames algorithms will loose the target contour. Figure 2.26 shows the tracking performance achieved by EKFs, UKFs, and PFs. EKFs and UKFs performs very similarly in this sequences, showing a very weak performance with respect to correlated artifacts. If the target movement is not slow, they easily get distracted from clutter and misstrack the contour. However, PFs display a very robust performance even for this kind of artifacts, showing a tracking accuracy scarcely inferior to the one achieved with non–correlated artifacts.

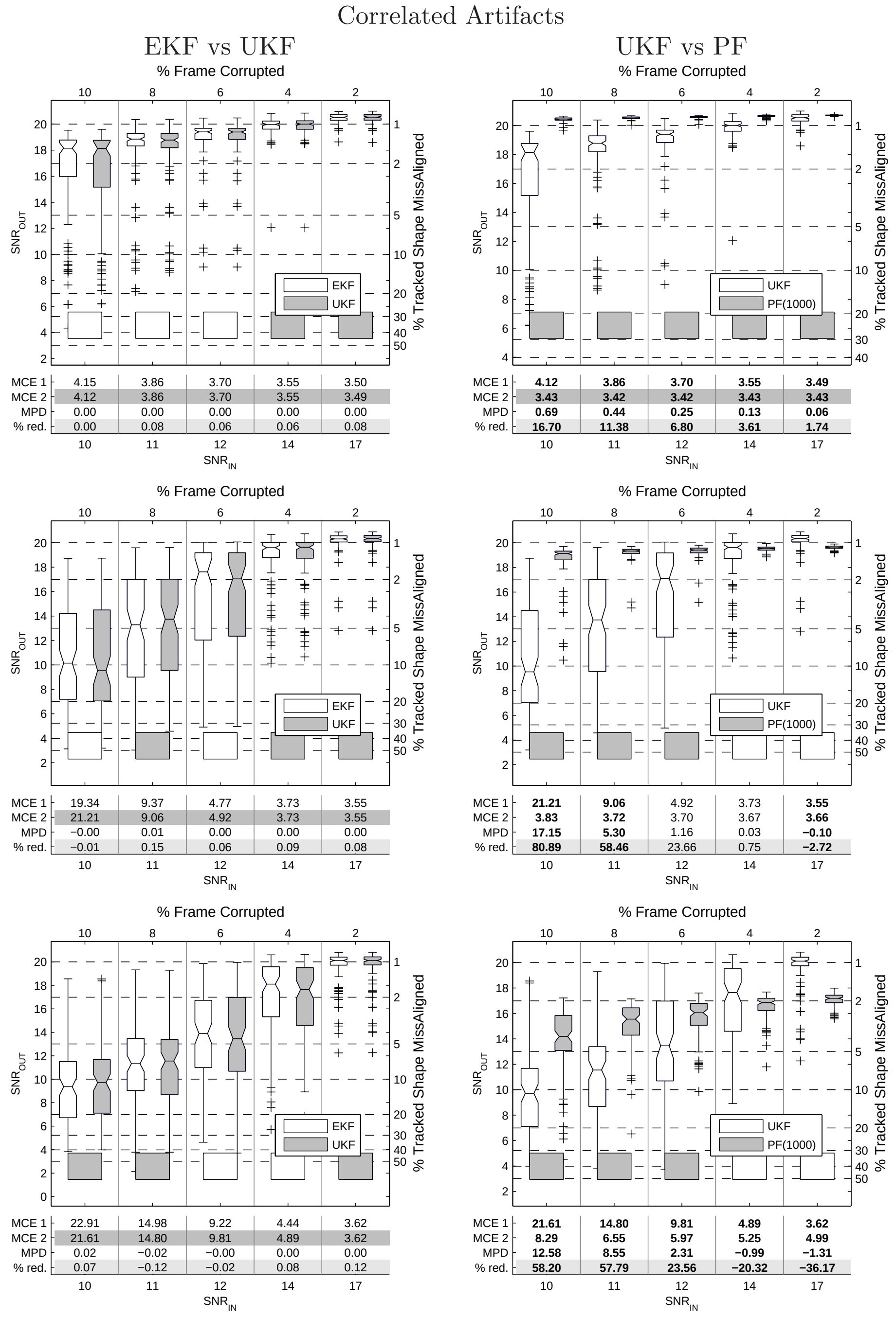
With these artifacts it is observed that, for high noise situations, Kalman–based filters usually generate estimations which corresponds to invalid parameterisations of the shape model, which explain the very poor performance in high noise scenarios. PFs does not suffer from this problem, as long as the importance function used generates just valid shape parameterisations [9].

Summary

Results obtained manifest the well–known superior performance of PFs with respect to Kalman–based approaches, provided a big enough number of particles is considered. A relevant point to remark is their similar behaviour for correlated and non–correlated artifacts, a behaviour not observed in Kalman–based estimators, which easily misstrack the target in the presence of correlated noise. On the other hand, in low noise scenarios, the use of a poor dynamical model affects more severely PFs than KFs, demanding a higher number of particles.

Concerning the comparison between EKFs and UKFs, results mainly show their equivalence in the analysed application. The UKF is slightly better in a major number

[9] In the experiments done, this is guaranteed by the constrained Brownian motion model used to model dynamics.



	10	11	12	14	17
MCE 1	4.15	3.86	3.70	3.55	3.50
MCE 2	4.12	3.86	3.70	3.55	3.49
MPD	0.00	0.00	0.00	0.00	0.00
% red.	0.00	0.08	0.06	0.06	0.08

	10	11	12	14	17
MCE 1	**4.12**	**3.86**	**3.70**	**3.55**	**3.49**
MCE 2	**3.43**	**3.42**	**3.42**	**3.43**	**3.43**
MPD	**0.69**	**0.44**	**0.25**	**0.13**	**0.06**
% red.	**16.70**	**11.38**	**6.80**	**3.61**	**1.74**

	10	11	12	14	17
MCE 1	19.34	9.37	4.77	3.73	3.55
MCE 2	21.21	9.06	4.92	3.73	3.55
MPD	−0.00	0.01	0.00	0.00	0.00
% red.	−0.01	0.15	0.06	0.09	0.08

	10	11	12	14	17
MCE 1	**21.21**	**9.06**	4.92	3.73	**3.55**
MCE 2	**3.83**	**3.72**	3.70	3.67	**3.66**
MPD	**17.15**	**5.30**	1.16	0.03	**−0.10**
% red.	**80.89**	**58.46**	23.66	0.75	**−2.72**

	10	11	12	14	17
MCE 1	22.91	14.98	9.22	4.44	3.62
MCE 2	21.61	14.80	9.81	4.89	3.62
MPD	0.02	−0.02	−0.00	0.00	0.00
% red.	0.07	−0.12	−0.02	0.08	0.12

	10	11	12	14	17
MCE 1	**21.61**	**14.80**	**9.81**	**4.89**	**3.62**
MCE 2	**8.29**	**6.55**	**5.97**	**5.25**	**4.99**
MPD	**12.58**	**8.55**	**2.31**	**−0.99**	**−1.31**
% red.	**58.20**	**57.79**	**23.56**	**−20.32**	**−36.17**

Figure 2.26: EKF vs UKF vs PF comparison under correlated noise. Performance achieved for the three evaluation sequences with correlated noise considered (from top to bottom, Seq1, Seq2, and Seq3).

of noisy conditions considered, although the difference of performance observed is not statistically significant in most cases.

2.6 Conclusions

In this chapter we have introduced the Active Contour approach to contour tracking, describing in detail the most relevant issues of their implementation: the modelisation of shape and dynamics, the measurement of contour in images, and the estimation methods applied to interpret correctly these measurements (Kalman–based and Particle–based filters). After this introduction, the chapter contributes with the results of experimental work done, devoted to discerning the best way to track contours in sequences.

Concerning KFs, our experiments conclude that to obtain the best tracking performance, observations have to be extracted from measurement lines normal to the predicted tracked contour, delimiting their length from the uncertainty on this prediction.

Concerning PFs, there is no advantage on making and adaptive control of the length of measurement lines. Determining the orientation of measurement lines according to the most likely direction of deformation of the contour improves the performance achieved in noisy situations. However, in practice the cost of determining the direction of measurement lines for each particle contour is noteworthy, and it is computationally cheaper to improve tracking performance by just increasing the amount of particles considered, using normal measurement lines. Our experiments also corroborate the superior performance of PFs with respect to KFs, provided the amount of particles considered is sufficiently big.

The chapter also evaluates the performance of estimation methods dealing with non–linear problems. We have proposed an original implementation of the EKF and UKF for contour tracking applications, which differs from other proposals in the literature in the more precise modelisation of the information provided by the observations. Results show the practical equivalence of the EKF and the UKF in the studied problem, and the superiority of PF to these methods, as long as the number of particles used is sufficiently large. It is observed that to overcome the performance of UKFs, the amount of particles required by the standard PF implementation increases notably if the contour dynamical model is poor.

Finally, we also evaluate the performance of the algorithms in sequences distorted by correlated noise. In these cases, the degradation of the performance of EKFs and UKFs is very pronounced, while the performance of PFs scarcely varies with respect to the one observed in sequences with uncorrelated noise. Thus, in applications where the target to be tracked present mostly persistent occlusions, PFs are clearly the algorithms to be applied.

Chapter 3

Contour Estimation and the Curse of Dimensionality

The preceding chapter describes the basic components of model–based sequential contour estimation, and evaluates the performance of different tracking proposals in synthetic sequences. In the experimental work done, local and global contour transformations (modelled respectively by linear and non–linear expressions) have been estimated separately. However, in real applications the outline of objects usually displays simultaneously local and global transformations (commonly affine or Euclidean similarities). This chapter focuses on the joint estimation of both contour transformations, proposing the application of novel methods to solve this problem in order to improve the performance achieved by standard PFs.

The simultaneous estimation of local and global contour transformations is easy to pose, as both transformations can be modelled, and consequently an estimator for them can be designed. The state vector of the shape to be tracked is defined as

$$\mathbf{x} = [\mathbf{x}^D \ \mathbf{x}^R]^T$$

where $\mathbf{x}^D$ and $\mathbf{x}^R$ denote respectively parameters of local deformation, and global rigid transformation. In the following, it is considered that $\mathbf{x}^D$ represents shape space parameters from $\mathcal{L}(\mathbf{W}^D, \bar{\mathbf{q}})$, and $\mathbf{x}^R$ affine transformation parameters given by

$$\mathbf{x}^R = [t_x \ t_y \ s_x \ s_y \ \theta]^T \ . \tag{3.1}$$

To synthesise the B-spline control points $\mathbf{q}$ of the 2D shape that $\mathbf{x}$ represents, both state vector parts are properly combined by

$$\mathbf{q} \quad = \quad \mathbf{W}^R{}_{(\mathbf{x}^R)} \left(\mathbf{W}^D\mathbf{x}^D + \bar{\mathbf{q}}\right) + \mathbf{T}\mathbf{x}^R \ , \tag{3.2}$$

where $\mathbf{W}^R{}_{()}$ and $\mathbf{T}$ have the form presented in Section 2.1.2. From this shape model, and using the same formalisation presented in the previous chapter, the contour tracking problem is described by the following dynamical and observation models

$$\begin{aligned}
\mathbf{x}_t - \bar{\mathbf{x}} &= \mathbf{A}(\mathbf{x}_{t-1} - \bar{\mathbf{x}}) + \mathbf{B}\mathbf{w}_t \ , \\
\mathbf{y}_t &= \mathbf{N}_m^t \mathbf{U}\left(\left(\mathbf{W}^R_{(\mathbf{x}^R)}\left(\mathbf{W}^D\mathbf{x}^D + \bar{\mathbf{q}}\right) + \mathbf{T}\mathbf{x}^R\right) - \right. \\
&\quad \left. - \left(\mathbf{W}^R_{(\mathbf{x}^R_m)}\left(\mathbf{W}^D\mathbf{x}^D_m + \bar{\mathbf{q}}\right) + \mathbf{T}\mathbf{x}^R_m\right)\right) + \mathbf{v}_t \ .
\end{aligned}$$

Commonly $\mathbf{A}$ is a block diagonal matrix, as $\mathbf{x}^R$ and $\mathbf{x}^D$ are uncorrelated in many practical problems. The noise term $\mathbf{v}_t$ disturbing $\mathbf{y}_t$ is assumed Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ for Kalman–based solutions, and arbitraryly distributed in PF approaches. As the relation between state $\mathbf{x}_t$ and observations $\mathbf{y}_t$ is non–linear, schemes like the EKF, UKF and PF can be used to solve the required contour tracking application. As remarked in previous experiments, the most robust and reliable performance will be obtained by means of PFs, as long as the number of particles used is appropriate. We have reproduced the experimental work presented in the previous chapter, this time evaluating algorithms in synthetic sequences showing simultaneous rigid and non–rigid contour transformations[1]. The tracking performance achieved by each method is shown in Figure 3.1.

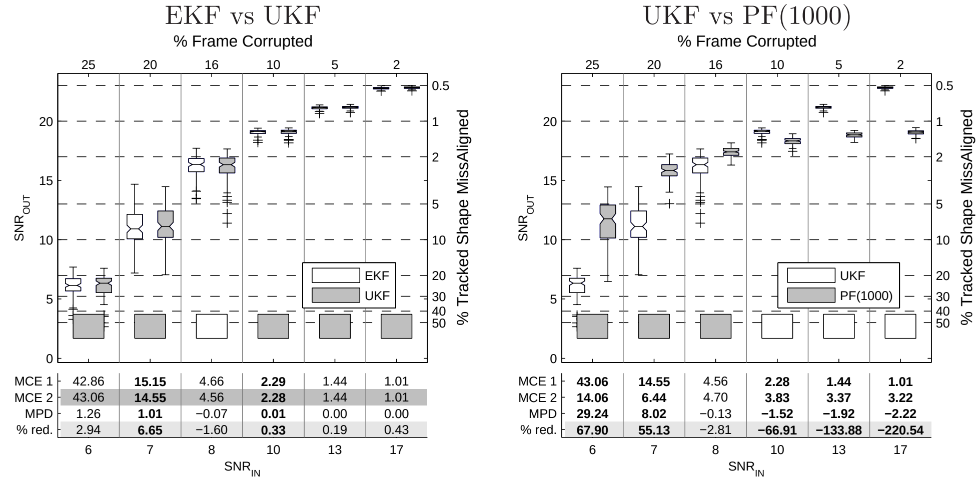


Figure 3.1: EKF vs UKF vs PF comparison. Left: EKF vs UKF. Right: UKF vs PF with 1000 particles.

As concluded in the previous chapter, the performance achieved by the EKF and the UKF is practically identical. Again, the PF is the method more robust in noisy situations, but now it is observed that in low noise situations it is significantly overcomed by Kalman–based filters. This is due to the higher dimensionality of the state now estimated, which requires to use a higher number of particles in the PF. Figure 3.2 shows the performance of a PF with twice and four times as much particles.

[1] We have generated sequences where the rigid transformations correspond to the ones of **Seq2** in the previous chapter, displaying at the same time local contour deformations.

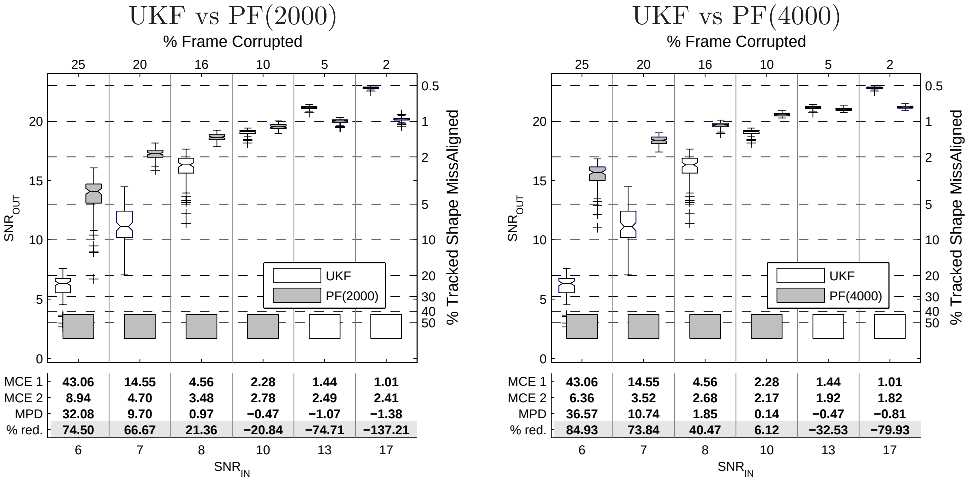


Figure 3.2: UKF vs PF comparison, using 2000 (left) and 4000 (right) particles.

Notice that although the number of particles is increased significantly, the accuracy achieved in low noise situations is still below a Kalman–based solution performance. This makes apparent the *curse of dimensionality* (previously introduced in Section 2.3.2) that suffer importance sampling based algorithms. This weakness of PF–solutions has motivated the proposal of novel schemes to diminish the effect of the space dimensionality in the number of required particles. Concerning the contour tracking problem, we propose in this chapter methods to deal with this problem from three different points of view:

- by using a *better* importance function in the SISR algorithm, which takes the most current observations into account,
- by solving part of the estimation problem analytically,
- by taking advantage that in most contour tracking applications, the contour rigid transformations can be estimated decoupled from the non–rigid ones.

Our first proposal is detailed on Section 3.1, and it is based on applying the unscented Kalman particle filter (UKPF) [115] in the contour tracking problem. The basic idea is at each time step, estimate for each particle a Gaussian approximation of the optimal importance function $p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})$. Using this importance function, samples are generated around image regions where there is some evidence of the presence of the tracked contour, and that makes possible tracking the shape of interest with fewer particles.

Our second proposal is based on applying a technique called *Rao-Blackwellisation.* This methodology proposes to estimate part of the state–vector $\mathbf{x}$ analytically, and the remaining part by means of a PF. Since this remaining part is obviously of lower dimension, the amount of particles required by the PF should be reduced. Section 3.2 details how we have adapted this technique to the studied problem.

The last technique that we propose in this chapter exploits the fact that more particles are required for a PF in a space of dimension N, than for instance by m PFs in spaces of dimension N/m. Hence, given an state $\mathbf{x}$, if it were feasible to decomposed it into decoupled parts, it could be estimated more efficiently. For the contour tracking problem analysed, this state decomposition is not in general feasible. However, we show in Section 3.3 that making some assumptions that hold in many practical cases, it is possible at least to estimate independently the global contour transformations $\mathbf{x}^R$ from the local ones $\mathbf{x}^D$. We take advantage on that using the the partitioned sampling technique (Section 3.3.2), achieving a more accurate contour tracking performance .

3.1 Contour Tracking Using an Unscented Kalman Particle Filter

To understand this contour tracking proposal, first it may be helpful for the reader reviewing briefly the essence of PFs (Section 2.3.2 in the preceding chapter). In short, the aim of PFs is characterising the distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ by means of a set of N samples from it. As this distribution is unknown and can not be sampled, what is done in practice is generating samples $\{\mathbf{x}_{0:t}^{(i)}\}_{i=1}^{N}$ from an importance function $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, and weight them according to a factor $\tilde{w}_t^{(i)}$, that evaluates if each $\mathbf{x}_{0:t}^{(i)}$ can be considered a real sample of $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Thus, $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ is approximated by

$$\widetilde{p_N}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \quad = \quad \sum_{i=1}^{N} \delta_{\mathbf{x}_{0:t}^{(i)}} \tilde{w}_t^{(i)} \quad ,$$

where $\tilde{w}_t^{(i)}$ is the normalised version of the factor

$$w_t^{(i)} \quad = \quad \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})} \quad ,$$

so that $\sum_{i=1}^{N} \tilde{w}_t^{(i)} = 1$.

PFs are SISR algorithms that use importance functions of the form

$$q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \quad = \quad q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}) \quad ,$$

so that the importance weight w_t can be computed (up to a normalising factor) by the recursive expression

$$w_t \quad \propto \quad \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}\mathbf{y}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{0:t}\mathbf{y}_{1:t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})} w_{t-1} \quad . \tag{3.3}$$

If the assumed state evolution along time corresponds to a Markov process, and observations are conditionally independent given the states, this expression reduces to

$$w_t \quad \propto \quad \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})} w_{t-1} \quad . \tag{3.4}$$

The key point of a SISR algorithm is the importance function $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})$ used to generate particles at each time step. In [41] it is proved that the optimal[2] function to carry this task is $p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t})$, whose determination in practical applications is commonly a non–trivial task. The Unscented Kalman Particle Filter (UKPF) [115] proposes approximating the optimal importance function locally around each particle $\mathbf{x}_{0:t-1}^{(i)}$ by means of a Gaussian distribution. In concrete, for each particle $\mathbf{x}_{0:t-1}^{(i)}$, an UKF is used to generate and propagate a Gaussian importance function

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t}) = \mathcal{N}(\hat{\mathbf{x}}_t^{(i)}, \mathbf{\Sigma}_t^{\mathbf{xx}^{(i)}}) \ .$$

In practice this means that for each particle $(\mathbf{x}_{0:t}^{(i)}, w_{1:t}^{(i)})$, now it is also required to maintain matrices $\mathbf{\Sigma}_{0:t}^{\mathbf{xx}^{(i)}}$ detailing the covariance of the associated Gaussian importance function at each time step. We propose to use this strategy in the described contour tracking problem, which leads to the procedure detailed in Algorithm 6.

Algorithm 6 Sequential Importance Sampling Resampling Iteration

```
$[\{\mathbf{x}_{0:t}^{(i)}, \mathbf{\Sigma}_{0:t}^{\mathbf{xx}(i)}, w_t^{(i)}\}_{i=1}^N] = \text{UKPF}[\{\mathbf{x}_{0:t-1}^{(i)}, \mathbf{\Sigma}_{0:t-1}^{\mathbf{xx}(i)}, w_{t-1}^{(i)}\}_{i=1}^N, \mathbf{y}_{1:t}]$

  for $i = 1$ to $N$ do
    $\left(\hat{\mathbf{x}}_t^{(i)}, \mathbf{\Sigma}_t^{\mathbf{xx}(i)}\right) = \text{UKF}\left[\left(\mathbf{x}_{t-1}^{(i)}, \mathbf{\Sigma}_{t-1}^{\mathbf{xx}(i)}\right)\right]$
    Set $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t}) = \mathcal{N}(\hat{\mathbf{x}}_t^{(i)}, \mathbf{\Sigma}_t^{\mathbf{xx}^{(i)}})$
    Draw $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t})$
    Set $\mathbf{x}_{0:t}^{(i)} \triangleq \left[\mathbf{x}_{0:t-1}^{(i)} \ \mathbf{x}_t^{(i)}\right]$ and $\mathbf{\Sigma}_{0:t}^{\mathbf{xx}(i)} \triangleq \left[\mathbf{\Sigma}_{0:t-1}^{\mathbf{xx}(i)} \ \mathbf{\Sigma}_t^{\mathbf{xx}(i)}\right]$
    $(\mathbf{y}_t, \mathbf{R}_t) = \text{FRAME\_PROCESSING}(\mathbf{x}_t^{(}i))$
    Update weight $w_t^{(i)} = \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}\mathbf{y}_{1:t})} w_{t-1}^{(i)}$
  end for
  Normalise weights $\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}} \quad \forall i = 1, \ldots, N$
  Compute $N_{eff} = \frac{1}{\sum_{i=1}^N \tilde{w}_t^{(i)}}$
  if $N_{eff} < N_{deg}$ then
    $[\{\mathbf{x}_{0:t}^{(i)}, \mathbf{\Sigma}_{0:t}^{\mathbf{xx}(i)}, w_t^{(i)}\}_{i=1}^N] = \text{RESAMPLE}[\{\mathbf{x}_{0:t}^{(i)}, \mathbf{\Sigma}_{0:t}^{\mathbf{xx}(i)}, \tilde{w}_t^{(i)}\}_{i=1}^N]$
  end if
```

Following the pseudocode in Algorithm 6, the execution of a UKF is required for each sample, resulting in a high computational cost. However, in practice one can take advantage of a remarkable fact of PFs applied to contour tracking applications. At each iteration, the amount of particles surviving the resampling step is commonly small (in most cases around the 25% of the total of particles). Due to this fact, at the time of propagating particles most of them are identical, and all of them can share the same linearisation of the optimal sampling function. Thus, at each iteration it have to be evaluated just as many UKFs as different surviving particles, reducing a great deal the computational cost of the algorithm.

[2] In terms of minimising the variance of the particle weights w_t.

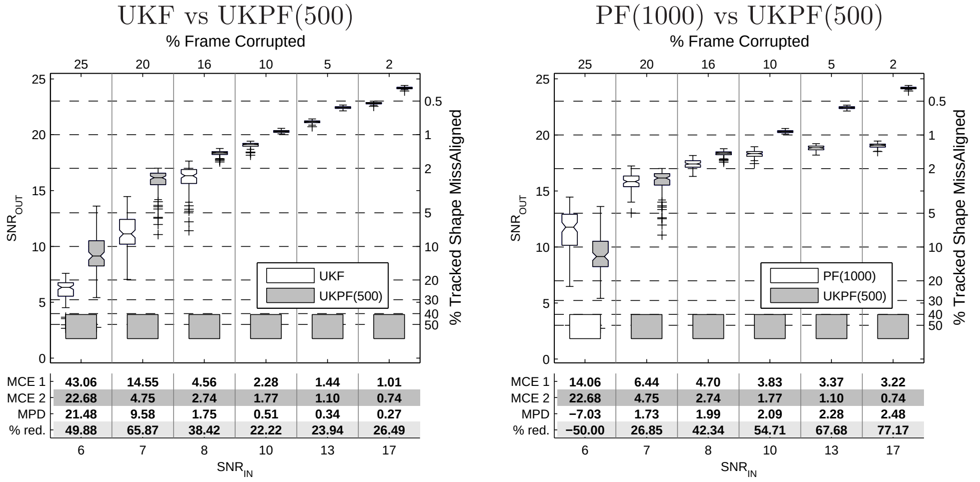
The application of the UKPF in visual contour tracking applications has been also simultaneously proposed by other authors. However, their approaches differ significantly from ours. In [99] a face tracker is presented, based on modelling the face outline by means of an ellipse of fixed size and orientation. The system only accounts for translations of the contour model. In [72] a contour tracker is proposed, aimed at the estimation of affine transformations of a rigid contour model. These two approaches apply unnecessary the UKPF on their tracking problem, because as long as they use linear system and observation models, a classical KF could be used to approximate the optimal importance function for each particle. In addition to the unnecessary use of the UKF, both approaches derive from a wrong modelisation of the contour measurement process. Both assume that from the observation process the disparity *coordinate–by–coordinate* between predicted and observed contours can be measured, while in practice only the normal disparity with respect to a defined measurement contour is really obtained. Our proposal differs from these ones in the fact that the use of the UKPF is completely necessary (non–linear expressions are involved), and that a right modelling of the observation process is done, interpreting more accurately the evidence extracted from the processed frames.

3.1.1 Experimental Evaluation

Before comparing the performance of the UKPF versus the PF, first we need to define an criterion to establish the amount of particles used in each of them, in order to make a fair comparison. The UKPF currently presented, as well as the two other methods detailed on subsequent sections, introduce modifications to the classical PF implementation, which require additional computational resources. From this fact, one may think of establishing the parameters of the compared algorithms (in this case, the number of particles considered) to fulfil a fixed computational cost. However, this criterion is of poor practical use, as an algorithm non–optimally implemented may be in clear disadvantage with respect to a *worse* algorithm better engineered. Moreover, the computational cost of an algorithm can vary in practice depending on the compiler used, and the characteristics of the machine available. To avoid the dependence on these factors, we propose to compare algorithms, by parameterising them in order to balance their capacity of extracting evidence from images. Hence, we set the number of particle of each algorithm, in such a way that all extract the same amount of observations per frame [3]. In that way, what our performance evaluation methodology reflects is the ability of each strategy in taking advantage of the measurement process. Following this criterion, given that a PF with N particles performs N image measurements to evaluate their likelihood, this algorithm will be compared against an UKPF with $N/2$ particles, since this one performs $N/2$ image measurements to estimate the linearised optimal importance function, and $N/2$ more to evaluate the particles likelihood. Using this criterion, Figure 3.3 compares the UKF and the PF with the UKPF.

The UKPF overcomes the UKF in all the tested situations. With respect to a classical PF, it performs significantly better in all cases, except in the case of

[3] Of course, Kalman–based algorithms are an exception to this criterion, since execute just a single contour measurement process per frame.



SNR_IN	6	7	8	10	13	17
MCE 1	43.06	14.55	4.56	2.28	1.44	1.01
MCE 2	22.68	4.75	2.74	1.77	1.10	0.74
MPD	21.48	9.58	1.75	0.51	0.34	0.27
% red.	49.88	65.87	38.42	22.22	23.94	26.49

SNR_IN	6	7	8	10	13	17
MCE 1	14.06	6.44	4.70	3.83	3.37	3.22
MCE 2	22.68	4.75	2.74	1.77	1.10	0.74
MPD	−7.03	1.73	1.99	2.09	2.28	2.48
% red.	−50.00	26.85	42.34	54.71	67.68	77.17

Figure 3.3: UKF vs PF vs UKPF comparison. Left: UKF vs UKPF with 500 particles. Right: PF with 1000 particles vs UKPF with 500 particles.

sequences with SNR=6db. In these cases, due to the non–Gaussianity of the noise artifacts, the linear approximation of the optimal sampling function is very unreliable, and results show that it is better to generate samples from the a priori assumed model of dynamics.

3.2 Rao–Blackwellized Contour Estimation

The strategy that we proposed on this section is based on factorising the desired density $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. If the state to be estimated is divided into two parts $\mathbf{x}_{0:t} = [\mathbf{x}_{0:t}^{P1}\mathbf{x}_{0:t}^{P2}]^T$, then this density corresponds to $p(\mathbf{x}_{0:t}^{P1}\mathbf{x}_{0:t}^{P2}|\mathbf{y}_{1:t})$, which can be factored as

$$p(\mathbf{x}_{0:t}^{P1}\mathbf{x}_{0:t}^{P2}|\mathbf{y}_{1:t}) \quad = \quad p(\mathbf{x}_{0:t}^{P1}|\mathbf{y}_{1:t})p(\mathbf{x}_{0:t}^{P2}|\mathbf{x}_{0:t}^{P1}\mathbf{y}_{1:t}) \ . \tag{3.5}$$

The Rao–Blackwell technique [40, 41] proposes to use this structural information to infer analytically a part of the state ($\mathbf{x}_{0:t}^{P2}$) conditionally upon the other part of the state ($\mathbf{x}_{0:t}^{P1}$), which is estimated respectively by a Sequential Monte Carlo algorithm. In the concrete case of Equation (3.5), this means solving:

- $p(\mathbf{x}_{0:t}^{P2}|\mathbf{x}_{0:t}^{P1}\mathbf{y}_{1:t})$ analytically;
- $p(\mathbf{x}_{0:t}^{P1}|\mathbf{y}_{1:t})$ by means of a PF.

From that, an estimation of the posterior density $p(\mathbf{x}_{0:t}^{P1}\mathbf{x}_{0:t}^{P2}|\mathbf{y}_{1:t})$ is obtained, given by the following mixture of densities

$$p_N(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} p(\mathbf{x}_{0:t}^{P2}|\mathbf{x}_{0:t}^{P1(i)}\mathbf{y}_{1:t}) \ . \tag{3.6}$$

The Rao–Blackwell approach states that if this analytic solution to $p(\mathbf{x}_{0:t}^{P2}|\mathbf{x}_{0:t}^{P1}\mathbf{y}_{1:t})$ represents precisely the inherent real distribution, then the proposed methodology directly leads to overcome the accuracy of a *classical* PF approach. This results from the fact that using this technique, the variance of the weight of particles is reduced, maintaining a bigger particle diversity after resampling [40]. Some examples of the application of Rao–Blackwellised Particle Filters (RBPFs) in different areas are: neural networks [5], map learning [87], positioning and navigation [52] and jump Markov linear systems [42]. In this thesis we contribute with the novel adaption of this technique to the problem of contour tracking, quantifying its potential for improving the performance of PFs in this application field.

Behind the Rao–Blackwell proposal lies a rule of thumb in estimation theory: if something can be solved analytically, do not solve it by means of sampling techniques. From the methods reviewed in the previous chapter, the ones suitable to provide an analytic solution to $p(\mathbf{x}_{0:t}^{P2}|\mathbf{x}_{0:t}^{P1}\mathbf{y}_{1:t})$ are the Kalman–based filters. Due to that, the idea behind Rao–Blackwell PFs has also been published from the perspective of managing a mixture of KFs [27]. The use of KFs inherently means modelling the estimation of $\mathbf{x}^{P2}$ given $\mathbf{x}^{P1}$ by means of linear Gaussian processes. In the problem of visual contour tracking, estimating part of the state analytically from $\mathbf{y}_{1:t}$ is in general bad posed, as non–Gaussian artifacts can distort observations. Thus, the benefits of applying the Rao–Blackwell technique onto this application can not be taken for granted a priori. To ascertain if there are more advantages than disadvantages in using this technique, we have evaluated its performance experimentally, providing quantitative insight onto this point (see Section 3.2.1). Let's first detail the Rao–Blackwell estimation of $p(\mathbf{x}_{0:t}^{P1}\mathbf{x}_{0:t}^{P2}|\mathbf{y}_{1:t})$. It requires updating at each time t the distribution in Equation (3.6), which implies:

- updating the sample distribution of $p(\mathbf{x}_{0:t}^{P1}|\mathbf{y}_{1:t})$ given by a weighted sample set $\{(\mathbf{x}_{0:t}^{P1(i)}, \tilde{w}_t^{(i)})\}_{i=1}^N$;
- updating for each i-th sample $\mathbf{x}_{0:t}^{P1(i)}$, the distribution $p(\mathbf{x}_{0:t}^{P2}|\mathbf{x}_{0:t}^{P1(i)}\mathbf{y}_{1:t})$. This is directly solved using a KF, which estimates the parameters of the normal density $\mathcal{N}(\hat{\mathbf{x}}_{0:t}^{P2(i)}, \mathbf{\Sigma}_{0:t}^{\mathbf{x}^{P2}\mathbf{x}^{P2}})$.

The estimation of $p(\mathbf{x}_{0:t}^{P1}|\mathbf{y}_{1:t})$ is solved using the Sequential Importance Sampling technique reviewed in the previous section. Now, samples $\mathbf{x}_{0:t}^{P1(i)}$ are generated from an importance function $q(\mathbf{x}_{0:t}^{P1}|\mathbf{y}_{1:t})$ following the form

$$q(\mathbf{x}_{0:t}^{P1}|\mathbf{y}_{1:t}) \quad = \quad q(\mathbf{x}_0^{P1})\prod_{k=1}^{t} q(\mathbf{x}_k^{P1}|\mathbf{x}_{0:k-1}^{P1}\mathbf{y}_{1:k}) \ .$$

Using this type of importance function, the (unnormalised) weight factor associated to each particle $\mathbf{x}_{0:t}^{P1(i)}$ is determined recursively as

$$w_t^{(i)} \quad \propto \quad \frac{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}\mathbf{x}_{0:t}^{P1(i)})p(\mathbf{x}_t^{P1(i)}|\mathbf{x}_{t-1}^{P1(i)})}{q(\mathbf{x}_t^{P1(i)}|\mathbf{y}_{1:t}\mathbf{x}_{0:t-1}^{P1(i)})} w_{t-1}^{(i)} \ .$$

If the prior distribution $p(\mathbf{x}_t^{P1}|\mathbf{x}_{t-1}^{P1})$ is used as importance distribution, then w_t is proportional to

$$w_t \quad \propto \quad p(\mathbf{y}_t|\mathbf{y}_{1:t-1}\mathbf{x}_{0:t}^{P1})w_{t-1} \ .$$

Thus, the unnormalized weight $w_t^{(i)}$ requires just to evaluate $p(\mathbf{y}_t|\mathbf{y}_{1:t-1}\mathbf{x}_{0:t}^{P1(i)})$. The nice point of the Rao–Blackwell strategy is that this term corresponds to the evaluation of $\mathbf{y}_t$ in the one-step-ahead Kalman prediction of the observation density computed when $p(\mathbf{x}_{0:t}^{P2}|\mathbf{x}_{0:t}^{P1(i)}\mathbf{y}_{1:t})$ is solved (Appendix B may help to understand this relationship). This density corresponds to

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}\mathbf{x}_{0:t}^{P1(i)}) \quad \sim \quad \mathcal{N}(\hat{\mathbf{y}}_{t|t-1}, \mathbf{\Sigma}_{t|t-1}^{\mathbf{yy}}) \ .$$

where $\hat{\mathbf{y}}_{t|t-1}$ and $\mathbf{\Sigma}_{t|t-1}^{\mathbf{yy}}$ are the mean and covariance of the observations expected from a contour with B–spline contour points corresponding to the joint state $[\mathbf{x}_t^{P1(i)} \ \hat{\mathbf{x}}_{t|t-1}^{P2}]^T$.

Algorithm 7 details the pseudocode of a generic implementation of the Rao–Blackwellized PF proposed. In practice, this pseudocode corresponds just to an iterative method to update a Gaussian mixture model in $p_N(\mathbf{x}_{0:t}^{P1}\mathbf{x}_{0:t}^{P2}|\mathbf{y}_{0:t})$. In this mixture, one could interpret that several hypothesis on the sequence of $\mathbf{x}^{P1}$ values are maintained, and for each one of them, a KF is used to estimate the posterior on $\mathbf{x}^{P2}$. Some points are important to be remarked. Notice that $\mathbf{x}^{P1}$ and $\mathbf{x}^{P2}$ require uncorrelated dynamical processes, specified by the process and noise covariance matrices $(\mathbf{A}^{P1}, \mathbf{Q}^{P1})$ and $(\mathbf{A}^{P2}, \mathbf{Q}^{P2})$ respectively. We denote with $H_{|\mathbf{x}_t^{P1(i)}}$ a linear observation matrix established from a given concrete value of $\mathbf{x}_t^{P1(i)}$, that converts state $\mathbf{x}_t^{P2}$ onto the vector of their expected observations.

3.2.1 Experimental Evaluation

We have applied the Rao–Blackwell technique in the contour tracking problem in the following way. We have divided the contour state into two parts, isolating the state parameters with a non–linear relation with the shape synthesised (i.e., $\mathbf{x}^{P1} = [s_x \ s_y \ \theta]^T$) from the ones with a linear one (i.e., $\mathbf{x}^{P2} = [t_x \ t_y \ \mathbf{x}^D]^T$). In the experiments, 1000 particles have been used in the RBPF, which implied 1000 contour measurement per frame, the same number than a PF with 1000 particles. Figure 3.4 shows the achieved performance.

Results conclude that, on the one hand, the RBPF outperforms PFs in low–noise situations (noise $< 10\%$ frame pixels), due to the fact that part of the state is estimated analytically. However, in these cases the UKF performs better than the RBPF, as this algorithm estimates analytically the whole contour state. On the other hand, in high–noise situations, the opposite behaviour is observed. The RBPF does not improve the PF performance, because estimating $\mathbf{x}^{P2}$ analytically implies that the observation model is affected by Gaussian noise, and this is a wrong assumption in these cases. However, with respect to the UKF, a better performance is achieved, as estimating the state part $\mathbf{x}^{P1}$ by means of particles makes the tracker more robust to transitory missestimations of this state part. In fact, results achieved manifest the previously cited rule of thumb of estimation theory: if something can be solved

Algorithm 7 Rao–Blackwell Estimation Iteration

```
$[\{\mathbf{x}_{0:t}^{P1(i)}, (\hat{\mathbf{x}}_{0:t}^{P2(i)}, \boldsymbol{\Sigma}_{0:t}^{\mathbf{x}^{P2}\mathbf{x}^{P2}(i)}), w_t^{(i)}\}_{i=1}^{N}] = \mathrm{RBPF}[\{\mathbf{x}_{0:t-1}^{P1(i)}, (\hat{\mathbf{x}}_{0:t-1}^{P2(i)}, \boldsymbol{\Sigma}_{0:t-1}^{\mathbf{x}^{P2}\mathbf{x}^{P2}(i)}), w_{t-1}^{(i)}\}_{i=1}^{N}, \mathbf{y}_{1:t}]$

  for $i = 1$ to $N$ do
    Draw $\mathbf{x}_t^{P1(i)} \sim p(\mathbf{x}_t^{P1}|\mathbf{x}_{t-1}^{P1(i)})$

    Kalman Prediction Step

        $\hat{\mathbf{x}}_{t|t-1}^{P2} = \mathbf{A}^{P2}\mathbf{x}_{t-1|t-1}^{P2(i)}$
        $\boldsymbol{\Sigma}_{t|t-1}^{\mathbf{x}^{P2}\mathbf{x}^{P2}} = \mathbf{A}^{P2}\boldsymbol{\Sigma}_{t-1|t-1}^{\mathbf{x}^{P2}\mathbf{x}^{P2}(i)}\mathbf{A}^{P2^T} + \mathbf{Q}_t^{P2}$
    Kalman Update Step

        $\mathbf{K} = \frac{\boldsymbol{\Sigma}_{t|t-1}^{\mathbf{x}^{P2}\mathbf{x}^{P2}}\mathbf{H}_{|\mathbf{x}_t^{P1(i)}}^T}{\mathbf{H}_{|\mathbf{x}_t^{P1(i)}}\boldsymbol{\Sigma}_{t|t-1}^{\mathbf{x}^{P2}\mathbf{x}^{P2}}\mathbf{H}_{|\mathbf{x}_t^{P1(i)}}^T + \mathbf{R}_t^{P2}}$
        $\hat{\mathbf{x}}_{t|t}^{P2} = \hat{\mathbf{x}}_{t|t-1}^{P2} + \mathbf{K}\left(\mathbf{y}_{1:t} - \mathbf{H}_{|\mathbf{x}_t^{P1(i)}}\hat{\mathbf{x}}_{t|t-1}^{P2}\right)$
        $\boldsymbol{\Sigma}_{t|t}^{\mathbf{x}^{P2}\mathbf{x}^{P2}} = \left(\mathbf{I} - \mathbf{K}\mathbf{H}_{|\mathbf{x}_t^{P1(i)}}\right)\boldsymbol{\Sigma}_{t|t-1}^{\mathbf{x}^{P2}\mathbf{x}^{P2}}$
    Set $\mathbf{x}_{0:t}^{P1(i)} \triangleq \left[\mathbf{x}_{0:t-1}^{P1(i)}\ \mathbf{x}_t^{P1(i)}\right]$
    Set $\hat{\mathbf{x}}_{0:t}^{P2(i)} \triangleq \left[\hat{\mathbf{x}}_{0:t-1}^{P2(i)}\ \hat{\mathbf{x}}_{t|t}^{P2}\right]$ and $\boldsymbol{\Sigma}_{0:t}^{\mathbf{x}^{P2}\mathbf{x}^{P2}(i)} \triangleq \left[\boldsymbol{\Sigma}_{0:t-1}^{\mathbf{x}^{P2}\mathbf{x}^{P2}(i)}\ \boldsymbol{\Sigma}_{t|t}^{\mathbf{x}^{P2}\mathbf{x}^{P2}}\right]$

    Update weight
      $w_t^{(i)} = \Phi(\mathbf{y}_{1:t} - \mathbf{H}_{|\mathbf{x}_t^{P1(i)}}\hat{\mathbf{x}}_{t|t-1}^{P2}, \mathbf{H}_{|\mathbf{x}_t^{P1(i)}}\boldsymbol{\Sigma}_{t|t-1}^{\mathbf{x}^{P2}\mathbf{x}^{P2}}\mathbf{H}_{|\mathbf{x}_t^{P1(i)}}^T + \mathbf{R}_t^{P2})w_{t-1}^{(i)}$
        where $\Phi(\mu, \boldsymbol{\Sigma}) = \frac{1}{|2\pi\boldsymbol{\Sigma}|^{1/2}}e^{-\frac{1}{2}\mu^T\boldsymbol{\Sigma}^{-1}\mu}$
  end for

  Normalise weights $\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}} \quad \forall i = 1, \ldots, N$
  Compute $N_{eff} = \frac{1}{\sum_{i=1}^{N}(\tilde{w}_t^{(i)})^2}$
  if $N_{eff} < N_{deg}$ then
    $[\{\mathbf{x}_{0:t}^{P1(i)}, (\hat{\mathbf{x}}_{0:t}^{P2(i)}, \boldsymbol{\Sigma}_{0:t}^{\mathbf{x}^{P2}\mathbf{x}^{P2}(i)}), w_t^{(i)}\}_{i=1}^{N}] = \mathrm{RESAMPLE}[\{\mathbf{x}_{0:t}^{P1(i)}, (\hat{\mathbf{x}}_{0:t}^{P2(i)}, \boldsymbol{\Sigma}_{0:t}^{\mathbf{x}^{P2}\mathbf{x}^{P2}(i)}), \tilde{w}_t^{(i)}\}_{i=1}^{N}]$
  end if
```

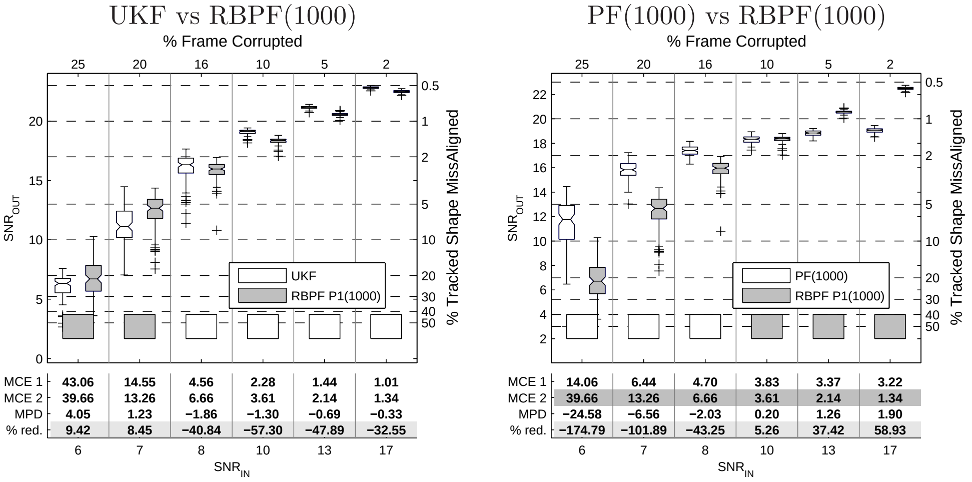


Figure 3.4: UKF vs PF vs RBPF comparison. Evaluation of the state partition $\mathbf{x} = [[s_x s_y \theta]^T [t_x t_y \mathbf{x}^D]^T]$. Left: UKF vs RBPF with 1000 particles. Right: PF vs RBPF, both with 1000 particles.

analytically, do not make it by means of particles. Thus, in low noise scenarios, estimating part of the state analytically is better than estimating the whole state using a PF, but it is worse than estimating the whole state analytically. On the other hand, in noisy situations, where the analytical solution of the contour tracking problem is bad posed, estimating part of the state by means of particles is better than estimating the whole state analytically, but worse than estimating the whole state using particles. The RBPF solution applied to contour tracking is half a way between analytic and particle–based solutions, and thus, combines the pros and cons of both methods. For this reason, it does not outperform both methods for a given noise situation. However, in specific applications where images could alternate high and low noise situations, it would be possible that the average performance of the RBPF could improve the one achieved by the UKF and the standard PF.

3.3 Contour Estimation by State Decoupling

Our third proposal to deal with the curse of dimensionality problem of PFs arises from the following observation: In most real contour tracking applications, the dynamics of global and local transformations are independent, and therefore they can be modelled separately. Could it be possible to estimate also them separately? In general, the answer to this question is clearly no, as observations manifest jointly the effect of the two modelled transformations. However, paying attention to specific contour tracking applications, it appears that in most cases a quite accurate estimation of the object global transformation $\mathbf{x}^R$ may be obtained independently of its local deformation $\mathbf{x}^D$. Indeed, commonly modelled objects deform just at localised regions of their outline, and the rest of the outline can be assumed as rigid (see Figure 3.5). Thus, changes observed in these rigid regions will be caused only by global contour transformations, and this can be used to estimate $\mathbf{x}^R$, whichever the parameters in $\mathbf{x}^D$ are.

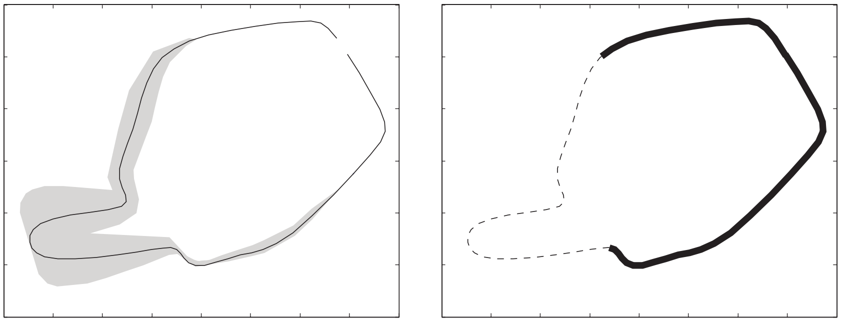


Figure 3.5: Almost rigid contour zone. Left: graphical representation of the contour representability of a shape space $\mathcal{L}(\mathbf{W}^D, \bar{\mathbf{q}})$. Right: the bold line marks the contour zones that vary minimally for any valid $\mathbf{x}^D$ parameterisation, and can approximately be assumed as rigid.

The estimation of $\mathbf{x}^D$ can not be done independently from $\mathbf{x}^R$, as rigid transformation affect the object contour globally. However, we will show that advantage can be taken from the isolated estimation of $\mathbf{x}^R$ to better estimate $\mathbf{x}^D$. Before arriving to that point, first is necessary to solve the estimation of $\mathbf{x}^R$ decoupled from $\mathbf{x}^D$, which is the topic of study in the next section.

3.3.1 Decoupled $\mathbf{x}^R$ Estimation

The basis to estimate $\mathbf{x}^R$ without being affected by contour deformations, is using with this objective the non–deformable regions of the contour to be tracked. Obviously, the existence of such regions depends on the specific object to be tracked, and requiring them to be perfectly rigid may be too restrictive for a practical use of this idea. To extend the applicability of the strategy proposed, the rigid region requirement is relaxed to use the less deformable contour zone to estimate $\mathbf{x}^R$. Obviously, the accuracy on this estimation will depend on the real rigidness of the contour part selected.

First of all, a representation of the rigidness/deformability of the points of the modelled contour is needed. As shown in Section 2.1.1, the space of contour deformations $\mathcal{L}(\mathbf{W}^D, \bar{\mathbf{q}})$ is established from a set of aligned contour examples, showing the different contour configurations to be modelled. From this same training set, the positional mean and covariance of contour points $(\bar{\mathbf{r}}, \mathbf{\Sigma}^{\mathbf{rr}})$ can be easily computed[4]. The positional contour variance (computed from the diagonal values of $\mathbf{\Sigma}^{\mathbf{rr}}$) inherently maintains the rigidness or deformability of points along the modelled contour. Thus, we propose to employ it to select the contour part that can confidently estimate $\mathbf{x}^R$

[4] Alternatively, $(\bar{\mathbf{r}}, \mathbf{\Sigma}^{\mathbf{rr}})$ can be computed from $\mathcal{L}(\mathbf{W}^D, \bar{\mathbf{q}})$ and the distribution of feasible $\mathbf{x}^D$ values.

confidently. We have proposed and tested two different methods for exploiting this idea: a metric–based approach, and a mask–based approach.

***Metric*–based Approach**

The result provided by any estimation algorithm is driven by the disparity observed between the predicted observations and the measured ones (i.e., the innovation). In the studied contour tracking problem, the innovation term corresponds to the disparity between the predicted and measured contour, projected along a given measurement direction. Concerning the global contour transformation $\mathbf{x}^R$, this projected disparity can be expressed as

$$\nu \quad = \quad \mathbf{y} - \mathbf{M}_m^T \mathbf{U}(\mathbf{W}^R_{(\mathbf{x}^R)}\mathbf{q}' + \mathbf{T}\mathbf{x}^R) \ . \tag{3.7}$$

where $\mathbf{M}_m^T$ is a matrix maintaining the measurement direction used at each contour point, $\mathbf{U}$ a matrix translating from contour points to contour samples, and $\mathbf{q}'$ any feasible contour configuration from $\mathcal{L}(\mathbf{W}^D, \bar{\mathbf{q}})$ (usually $\bar{\mathbf{q}}$). In order to estimate $\mathbf{x}^R$ the idea behind the metric–based approach is to weight the components in vector ν according to rigidness of their corresponding contour zone. In that way, measurements on rigid regions have bigger influence on the $\mathbf{x}^R$ estimation. In the approach used in this thesis, a natural way to implement that consist in setting the confidence given to observations according to the contour positional variance. In practical terms, this corresponds to adjust the covariance of the assumed observation noise $\mathbf{R}$, expressing that measures obtained in deformable zones are less reliable, since the information that supply about $\mathbf{x}^R$ may be distorted by the local contour deformation. Our proposal to implement this idea is given in Algorithm 8, which is based in computing a measure of the rigidity of each contour point to adjust accordingly $\mathbf{R}$.

***Mask*–based Approach**

This approach is a derivation of the previous one, based on making a rougher use of the contour covariance information. The idea is constructing a mask that delimits the rigid and non–rigid parts of a contour, in order to consider measurements only on the rigid contour zones. Analogously to the previous approach, we model this by adjusting the measurement noise covariance $\mathbf{R}$. In this case, their values are modified by the following criterion

$$\mathbf{R}(i,i) = \left\{ \begin{array}{cc} \mathbf{R}(i,i) & \mathbf{Factor}(i,i) \leq \text{treshold} \\ \sigma^2_{noMeas} & \text{otherwise} \end{array} \right. .$$

where **Factor** is the term detailed in Algorithm 8. However, in practice the implementation of this idea is completely different, being the strong point of this proposal. As measurements in non–rigid zones are *de facto* discarded, there is no need to process measurement lines on these regions. This reduces the computational effort required to estimate $\mathbf{x}^R$, which is always an interesting point.

Algorithm 8 Rigidity Definition

```
$[\mathbf{R}] = \mathbf{R}\text{–}\mathbf{RigidityTunning}\,[\mathbf{R}, \boldsymbol{\Sigma}^{\mathbf{rr}}]$

  Average contour variance computation:

    $\boldsymbol{\Sigma}^{\mathbf{pp}} = \left([\mathbf{I}_{N_s} \quad \mathbf{0}_{N_s}]\boldsymbol{\Sigma}^{\mathbf{rr}}[\mathbf{I}_{N_s} \quad \mathbf{0}_{N_s}]^T + [\mathbf{0}_{N_s} \quad \mathbf{I}_{N_s}]\boldsymbol{\Sigma}^{\mathbf{rr}}[\mathbf{0}_{N_s} \quad \mathbf{I}_{N_s}]^T\right)/2$

  Minimum average variance identification:

    $\sigma^2_{min} = \min \boldsymbol{\Sigma}^{\mathbf{pp}}(i,i) \quad \forall i = 1, \ldots, N_s$

  Rigidity Factor computation:

    $\mathbf{Factor} = \mathbf{0}_{N_s}$
    $\mathbf{Factor}(i,i) = \sigma^2_{min}/\boldsymbol{\Sigma}^{\mathbf{pp}}(i,i) \quad \forall i = 1, \ldots, N_s$

  Measurement noise variance tuning:

    $\mathbf{R}(i,i) = \mathbf{R}(i,i)/\mathbf{Factor}(i,i) \quad \forall i = 1, \ldots, N_s$
```

Evaluation of $\mathbf{x}^R$ estimation proposals

The methodology used to evaluate the performance of the two proposed methods is very similar to the one used in the preceding experiments. In this case, sequences showing a hand with a pointing finger, displaying simultaneously global and local transformations are processed by different trackers. As each tracker just estimates global transformation, this time the tracking performance is evaluated just from the mean contour error between the estimated contours and the ones corresponding to the ground truth parameters (Figure 3.6).

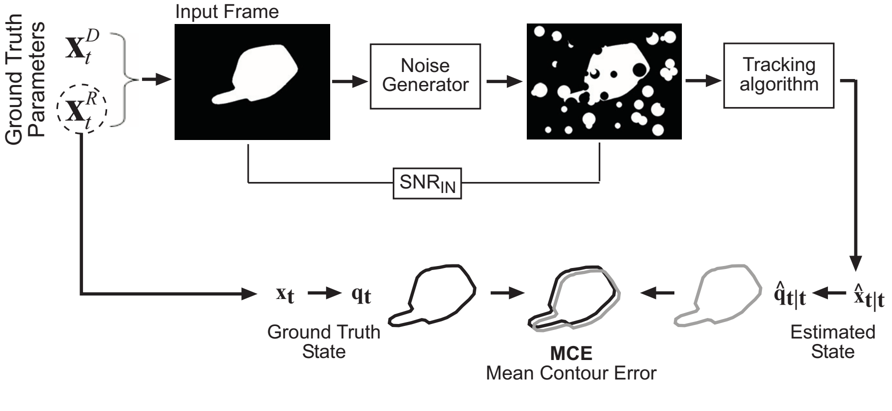


Figure 3.6: Procedure to evaluate the decoupled $\mathbf{x}^R$ estimation performance.

Figure 3.7 shows the $\mathbf{x}^R$ estimation results obtained by an UKF and a PF, adjusting the confidence given to the measured contour disparities in three different ways: uniformly, according to the metric–based approach, and according to the mask–based approach. The best performance is achieved by the PF using the mask–based approach, which is remarkable. It is the method using less information extracted from frames, but at the same time is the one most selective in using just *reliable* information, which seems to be the key point to obtain the best performance. The UKF behaves similarly in low–noise sequences, but when the presence of artifacts is high, using just part of the total contour to estimate $\mathbf{x}^R$ easily leads to misstracking. In these cases, results show that is better for the UKF to trust all the contour observations equitably, although some of them are distorted by local contour deformation.

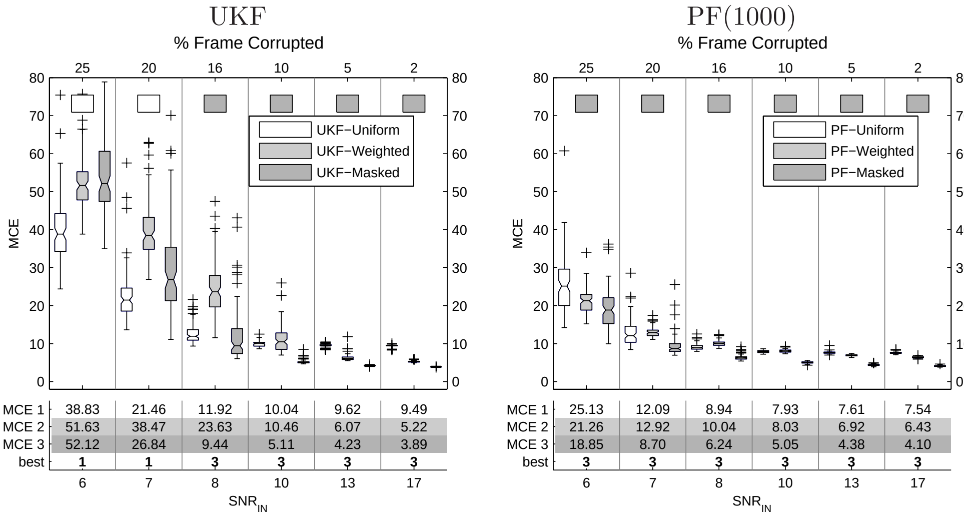
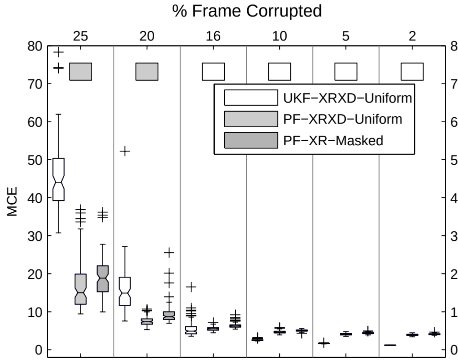


Figure 3.7: Estimation of the affine transformation of a contour which also presents local deformations. Mean contour error is determined from the disparity of contours synthesised using the estimated parameters, and the ground truth parameters.

Finally, it has been evaluated the performance of our best proposal (the mask–based approach using a PF) with respect to the joint estimation of $[\mathbf{x}^R\mathbf{x}^D]^T$ using an UKF and a PF (Figure 3.8). Results show that estimating at once the whole contour transformations, a more accurate $\mathbf{x}^R$ estimation is also obtained (by using an UKF in low–noise situations, and a PF in high–noise ones). This is due to the fact that, although scarcely, $\mathbf{x}^D$ alters the contour zone used to estimate $\mathbf{x}^R$, and this disturbs its precise estimation. In spite of that, the results achieved are very close to the ones of a PF estimating the whole state, certifying the validity of the proposal for our purposes.

3.3.2 Exploiting the $\mathbf{x}^R$ Estimation: Partitioned Sampling

Previous section has described two ways to decouple the estimation of $\mathbf{x}^R$ from $\mathbf{x}^D$, when there are nearly rigid zones in the tracked contour. Applying a similar approach



MCE 1	44.09	14.90	4.89	2.53	1.65	1.21
MCE 2	15.03	7.40	5.53	4.59	4.09	3.92
MCE 3	18.85	8.70	6.24	5.05	4.38	4.10
best	2	2	1	1	1	1
	6	7	8	10	13	17

$\mathrm{SNR}_{\mathrm{IN}}$

Figure 3.8: Complete vs partial contour transformation estimation. Mean contour error of the affine transform estimation, by estimating the whole contour transformation ($\mathbf{x}^R$ and $\mathbf{x}^D$), or just the affine transformation in the most rigid contour zones.

to estimate $\mathbf{x}^D$ is not possible, as long as rigid transformations affect the contour globally, and no point along it is insensible to $\mathbf{x}^R$. Thus, the estimation of $\mathbf{x}^D$ requires also taking $\mathbf{x}^R$ in consideration. However, thanks to the proposal of previous section, for estimating $\mathbf{x}^D$ at a given instant we have now the estimation of $\mathbf{x}^R$ available, and we can take advantage from that. We propose to use this $\mathbf{x}^R$ *pre–estimation* to define a better importance function for the PF. In principle, one may argue that this has little sense, as methods like the EKPF and the UKPF already allow to define such an importance function, but for the whole state $\mathbf{x}$. Thus, why just taking advantage of that for $\mathbf{x}^R$?. The point is that now, for the state part concerning $\mathbf{x}^R$, we are not limited to just using a Gaussian importance function: we can use the $\mathbf{x}^R$ *pre–estimation* to define it in terms of a weighted particle set. A method that exploits this idea exists, which is denoted as *Partitioned Sampling* (PS) [77] .

Partitioned Sampling is a generic term for the weight variance reduction strategy proposed in [77], based on modifying the SISR algorithm by inserting additional resampling operations in its procedure. It consists on dividing the state space into partitions, which can be propagated independently along time using their expected dynamics. Each partition is manipulated sequentially, propagating samples applying the partition dynamics followed by a *weighted resampling step*, which is the key point of this proposal. This resampling is driven by a *weighting function* that approximately evaluates the likelihood of particles from just the state partitions processed up to this point. The objective is, at each resampling step, to move particles closer to the posterior distribution to be estimated. After processing all partitions, most particles concentrate around the peaks of the posterior distribution, improving [5] the result

[5] This improvement is in terms of the variance of the weights of particles, which is diminished, attenuating in that way the degeneracy problem of the PF.

provided by the SISR algorithm. This procedure requires the following points to be fulfilled:

- the state has to be able to be decomposed into parts $\mathbf{x} = [\mathbf{x}^{Pi}]_{i=1}^{N}$ with uncorrelated dynamics,

- for each state partition $\mathbf{x}^{Pi}$ a weighting function $g_{Pi()}$ is needed (continuous and strictly positive), which ideally is peaked in the same region as the likelihood of the state part composed by partitions up to $\mathbf{x}^{Pi}$ (that is $[\mathbf{x}^{Pj}]_{j=1}^{i}$).

In the contour tracking application studied in this chapter, both conditions are accomplished by partitioning the state as $[\mathbf{x}_t^R \ \mathbf{x}_t^D]^T$. Figure 3.9 compares graphically the classical importance sampling procedure, with the partitioned sampling proposal. It shows how both algorithms generate a weighted sample set of a function $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}_2, \sigma^2\mathbf{I}_2)$ with $\mathbf{x} = [x \ y]^T$, sampling from an importance function $q(\mathbf{x}) = \mathcal{N}(\mathbf{0}_2, (4\sigma)^2\mathbf{I}_2)$. In the partitioned sampling case, a likelihood function $g(x) = \mathcal{N}(0, \sigma^2)$ is available for performing the weighted resampling. The example shows that partitioned sampling provides a sample set with minor weight variance. A drawback of this technique is that diversity is lost in the state part used in the weighted resampling, but as noted in Section 2.3.2 of previous chapter, there are methods to attenuate this problem.

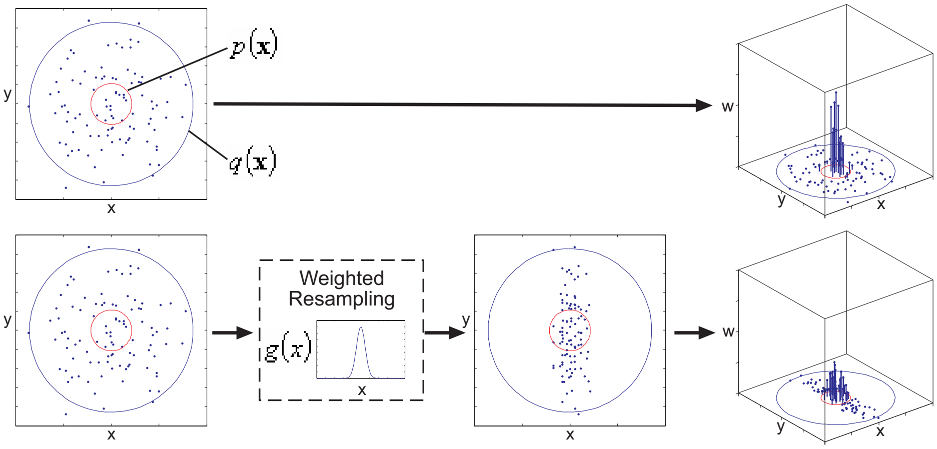


Figure 3.9: Importance Sampling (top) vs Partitioned Sampling (bottom) density approximation.

This technique has been commonly applied to multi-object tracking applications [77, 103], and to tracking applications concerning articulated models [78, 124]. Here we adapt it to the shape modelisation that we use. Before detailing how this idea is implemented for estimating $[\mathbf{x}^R\mathbf{x}^D]$, first the *weighted resampling* procedure is described more formally.

Weighted Resampling Method

Let $g(\mathbf{x})$ be a strictly positive, continuous function, denoted as *weighting function*. Weighted resampling of a particle set with respect to $g()$ is an operation which populates the peaks of $g()$ with particles, without altering the distribution actually represented by the particle set. This is carried out by a procedure derived from the importance sampling method.

Given a particle set $\{\mathbf{x}^{(i)}, \tilde{w}^{(i)}\}_{i=1}^{N}$ approximating $p(\mathbf{x})$ as

$$\widetilde{p_N}(\mathbf{x}) \quad = \quad \sum_{i=1}^{N} \tilde{w}^{(i)} \delta_{\mathbf{x}^{(i)}} \quad ,$$

first an importance function $q(\mathbf{x})$ is defined by evaluating a weighting function $g()$ on the elements of this particle set. That is

$$\begin{aligned} q(\mathbf{x}) &= \sum_{i=1}^{N} \frac{g(\mathbf{x}^{(i)})}{\sum_{j=1}^{N} g(\mathbf{x}^{(j)})} \delta_{\mathbf{x}^{(i)}} && (3.8) \\ &= \sum_{i=1}^{N} \rho^{(i)} \delta_{\mathbf{x}^{(i)}} \quad . && (3.9) \end{aligned}$$

The importance sampling method states that a distribution $p(\mathbf{x})$ can be approximated from samples $\mathbf{x}'^{(i)} \sim q(\mathbf{x})$, with an associated unnormalised weight equal to $p(\mathbf{x}'^{(i)})/q(\mathbf{x}'^{(i)})$. From that result, an alternative representation of $p(\mathbf{x})$ is given by the (unnormalised) particle $\{\mathbf{x}'^{(i)}, w'_i\}_{i=1}^{N}$, where

$$\begin{aligned} \mathbf{x}'^{(i)} &= \mathbf{x}^{(k_i)} \quad , \\ w'^{(i)} &= p(\mathbf{x}'^{(k_i)})/q(\mathbf{x}'^{(k_i)}) \quad , && (3.10) \\ &= \tilde{w}^{(ki)}/\rho^{(ki)} \quad , \end{aligned}$$

and k_i is the index identifying the i-th particle sampled from $q(\mathbf{x})$. Thus, weighted resampling is just a method to generate a weighted sample equivalent to a given weighted sample set. The idea is to obtain a sample set with lower weight variance, and the key point to achieve that is defining a good importance function $q(\mathbf{x})$. Unlike other proposals, now $q(\mathbf{x})$ is defined in terms of a weighted particle set, with particles $\mathbf{x}^{(i)}$ identical to ones of the approximated sampled distribution[6] $\widetilde{p_N}(\mathbf{x})$, but differently weighted. .

Algorithm 9 details the implementation of this procedure. Notice that now the RESAMPLE function returns the indexes $\{k_i\}_{i=1}^{N}$ identifying the particles selected from the resampled particle set.

3.3.3 Final PS Algorithm

Once described the weighted resampling method, it is quite direct taking advantage of the availability of $p(\mathbf{x}_t^R|\mathbf{y}_t)$, using it to define the importance function $\mathbf{q}()$ in Algorithm 9. Algorithm 10 shows the resultant partitioned sampling method, which is based on adding the weighted resampling step in the generic SISR process of PFs.

[6] Otherwise, the term in (3.10) could not be evaluated.

Algorithm 9 Weighted Resampling Iteration

```
$[\{\mathbf{x}'^{(i)}_{0:t}, w'^{(i)}_t\}_{i=1}^N] = \text{WEIGHTED RESAMPLING}[\{\mathbf{x}^{(i)}_{0:t}, w^{(i)}_t\}_{i=1}^N, g()]$

  for $i = 1$ to $N$ do
    $\varrho^{(i)} = g(\mathbf{x}^{(i)}_{0:t})$
  end for
  Compute $\rho^{(i)} = \frac{\varrho^{(i)}}{\sum_{j=1}^N \varrho^{(j)}} \quad \forall i = 1, \ldots, N$
  $[\{\mathbf{x}'^{(i)}_{0:t}\}_{i=1}^N, \{k_i\}_{i=1}^N] = \text{RESAMPLE}[\{\mathbf{x}^{(i)}_{0:t}, \rho^{(i)}\}_{i=1}^N]$
  for $i = 1$ to $N$ do
    $w'^{(i)}_t = w^{(k_i)}_t / \rho^{(k_i)}$
  end for
```

Algorithm 10 Partitioned Sampling Iteration

```
$[\{\mathbf{x}^{(i)}_{0:t}, w^{(i)}_t\}_{i=1}^N] = \text{PS}[\{\mathbf{x}^{(i)}_{0:t-1}, w^{(i)}_{t-1}\}_{i=1}^N, \mathbf{y}_{1:t}]$
  {Remember that $\mathbf{x}^{(i)}_{0:t-1} \triangleq [\mathbf{x}^{R(i)}_{0:t-1} \ \mathbf{x}^{D(i)}_{0:t-1}]$}
  for $i = 1$ to $N$ do
    Draw $\mathbf{x}^{R(i)}_t \sim p(\mathbf{x}^R_t | \mathbf{x}^{R(i)}_{0:t-1})$
    $\mathbf{x}^{R(i)}_{0:t} = [\mathbf{x}^{R(i)}_{0:t-1} \mathbf{x}^{R(i)}_t]$
  end for
  $[\{\mathbf{x}^{R(i)}_{0:t}, w^{,(i)}_t\}_{i=1}^N] = \text{WEIGHTED RESAMPLING}[\{\mathbf{x}^{R(i)}_{0:t}, w^{(i)}_{t-1}\}_{i=1}^N, p(\mathbf{y}_t | \mathbf{x}^R_t)]$
  for $i = 1$ to $N$ do
    Draw $\mathbf{x}^{D(i)}_t \sim p(\mathbf{x}^D_t | \mathbf{x}^{D(i)}_{0:t-1})$
    $\mathbf{x}^{D(i)}_{0:t} = [\mathbf{x}^{D(i)}_{0:t-1} \mathbf{x}^{D(i)}_t]$
    Set $\mathbf{x}^{(i)}_{0:t} = [\mathbf{x}^{R(i)}_{0:t} \mathbf{x}^{D(i)}_{0:t}]$
    Update weight $w^{(i)}_t = w^{,(i)}_t p(\mathbf{y}^{(i)}_t | \mathbf{x}^{(i)}_{0:t})$
  end for

  Normalise weights $\tilde{w}^{(i)}_t = \frac{w^{(i)}_t}{\sum_{j=1}^N w^{(j)}_t} \quad \forall i = 1, \ldots, N$
  Compute $N_{eff} = \frac{1}{\sum_{i=1}^N (\tilde{w}^{(i)}_t)^2}$
  if $N_{eff} < N_{deg}$ then
    $[\{\mathbf{x}^{(i)}_{0:t}, w^{(i)}_t\}_{i=1}^N] = \text{RESAMPLE}[\{\mathbf{x}^{(i)}_{0:t}, \tilde{w}^{(i)}_t\}_{i=1}^N]$
  end if
```

A point to be remarked is that the importance function $q(\mathbf{x}_t|\mathbf{x}^{(i)}_{0:t-1}\mathbf{y}_{1:t})$ in the SISR part of the algorithm corresponds to

$$p(\mathbf{x}_t|\mathbf{x}^{(i)}_{0:t-1}) \quad = \quad p(\mathbf{x}^R_t|\mathbf{x}^{R(i)}_{0:t-1})p(\mathbf{x}^D_t|\mathbf{x}^{D(i)}_{0:t-1}) \ ,$$

which is defined in terms of the dynamics of each state partition, which as required, are uncorrelated.

3.3.4 Additional Advantages of Partitioned Sampling

The different resampling stages inherent on a Partitioned Sampling algorithm allows a more flexible distribution of the computational resources in the estimation of the different state parts. Given a partitioned state, it may happen that one of its parts has a more erratic and unpredictable behaviour than the others, and consequently its dynamic model is more imprecise. This means that in every PF iteration, the region of its feasible predicted states is bigger than the one of the other parts. Thus, for a fixed number of particles, this region is less densely inspected for this part than for the others, resulting in a more unaccurate estimation of this part and, consequently, of the overall state. In these cases, a proper strategy can be adjusting the amount of particles assigned to each part according to their predictability, devoting more computational resources to the more erratic state parts. The weighted resampling step in Partitioned Sampling allows to implement this strategy, which was proposed in [78]. Figure 3.10 shows an scheme of this proposal.

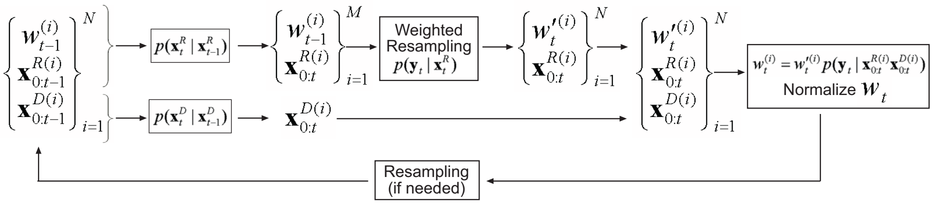
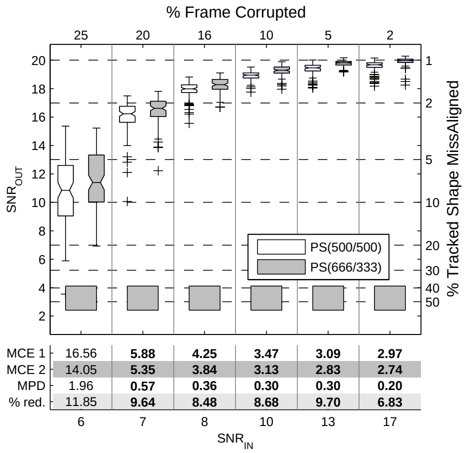

$\left\{ w^{(i)}_{t-1}, \mathbf{x}^{R(i)}_{0:t-1}, \mathbf{x}^{D(i)}_{0:t-1} \right\}^{N}_{i=1}$
$p(\mathbf{x}^R_t|\mathbf{x}^R_{t-1})$
$\left\{ w^{(i)}_{t-1}, \mathbf{x}^{R(i)}_{0:t} \right\}^{M}_{i=1}$
Weighted Resampling $p(\mathbf{y}_t|\mathbf{x}^R_t)$
$\left\{ w'^{(i)}_{t}, \mathbf{x}^{R(i)}_{0:t} \right\}^{N}_{i=1}$
$p(\mathbf{x}^D_t|\mathbf{x}^D_{t-1})$
$\mathbf{x}^{D(i)}_{0:t}$
$\left\{ w'^{(i)}_{t}, \mathbf{x}^{R(i)}_{0:t}, \mathbf{x}^{D(i)}_{0:t} \right\}^{N}_{i=1}$
$w^{(i)}_t = w'^{(i)}_t p(\mathbf{y}_t|\mathbf{x}^{R(i)}_{0:t}\mathbf{x}^{D(i)}_{0:t})$
Normalize w_t
Resampling (if needed)

Figure 3.10: Sketch of the implemented PS algorithm. The number of particles used in the different samplings of distributions carried out is detailed in the superindices of the propagated particle sets.

In the case–study analysed in this chapter, rigid transformations (i.e., the whole hand movement) can change more rapidly and unpredictably between frames, while deformations tend to change more slowly. Thus, a number of particles $M \gg N$ should be used to inspect the feasible $\mathbf{x}^R$ states to be more robust to their unpredictability. Then, the resultant distribution is resampled back to N samples in the weighting resampling step, to recompose the complete state and then compute its likelihood. This strategy allows, for fixed computational resources, achieving a better tracking performance. Notice that to estimate correctly $\mathbf{x}^D$, measurement lines have to provide good measurements of that. As long as these measurement lines are placed relative to the hand position (i.e., $\mathbf{x}^R$ values), the strategy of using more samples to estimate $\mathbf{x}^R$ reverts in obtaining a better $\mathbf{x}^D$ estimation, and, thus, a better overall state estimation.

Performance Evaluation

Our first experiment has been designed to check whether there is a real advantage on managing a different number of particles in the different sample evaluation phases of the PS algorithm, namely the weighting resampling of the $\mathbf{x}^R$ importance samples, and the likelihood evaluation of the complete state $[\mathbf{x}^R\ \mathbf{x}^D]^T$. Two different cases are evaluated, one distributing the same number of particles in the both phases, that is $(N/2, N/2)$, and one distributing particles following the proportion $(2N/3, N/3)$. Figure 3.11 shows, for the case $N = 1000$, the benefits of investing more particles in the weighted resampling step of $\mathbf{x}^R$, which improves to tracking performance in all noise situations considered.

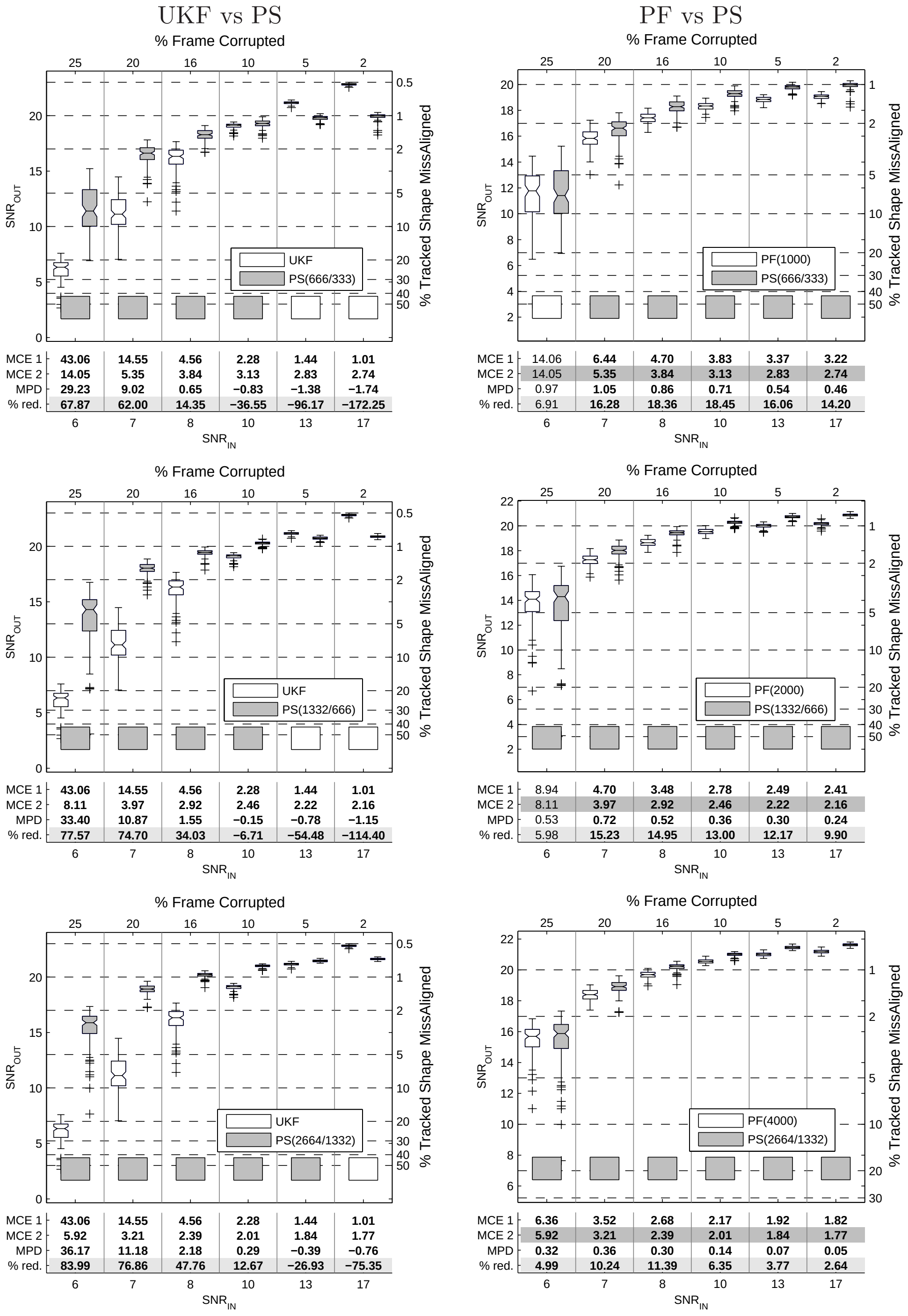


	6	7	8	10	13	17
MCE 1	16.56	**5.88**	**4.25**	**3.47**	**3.09**	**2.97**
MCE 2	14.05	**5.35**	**3.84**	**3.13**	**2.83**	**2.74**
MPD	1.96	**0.57**	**0.36**	**0.30**	**0.30**	**0.20**
% red.	11.85	**9.64**	**8.48**	**8.68**	**9.70**	**6.83**

Figure 3.11: Performance of two PS, with different proportion of particles devoted to the weighted resampling step. A better performance is achieved when more particles are devoted to the weighted resampling step.

Our next experiment compares the performance of the PS $(2N/3, N/3)$ against the one of UKFs and PFs. Figure 3.12 shows the results obtained for a PS with an amount of particles equivalent to a PF with 1000, 2000 and 4000 particles. In low noise situations, the UKF performs better than the PS, showing a behaviour similar to the one observed with the standard PF. However, the comparative study of both methods certify the superior performance of the PS with respect to an standard PF in all situations, although in the worst noise situation (i.e., $SNR_{IN} = 6dB$) their difference is usually not statistically significant. The performance improvement achieved is due to the inner weighted resampling step in the PS, which delimits more tightly the *interesting* part of the state space to be explored. Thanks to that, more particles get closer to the ideal state parameters, and the tracking accuracy increases.

We have also compared the PS against the UKPF, the method that up to this point has shown in general the best performance. Figure 3.13 shows that the UKPF achieves a better performance, as long as the implicit linear approximation of the optimal importance function is accurate. However, in high noise situations this does



SNR_{IN}	6	7	8	10	13	17
MCE 1	43.06	14.55	4.56	2.28	1.44	1.01
MCE 2	14.05	5.35	3.84	3.13	2.83	2.74
MPD	29.23	9.02	0.65	−0.83	−1.38	−1.74
% red.	67.87	62.00	14.35	−36.55	−96.17	−172.25

SNR_{IN}	6	7	8	10	13	17
MCE 1	14.06	6.44	4.70	3.83	3.37	3.22
MCE 2	14.05	5.35	3.84	3.13	2.83	2.74
MPD	0.97	1.05	0.86	0.71	0.54	0.46
% red.	6.91	16.28	18.36	18.45	16.06	14.20

SNR_{IN}	6	7	8	10	13	17
MCE 1	43.06	14.55	4.56	2.28	1.44	1.01
MCE 2	8.11	3.97	2.92	2.46	2.22	2.16
MPD	33.40	10.87	1.55	−0.15	−0.78	−1.15
% red.	77.57	74.70	34.03	−6.71	−54.48	−114.40

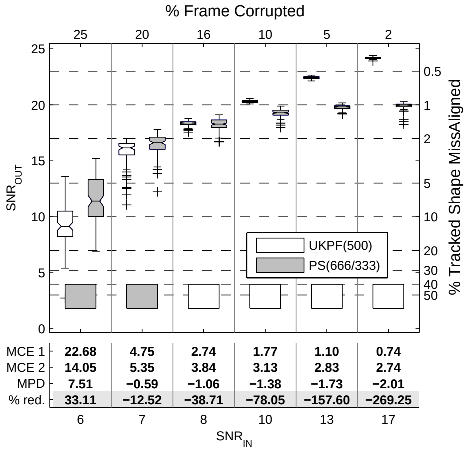
SNR_{IN}	6	7	8	10	13	17
MCE 1	8.94	4.70	3.48	2.78	2.49	2.41
MCE 2	8.11	3.97	2.92	2.46	2.22	2.16
MPD	0.53	0.72	0.52	0.36	0.30	0.24
% red.	5.98	15.23	14.95	13.00	12.17	9.90

SNR_{IN}	6	7	8	10	13	17
MCE 1	43.06	14.55	4.56	2.28	1.44	1.01
MCE 2	5.92	3.21	2.39	2.01	1.84	1.77
MPD	36.17	11.18	2.18	0.29	−0.39	−0.76
% red.	83.99	76.86	47.76	12.67	−26.93	−75.35

SNR_{IN}	6	7	8	10	13	17
MCE 1	6.36	3.52	2.68	2.17	1.92	1.82
MCE 2	5.92	3.21	2.39	2.01	1.84	1.77
MPD	0.32	0.36	0.30	0.14	0.07	0.05
% red.	4.99	10.24	11.39	6.35	3.77	2.64

Figure 3.12: UKF vs PF vs PS comparison. Performance achieved for different amounts of particles considered (from top to bottom, 1000, 2000 and 4000 particles in a PF, and the equivalent amount for a PS with the proportion $(2N/3, N/3)$).

not happen, and its performance degradates. The PS maintains its better performance even in high–noise situations, as the redistribution of particles done by the weighted resampling step adapts to any multimodal distribution that can be suggested by the observations, being thus less disturbed by the non-Gaussian noise artifacts.



MCE 1	22.68	4.75	2.74	1.77	1.10	0.74
MCE 2	14.05	5.35	3.84	3.13	2.83	2.74
MPD	7.51	−0.59	−1.06	−1.38	−1.73	−2.01
% red.	33.11	−12.52	−38.71	−78.05	−157.60	−269.25

Figure 3.13: UKPF vs PS Comparison. Performance achieved by the two methods, using both an amount of particles equivalent to 1000 particles in a PF.

3.4 Conclusions

In this chapter it has been studied the problem of contour tracking, when this contour presents simultaneously local and global (affine) transformations. We have worked in two main concepts concerning this problem:

- the development of a new methodology to estimate global transformations ($\mathbf{x}^R$) uncoupled form local contour deformations ($\mathbf{x}^D$);
- the novel application of three different variance reduction techniques to the problem of contour tracking using PFs, and their comparative study.

With regard to the first point, we have proposed an strategy for estimating $\mathbf{x}^R$ uncoupled form $\mathbf{x}^R$, based on giving a bigger confidence to observations in contour zones less likely to suffer from local deformations. Two alternatives has been studied to modulate the reaction of estimation methods to observations at different contour zones. In a case–study, the validity of both proposals has been tested, obtaining the best $\mathbf{x}^R$ estimation using a PF that ignores observations in locally deformable contour zones. Although the accuracy achieved is slightly inferior to the one obtained tracking the whole contour transformations, it proves useful to assist a PF to improve their performance through the partitioned sampling technique.

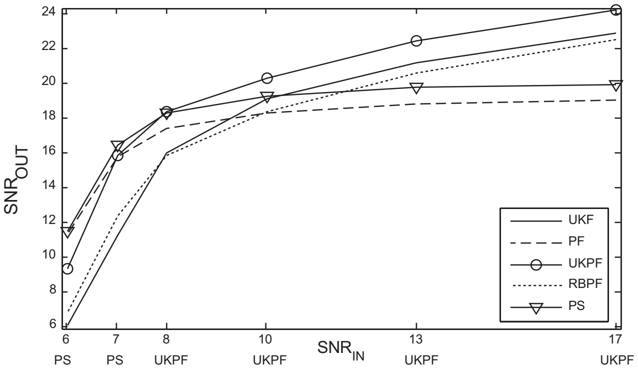
The work developed concerning the use of variance reduction techniques in contour tracking applications has been devoted to attenuate the *curse of dimensionality* problem affecting PFs. As now global and local contour transformations have to be recovered from frames, the dimensionality of the state to be estimated increases, and this diminishes the effectivity of PFs. Three different techniques have been newly adapted to the contour tracking problem, and their performance evaluated:

- The Unscented Kalman Particle Filter.
- The Rao–Blackwellized Particle Filter.
- The Partitioned Sampling Particle Filter.

These three proposed techniques try to improve the performance of PFs from different perspectives, namely:

- computing for each particle a linear version of the optimal sampling function to be used for propagating particles in the SISR procedure;
- estimating part of the state analytically;
- pre-evaluating the likelihood of particles from part of their content, which allows to propagate particles closer to the real contour state.

Figure 3.14 summarises the mean performance[7] of the three proposed techniques, as well as the one achieved by the UKF and PF. To make a fair comparison between them, the number of measurement processes that each method[8] makes on each frame has been fixed to 1000.



6	7	8	10	13	17
PS	PS	UKPF	UKPF	UKPF	UKPF
PF	UKPF	PS	PS	UKF	UKF
UKPF	PF	PF	UKF	RBPF	RBPF
RBPF	RBPF	UKF	RBPF	PS	PS
UKF	UKF	RBPF	PF	PF	PF

Figure 3.14: Comparison of all proposed methods. For each noise situation evaluated, the rank list of methods is also detailed.

The UKPF is the method that provides better results, overperforming in most situations both the UKF and the PF. It allows to reduce very significantly the number

[7] Notice that in boxplots is remarked the median performance.
[8] Except for the UKF, which makes a single one.

of particles with respect to a PF, for a given level of performance. Only in very noisy situations this method has problems, performing worse than a classical PF.

The RBPF applied to the contour tracking problem has turn out to be a method half–a–way between Kalman–based and Particle–based filters. In this context, it combines the pros and cons of both approaches, remaining in between of the performances achieved by these methods. Thus, in low noise situations, performs worse than an UKF (the whole state can be well estimated analytically), but better than a PF. In high noise situations, performs better than an UKF, but worse than a PF (the assumption that part of the state can be estimated analytically does not holds, as noise artifacts are non–Gaussian). Hence, if an specific contour tracking application has a very well defined noise situation, it will not be clearly the best choice. On the other hand, in an application where images could alternate high and low noise situations, we think that it would be possible that the average performance of the RBPF could improve the one achieved by the UKF and the standard PF.

Finally, the contour tracker based on PS performs always as good as or even better than the classical PF solution. Hence, it is a clear candidate to replace this algorithm, as long as part of the state can be approximately estimated decoupled from the other part. Although this technique is the one more reliable in improving the standard PF performance in all noise situations, this does not mean that it is the best method in all situations. In low noise situations, the UKPF is clearly the best performing method. We claim that the reason is that, in low noise situations, the optimal sampling function can be approximates accurately using a Gaussian distribution, which is what the UKPF does. The PS models this distribution by means of a particles, and this can not overcome the analytic representation of the Gaussian distribution. In high noise situations the optimal importance function turns out to be non–Gaussian, and the PS achieves a better performance, as the particle representation that uses can model the shape of any arbitrary density.

Chapter 4

A Multi Model Approach to Contour Tracking

The previous chapter has addressed the general problem of contour tracking (i.e., the estimation of local and global contour transformations), and has proposed the use of different techniques to improve the performance of PFs in this problem. Applied techniques were based on different strategies to reduce the amount of particles required by PFs to achieve a given performance, as this amount grows exponentially with the dimension of the state to be estimated. Techniques analysed were based either on methods to propagate particles closer to the real target state (UKPF and PS), or on estimating part of the state analytically (RBPF). In this chapter we propose a novel solution to this same problem from a different perspective. As the main problem of PFs is the state dimensionality, we propose to reduce this dimensionality by using an alternative modelisation of the contour to be tracked. The basic idea is replacing the classical single shape model used in the previous chapters, by a collection of multiple shape models of lower dimensionality. This chapter describes a new methodology proposed to implement this strategy, detailing algorithms designed to solve the following topics:

- the automatic generation from training data of a *good* collection of shape models of low dimensionality;
- the management of multiple shape models to estimate the state of a contour target along time.

Our proposal to learn multiple models from training data derives from an strategy commonly used to improve the specificity of single shape models. A general problem concerning generative models is that their representability is far bigger than the one desired. That is, there are parameter configurations that, when applied to the model, generate unfeasible instances of the modelled object. In these cases, it is very useful delimiting the region in the space of parameters that generates valid target representations, to use it properly for a desired task (target synthesis, target tracking, etc.). In the context of shape modelling, this region is commonly denoted as Subspace of Valid Shapes (SVS) . Usually, in applications where the modelled shape can display a big variability of configurations, the SVS has a complex topology, consisting of unconnected regions of irregular shape. Due to the impossibility of expressing the SVS with a single parametric model, it is popularly described by a combination of hyper-ellipsoidal regions given by a Gaussian Mixture Model (GMM) . In this chapter we propose a method to unsupervisedly learn the GMM corresponding to the SVS of a given shape model. Then, using the learned GMM, we propose a novel methodology to derive a collection of low dimensional models to replace the original shape model.

Once the collection of models is available, the next step is formalising their use for tracking purposes. Now, tracking not only implies estimating the contour transformations at each instant, but also determining which model has to be used in this estimation. Thus, an additional discrete variable is added to the target state, identifying the active model at each instant. This discrete variable together with the collection of linear shape models conforms what is commonly referred as a Jump Markov System (JMS) , and we propose a new algorithm to use it for contour tracking purposes, which is based on the PS technique.

Next sections detail the different algorithms required to generate a collection of linear models from training data, and their use in tracking. Section 4.1 reviews different approaches to determine the SVS of a given shape model, and proposes an unsupervised algorithm to parameterise a GMM and solve this task. After evaluating quantitatively the performance of the proposed method, Section 4.2 details our proposal to generate a collection of models from the GMM adjusted to the SVS of a given shape model. For a case study application, it is shown that by using multiple models, a better[1] description of the target variability is achieved. Section 4.3 reviews different proposals in the bibliography to use multiple models in tracking applications, and proposes a PS–based algorithm to fulfil this task. It is shown in a case–study application, that the use of multiple shape models leads to a better tracking performance.

4.1 SVS Modelisation

Given the training set $\{\mathbf{q}_i\}_{i=1}^N$ used to construct a linear shape model $\mathcal{L}(\mathbf{W}, \bar{\mathbf{q}})$, the model parameters $\{\mathbf{c}_i\}_{i=1}^N$ that better reconstruct them are obtained from[2]

$$\mathbf{c}_i = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T(\mathbf{q}_i - \bar{\mathbf{q}}) \ .$$

This projection of the training set onto the shape space reveals the region in this space where the parameters of desired shapes are confined. Hence, this projection can be used to delimit the SVS model. Thus, determining the SVS turns to be the estimation of an accurate representation of the distribution of the projections $\mathbf{c}_{1:N}$. The usual approach consists in modelling this distribution by means of a combination of simple parametric functions, which provides an accurate representation of the modelled data, requiring a reduced number of parameters (see Figure 4.1).

[1] In term of the Bayesian Information Criterion.

[2] If the Euclidean norm has been used in the PCA of $\{\mathbf{q}_i\}_{i=1}^N$ that determines $\mathbf{W}$, this expression reduces to $\mathbf{c}_i = \mathbf{W}^T(\mathbf{q}_i - \bar{\mathbf{q}})$. In our case, a norm specific for spline contour points has been used (see [18]).

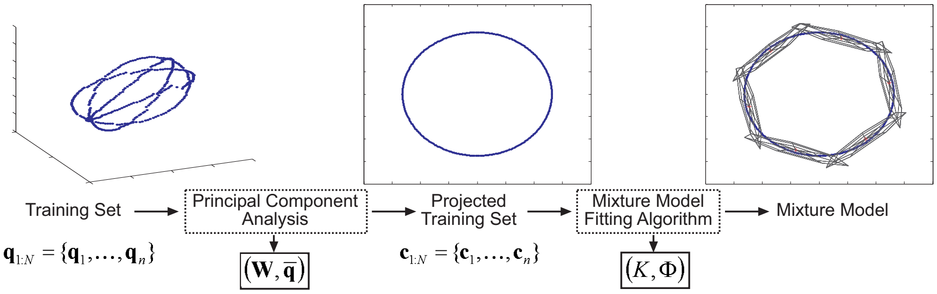

Training Set
$\mathbf{q}_{1:N} = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$
Principal Component Analysis
$(\mathbf{W}, \bar{\mathbf{q}})$
Projected Training Set
$\mathbf{c}_{1:N} = \{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$
Mixture Model Fitting Algorithm
(K, Φ)
Mixture Model

Figure 4.1: Construction of a Linear Shape Model constrained with a GMM.

Different proposals have been done to describe the distribution of $\mathbf{c}_{1:N}$. In [56], the k-means algorithm is used to represent it by means of a combination of hyperellipsoids. Similarly, in [30] a Gaussian Mixture Model is used, fitted to $\mathbf{c}_{1:N}$ using the Expectation Maximisation algorithm (EM) [37]. Both proposals are based on iterative schemes that fit a given initial parameterisation of a mixture model to a target distribution. Depending on their initial conditions, these schemes can converge to a local maximum, which poorly represent the real data distribution. Another weak point of these proposals is that they fix a priori the amount of components considered (i.e., the *order* of the mixture model K), compromising the quality of the distribution approximation obtained.

Our approach in this thesis is based on modelling the SVS using a GMM too. The novelty of our proposal is that, in order to fit the GMM to the distribution of $\mathbf{c}_{1:N}$, we propose an algorithm that does not demand to choose a priori the number of GMM components to consider, neither their initial conditions. The proposed method has the aim of generating an accurate representation of the SVS, using a parsimonious number of components. In that way, methods that use the SVS model to increment the robustness of a given tracking application (for instance, [23, 57]) will require less computational resources. Next section details the proposed method.

4.1.1 Unsupervised Parameterisation of GMM

Mixture Models increase the representability of classical parametric distributions by establishing a linear combination of several of them. In that way, they describe more precisely a dataset distribution, while requiring a reduced amount of parameters in comparison with the modelled data. The probability density function of such models can be expressed as

$$p(\mathbf{x}) = \sum_{i=1}^{K} p(\mathbf{x}|\Theta_i) w_i \ , \tag{4.1}$$

where K indicates the amount of mixture components, $p(\mathbf{x}|\Theta_i)$ the probability density function with parameters Θ_i of the i-th component of the mixture, and w_i its weight or mixing coefficient. The values w_i are positive, and sum up to unity.

Given a Mixture Model with K components, the parameters $(w_i, \Theta_i)_{i=1}^{K}$ are usually determined by maximising their likelihood for a given dataset. This is a non-trivial task in general, but various procedures exist when mixture components are Gaussian density functions [14]. These methods follow an iterative scheme that fit a given initial parameterisation of the mixture model to data.

The majority of popular and computationally feasible techniques using mixture models consider Gaussian components, given by

$$p(\mathbf{x}|\Theta_i) = \frac{1}{|2\pi\boldsymbol{\Sigma}_i|^{1/2}} \exp^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\mu_i)} \ , \qquad (4.2)$$

where $\Theta_i = (\mu_i, \boldsymbol{\Sigma}_i)$ maintains the Gaussian mean and covariance. For a fixed number of components K, the mixture model is characterised by the set of parameters $\boldsymbol{\Phi}_{1:K} = \{(w_i, \mu_i, \boldsymbol{\Sigma}_i)\}_{i=1}^{K}$. Given an initial guess of parameters K and $\boldsymbol{\Phi}_{init}$, the EM algorithm (see Algorithm 11) is the standard method used to determine the maximum likelihood estimation of $\boldsymbol{\Phi}_{1:K}$. As remarked before, depending on $\boldsymbol{\Phi}_{1:K}$ the algorithm can converge to a local maximum, as the likelihood of a GMM is not unimodal, and obviously the value of K is crucial to correctly model the real distribution.

Algorithm 11 EM Algorithm

```
$[\boldsymbol{\Phi}_{1:K}] = \text{EM}\left[K, \boldsymbol{\Phi}_{init}, \{\mathbf{x}_i\}_{i=1}^{N}\right]$

  Set initial mixture model $\boldsymbol{\Phi}_{1:K} \leftarrow \boldsymbol{\Phi}_{init}$
  repeat
    {Expectation Step}
    for $i = 1$ to $K$ do
      for $j = 1$ to $N$ do
        $e_{(i,j)} = p(\Theta_i|\mathbf{x}_j) = \frac{p(\mathbf{x}_j|\Theta_i)w_i}{\sum_{k=1}^{K} p(\mathbf{x}_j|\Theta_k)w_k}$
      end for
    end for
    {Maximisation Step}
    for $i = 1$ to $K$ do
      $w_i = \frac{\sum_{j=1}^{N} e_{(i,j)}}{N}$
      $\mu_i = \frac{\sum_{j=1}^{N} e_{(i,j)}\mathbf{x}_j}{\sum_{j=1}^{N} e_{(i,j)}}$
      $\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^{N} e_{(i,j)}(\mathbf{x}_j-\mu_i)(\mathbf{x}_j-\mu_i)^T}{\sum_{j=1}^{N} e_{(i,j)}}$
    end for
  until convergence
```

There exist different strategies to establish a GMM with a proper number of components. One possibility consists in inferring the value of K by making a restricted search on this parameter. First a range of feasible values for K has to be given, assuming that it contains the desired optimal number of components. Then, for each value, several random initialisations of the GMM are generated and processed with the EM algorithm, trying to avoid to get trapped in a local maximum . This generates multiple GMM solutions, from where the one that maximises or minimises a given cost function is selected. The works in [90, 98] describe and evaluate the performance of several criteria proposed to make this model-order selection.

Another possibility to fully parameterise GMMs is taking advantage of the information provided by data clustering methods. In [45] good results are obtained by initialising GMMs of different order from the application of the Ward's clustering method to the dataset to be modelled. These GMMs are then refined with the EM algorithm, and the one that maximises a given criterion is selected.

Another strategy is to search for the optimal GMM by means of Markov Chain Monte Carlo methods. The work in [95] proposes a Bayesian hierarchical model for mixtures, where they detail an assumed prior distribution for the parameters of a univariated GMM. Then, the Reversible Jump Sampling Algorithm is applied, refining an initial guess of the GMM by updating the component parameters, splitting/merging components, or adding/deleting components, scanning with this process the space of GMMs of variable order defined in the Bayesian priors. A variation of this technique is applied in [97] to several clustering problems, where the author remark that even this sofisticated and theoretically well–sustained proposal can get stuck in suboptimal solutions. However, the major problem of this technique is its high demand on computation.

No one of the reviewed methodologies assures an optimal solution in the modelling of a given data distributions using a GMM. In this thesis we present a novel proposal that also does not, but provides an strategy to find suboptimal solutions avoiding an exhaustive search in the space of feasible GMMs, and the definition of explicit priors of the GMM parameters, which is very interesting in practice. It consists of a greedy algorithm that iteratively fits a GMM to a given dataset, increasing at each iteration the order of the GMM if the data modelisation is unsatisfactory for a given criterion. The implicit assumption in the proposed procedure is that when a data distribution is modelled using a GMM, this data can be grouped in normally–distributed clusters. Starting from this idea, we propose to embed the parameterisation of GMMs inside a classical hierarchical clustering technique.

Classical Cluster Analysis

Very popular methods used in Cluster Analysis are the Hierarchical Clustering Algorithms (HCAs). These algorithms group data from the computation of their dendrogram: a branching diagram representing a hierarchy based on the degree of similarity between elements in a data set. Among these algorithms, there exist two different philosophies: Agglomerative HCAs and Divisive HCAs. The formers construct a dendrogram by initially establishing a cluster for every element of the dataset, and fusing them iteratively until a unique cluster is obtained. The key point of these methods is the fusion criterion, and many proposals have been done concerning that topic. Figure 4.2 shows examples of Agglomerative HCA for the simple single linkage and the Ward's method (the one most commonly used). The single Linkage method fuses at each iteration the two groups that are closer, defining the distance between groups as the distance between the closest pair of objects. On the other hand, the Ward's method [120] evaluates at each iteration the union of every cluster pair. The two cluster whose union results in a cluster of minimum variance are fused. Figure 4.2 shows, for a given dataset, that the Single Linkage method suggests the presence of just a single cluster, while the Ward's method identifies clearly the two present clusters. No wonder, therefore, that the Ward's algorithm has been applied to establish the initialisation of GMMS, prior to the execution of the EM algorithm [45].

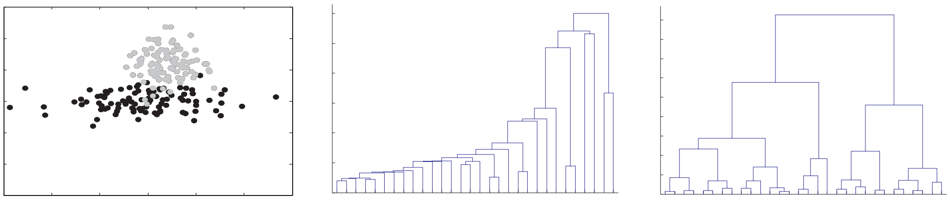


Figure 4.2: Dataset and its associated dendrogram. Left: data from two Gaussian distributions. Center: dendrogram applying the single linkage method. Right: dendrogram applying the Ward's method.

Divisive HCAs propose to analyse hierarchically the data but in the opposite direction. Starting out from a cluster grouping the whole dataset, this one is iteratively subdivided such that the objects in one subgroup are far from the objects in the other groups. When a given number of subclusters is obtained, the algorithm stops. This approach has one major problem: there is no splitting criterion that assures a good partition of data, so incorrect groups are propagated along the cluster hierarchy. This problem can be avoided if the splitting decision does not mean establishing definitive clusters in the dataset, but a sign that the actual clustering gives a poor description of the dataset. From this idea emerges our proposal described in this thesis, which consists in embedding the EM algorithm inside the basic procedure of divisive HCA.

Adaptive Gaussian Mixture Modelling

Given a dataset $\mathbf{x}_{1:N} = \{\mathbf{x}_i\}_{i=1}^{N}$, the simplest GMM to describe it corresponds to the one with a single component, which is defined by the mean and covariance $(\mu, \mathbf{\Sigma})$ of the data to be modelled. So an initial GMM is defined by $\mathbf{\Phi}_1 = \{(w_1 = 1, \mu_1 = \mu, \mathbf{\Sigma_1} = \mathbf{\Sigma})\}$. If this representation describes precisely the real data distribution, there is no need to explain it using more components. Otherwise, more components should be considered. So a test to check the *fitness* of the GMM with respect to the data is needed. We propose to check that by verifying if the components of the GMM delimit normally distributed data clusters in $\mathbf{x}_{1:N}$. For the first GMM considered, this means evaluating a *Test of Multivariate Normality* on $\mathbf{x}_{1:N}$. If the test fails, the Gaussian is split in two, and the EM algorithm is applied next to make converge this new defined GMM.

When the GMM has more than one component, the dataset is subdivided into clusters by assigning each data element $\mathbf{x}_i$ to the component which provides the maximum conditional probability $p(\Theta_i|\mathbf{x}_i)$. Then, the normality of each cluster is checked in decreasing order of its weight w_i, analysing first the components that support a bigger part of the dataset. When a cluster fails the test, its corresponding component is split in two, the EM is applied to the new GMM, and the overall process is repeated again. In this procedure an stop condition is added to avoid splitting components that model *underpopulated* clusters, as in these cases the normality test usually fails. Algorithm 12 summarises the proposed procedure. Figure 4.3 shows an example of the procedure carried out.

Algorithm 12 Adaptive Gaussian Mixture Model Fitting Algorithm

```
[$\mathbf{\Phi}_{1:k}$] = AGMMFA [$\mathbf{x}_{1:N}$]

  Set $k \leftarrow 1$, $\mathbf{\Phi}_{1:k} \leftarrow (1, \mu, \mathbf{\Sigma})$ from $\underline{\mathbf{x}}$ statistics.
  Set Split ← TRUE
  while Split = TRUE do
    Group $\mathbf{x}_{1:N}$ into clusters $\{\mathbf{x}^j_{1:N_j}\}^k_{j=1}$ by $\mathbf{x}_i \in \mathbf{x}^j_{1:N_j}$ if $j = \arg\max_k p(\Theta_k|\mathbf{x}_i)$
    Sort components $\{\mathbf{\Phi}_i\}^k_{i=1}$ in $\mathbf{\Phi}_{1:k}$ in decreasing order of $w_i$.
    Set $i \leftarrow 1$,Split ← FALSE
    while $i \leq k$ and Split = FALSE do
      if $\#(\mathbf{x}^i_{1:N_i}) >$ threshold & NORMALITY TEST($\mathbf{\Phi}_i$) = FALSE then
        Split ← TRUE
      end if
      if Split = TRUE then
        $(\mathbf{\Phi}_a, \mathbf{\Phi}_b) \leftarrow$ SPLIT COMPONENT($\mathbf{\Phi}_i$)
        $\mathbf{\Phi}_{1:k-1} \leftarrow \mathbf{\Phi}_{1:k} \backslash \{\mathbf{\Phi}_i\}$.
        $\mathbf{\Phi}_{1:k+1} \leftarrow \mathbf{\Phi}_{1:k-1} \cup \{\mathbf{\Phi}_a\} \cup \{\mathbf{\Phi}_b\}$.
        $k \leftarrow k + 1$.
        $\mathbf{\Phi}_{1:k} \leftarrow \mathrm{EM}(k, \mathbf{\Phi}_{1:k}, \mathbf{x}_{1:N})$.
      end if
      $i \leftarrow i + 1$.
    end while
  end while
```

Multivariate Normality Test To perform this test the multivariate generalisation of the Wald-Wolfowitz Two-sample test proposed in [46] has been used. This test determines if two sample sets $\mathbf{x}_{1:N_x}$ and $\mathbf{y}_{1:N_y}$ are drawn from the same distribution, by evaluating if both are very intermixed. In the case of multivariate data, this can be discerned by constructing the Minimal Spanning Tree (MST) of the joined dataset, and counting the links l that join elements from $\mathbf{x}_{1:N_x}$ to $\mathbf{y}_{1:N_y}$.

From the observed MST topology the following statistic can be defined

$$w = \frac{l - E[l]}{\sqrt{Var[l]}} \quad , \tag{4.3}$$

where $E[l]$ and $Var[l]$ are the expectation and the variance of the value of l respectively, considering all the possible connectivity relationships between the edges of a MST of the joined dataset $\{\mathbf{x}_{1:N_x}, \mathbf{y}_{1:N_y}\}$. Under the hypothesis that $\mathbf{x}_{1:N_x}$ and $\mathbf{y}_{1:N_y}$ are independent random samples from a given distribution, it is shown in [46] that for large sample sets the distribution of w approaches an standard normal distribution $\mathcal{N}(0, 1)$. This can be used to judge if two set are equally distributed. If w is a negative value far from zero, this manifests a spatial separation between $\mathbf{x}_{1:N_x}$ and $\mathbf{y}_{1:N_y}$. On

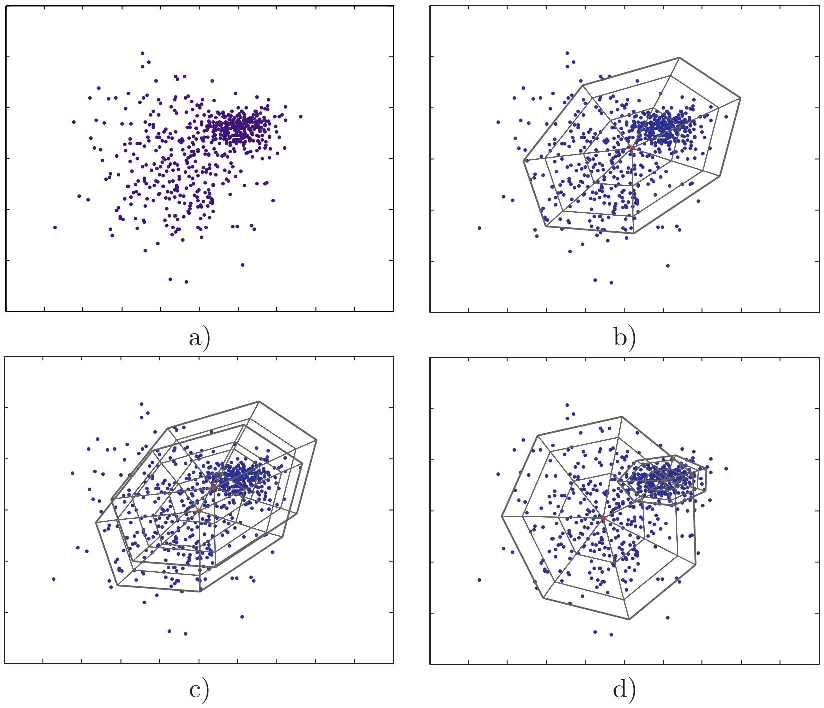


Figure 4.3: Unsupervised method to parameterise a GMM. a) Dataset. b) Initial GMM. c) The test of Multivariate Normality fails and the component is split. d) Final GMM obtained by applying the *EM* algorithm.

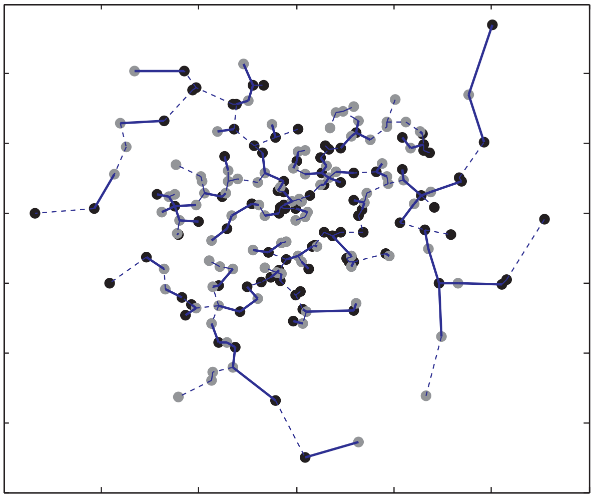


Figure 4.4: Minimal Spanning Tree of two datasets corresponding to the same distribution. *Solid lines* correspond to X-to-Y edges.

the other hand, the bigger the w value, the more intermixed are the two sets. Thus, a one–sided test on the w value is appropriate for detecting if the distribution of $\mathbf{x}_{1:N_x}$ and $\mathbf{y}_{1:N_y}$ are equal or not. For a significance level of 0.05, this means that $\mathbf{x}_{1:N_x}$ and $\mathbf{y}_{1:N_y}$ are assumed to be equally distributed when $w \geq -1.645$.

This same strategy can be readapted to design a test of multivariate normality. Given a data cluster $\mathbf{x}_{1:N_x}$, its sample mean and covariance $(\mu_{\mathbf{x}}, \mathbf{\Sigma}_{\mathbf{x}})$ is computed. Then, a dataset $\mathbf{y}_{1:N_y}$ is randomly generated, according to the computed $\mathbf{x}_{1:N_x}$ statistics. If both $\mathbf{x}_{1:N_x}$ and $\mathbf{y}_{1:N_y}$ are intermixed (the test on w is satisfied), this means that $\mathbf{x}_{1:N_x}$ is normally distributed. An evaluation of the performance of this test concerning finite sample sets is detailed in [104].

Component Splitting For splitting a component, there is no criterion that one can guarantee as the correct one (i.e., the one that best describes data). Otherwise, hierarchical clustering methods would perform perfectly. However, the inaccuracy in splitting a component is not a very critical problem in the proposed method, as long as posteriorly this partition is *rectified* by the EM algorithm. To implement this task, we have evaluated two different splitting strategies, in order to check the relevance of the splitting criterion used on the results obtained. The methods checked are based on splitting components:

- using a classical clustering algorithm,
- using an heuristical method.

The first proposal uses the Ward's method to construct a dendrogram on the data associated to the component to be split, and then separates this data in two clusters. The mean and covariance of the data in each cluster is computed, and used to establish the initial parameters of the new components. The weight of each component is established proportional to the amount of data elements that it has associated. With this process, it is expected a good initialisation for at least one of the two components.

The second proposal is a *blind* method based on arbitrarily subdividing the component into two partially overlapped new ones, which together approximately recover the divided one. The objective of this partitioning method is suggesting the necessity of splitting a component, more than establishing a good initial partition. The method proposed replaces a component $(w_i, \mu_i, \mathbf{\Sigma}_i)$ by $(w_a, \mu_a, \mathbf{\Sigma}_a)$, $(w_b, \mu_b, \mathbf{\Sigma}_b)$, where

$$\begin{aligned} w_a &= w_b = \frac{1}{2} w_i \ , \\ \mu_a &= \mu_i + \frac{\alpha}{3}\mathbf{v} \ , \\ \mu_b &= \mu_i - \frac{\alpha}{3}\mathbf{v} \ , \\ \mathbf{\Sigma}_a &= \mathbf{\Sigma}_b = \mathbf{\Sigma}_i \ , \end{aligned}$$

being $\mathbf{v}$ the eigenvector of $\mathbf{\Sigma}_i$ with the biggest associated eigenvalue, and α is the square root of the biggest eigenvalue. Figure 4.5 shows a one-dimensional example of this splitting process. It generates two identical Gaussians, both slightly separated along the principal axis of the divided one. This proposal relies on the EM to refine such arbitrary data partition.

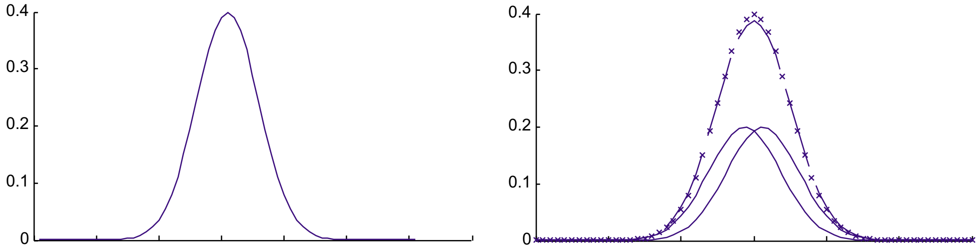


Figure 4.5: Heuristical Gaussian splitting process. Left: initial distribution. Right: mixture resulting of the splitting process (*solid line*). Sum of the elements in the mixture (*dashed line*). Initial distribution (*crosses*).

Experimental Evaluation

We have tested the effectivity of our proposal by applying it to model data drawn from randomly generated GMMs. The quality of the resultant GMMs in representing the data is evaluated using the Bayesian Information Criterion (BIC) [102]. This criterion is equivalent to he Minimum Description Length criterion used in information/coding theory, and simultaneously judges both the complexity and the accuracy of a given model. It gives a mathematical formulation of the principle of parsimony in model building, so penalises models that with more mixture components give only a little improvement in their fitting to data. The BIC is probably the most commonly used model selection criterion, and its good performance is reported in the comparative study done in [98]. Being $\mathbf{x}_{1:N}$ a set of N samples, the BIC coefficient of a GMM whose amount of parameters[3] is M is computed by:

$$\mathrm{BIC}(\mathbf{x}_{1:N}) = \sum_{i=1}^{N} \log p(\mathbf{x}_i) - \frac{1}{2} M \log(N) \quad . \tag{4.4}$$

The first term on the right side of (4.4) judges the accuracy of the approximation, while the second one judges its complexity. Using this expression, we have evaluated our proposed algorithm with the following procedure: first, a *ground truth* GMM is randomly parameterised and then sampled to generate a dataset to be modelled. Then, this dataset is processed by different modelling methods, obtaining from them a given GMM solution. The tested methods are:

M_1**:** the proposed AGMMFA with the Ward's splitting component method;

M_2**:** the proposed AGMMFA with the *blind* splitting component method;

M_3**:** the proposal in [45], i.e., an exhaustive search method that establishes EM initialisations using the Ward's clustering technique.

[3] The amount of values that have to be maintained to represent computationally $\{(w_i, \mu_i, \mathbf{\Sigma}_i)\}_{i=1}^{K}$.

The solutions obtained by each method M_i are evaluated computing their respective BIC value (denoted as BIC_{M_i}), and comparing it against the one of the *ground truth*(GT) GMM (BIC_{GT}). Plots in Figure 4.6 show the paired difference between BIC_{M_i} and BIC_{GT}, for experiments carried out on datasets sampled from GMMs of different number of components. For each number of components considered, 100 different experiments have been done.

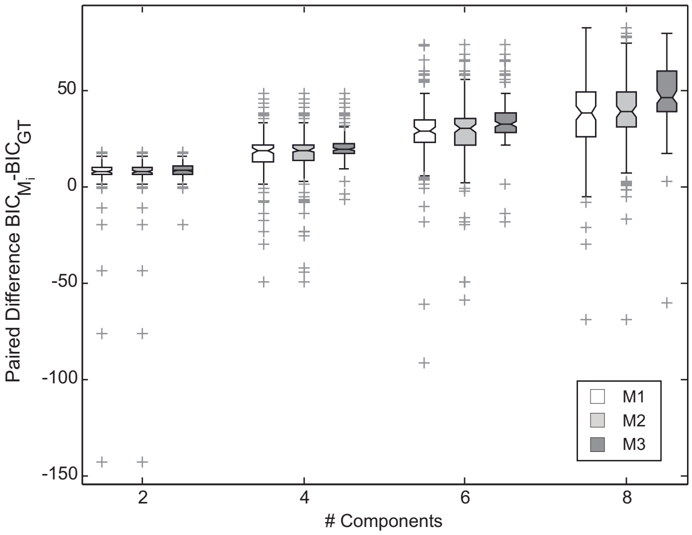


Figure 4.6: Comparison of the BIC value of the analysed methods. Paired difference between the BIC value of each M_i method and the BIC value of the ground truth model, in datasets of 400 samples.

Positive differences in the plot manifest that the GMM estimated by the method represent the modelled datasets better than the GT model does. This behaviour is due to the fact that the datasets analysed have an sparse number of samples (400), and in these cases is natural that the estimated GMMs fit the distribution of samples more tightly than the GT model does. The fact that the difference increases with the amount of components of the GT model manifest the fact that in many cases, less GMM components than in the GT model are needed to represent the dataset distribution. For instance, if the dataset to be modelled is generated from a GMM which has two nearly identical components, the data elements generated from this two components will be described by a single component in the learned GMM. Thus, the generated model will be simpler, and will have a better BIC value.

Results show that our proposed methodology (methods M_1 and M_2) performs comparably to the exhaustive search method M_3, except for GMMs of 8 components, whose performance is slightly worse. However, even in this case, the solution achieved is satisfactory, in comparison with the BIC value of the GT model. Hence, although our proposal is not optimal, achieves a fair performance for our purposes, using a low cost greedy strategy.

With respect to the splitting component method used, the performance of the proposed method does not change noteworthy (i.e., the difference in performance between M_1 and M_2 is not statistically significant). We claim that this is due to the fact that when a component is split into two, in most of the cases it should be really split into a major number of partitions, according to the data cluster that models. Therefore, any partition in two components is a *provisional* rough model, and for this reason no splitting criterion seems to be a priori better than another one.

We have also checked the performance of our algorithm to model datasets not generated from GMMs. Figure 4.7 displays convergences obtained in three-dimensional datasets, showing a qualitatively good behaviour of the proposed method to represent complex data distributions (the ones commonly used in the literature).

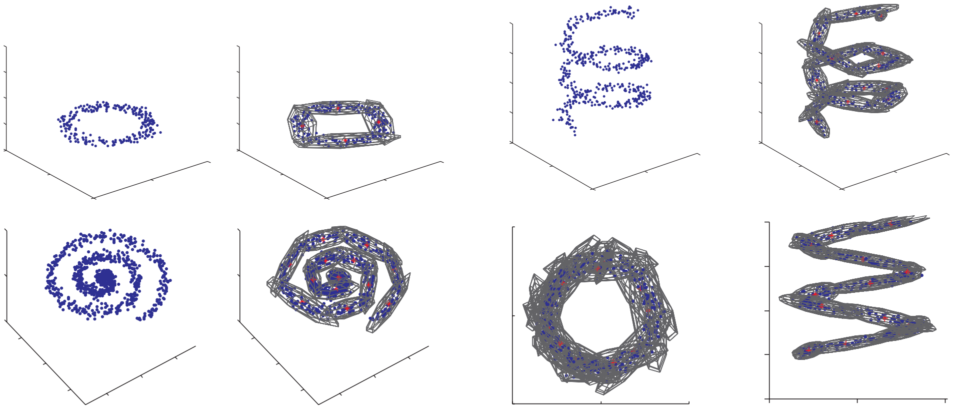


Figure 4.7: Convergence of the algorithm to datasets not generated from GMMs. Left: circle and spiral-shaped dataset. Right: different views of the model generated for a spring-shaped dataset.

4.2 Generation of Multiple Shape Models

Given the GMM modelling the SVS, each one of its components groups elements whose shape variability can be delimited by a Gaussian distribution. We take advantage on that, proposing to replace the linear model constrained by a GMM, with a collection of linear models, each one constrained by a single Gaussian. First, given the training set $\mathbf{q}_{1:N}$ used to learn the shape space $\mathcal{L}(\mathbf{W}, \bar{\mathbf{q}})$, its elements are projected in this space, generating the set of model parameterisations $\mathbf{c}_{1:N}$. Each projected training sample $\mathbf{c}_i$ is assigned to the SVS component k that maximises its likelihood. This defines K subsets $\mathbf{q}^i_{1:N_i} \in \mathbf{q}_{1:N}$, where

$$\mathbf{q}^i_{1:N_i} = \cup\{\mathbf{q}_j\} \ \forall \ \mathbf{c}_j \mid i = \arg\max_{k=1}^{K} p(\mathbf{c}_j|\Theta_k) w_k \ .$$

This gathers elements in the training set, organising them in clusters of similar shape. What we propose is defining an specific linear shape model for each of them, constraining its respective SVS with a single Gaussian. Figure 4.8 summarises this procedure.

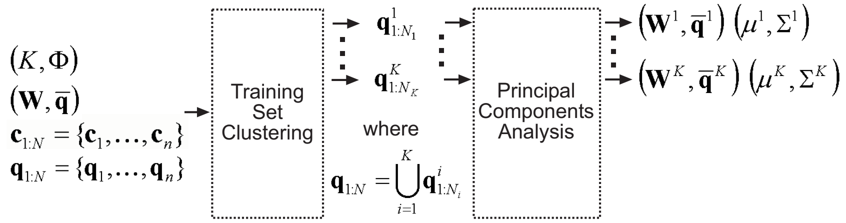

(K,Φ)
$(\mathbf{W},\bar{\mathbf{q}})$
$\mathbf{c}_{1:N} = \{\mathbf{c}_1,\ldots,\mathbf{c}_n\}$
$\mathbf{q}_{1:N} = \{\mathbf{q}_1,\ldots,\mathbf{q}_n\}$
Training Set Clustering
$\mathbf{q}^1_{1:N_1}$
$\mathbf{q}^K_{1:N_K}$
where
$\mathbf{q}_{1:N} = \bigcup_{i=1}^{K} \mathbf{q}^i_{1:N_i}$
Principal Components Analysis
$(\mathbf{W}^1,\bar{\mathbf{q}}^1)\ (\mu^1,\Sigma^1)$
$(\mathbf{W}^K,\bar{\mathbf{q}}^K)\ (\mu^K,\Sigma^K)$

Figure 4.8: Construction of multiple linear shape models, from a linear shape model constrained with a mixture of models.

As elements in $\mathbf{q}^i_{1:N_i}$ are commonly quite similar, it seems logical to expect that the *local* shape model $\mathcal{L}(\mathbf{W}^i,\bar{\mathbf{q}}^i)$ generated from them should require a dimensionality less or equal than the global model $\mathcal{L}(\mathbf{W},\bar{\mathbf{q}})$. In order to assure that, we propose a criterion to determine the local model dimensionality, which promotes a reduced model dimension while maintaining an accuracy similar to the one of the global model. This criterion determines the local model dimensionality as the minimal N_{d^i} for which the mean square reconstruction error of the local model is smaller or equal to the obtained using the global model. That is,

$$\min_{N_{d^i}} \mid \sum_{\mathbf{q}_j\in\mathbf{q}^i_{1:N_i}} \left\|\mathbf{q}_j - (\mathbf{W}^i\mathbf{c}^i_j + \bar{\mathbf{q}}^i)\right\|^2_{\mathcal{M}} \leq \sum_{\mathbf{q}_j\in\mathbf{q}^i_{1:N_i}} \left\|\mathbf{q}_j - (\mathbf{W}\mathbf{c}_j + \bar{\mathbf{q}})\right\|^2_{\mathcal{M}} \quad ,$$

where $\mathbf{c}^i_j$ is the projection of $\mathbf{q}_j$ onto the $\mathcal{L}(\mathbf{W}^i,\bar{\mathbf{q}}^i)$ space, and $\|\ \|_{\mathcal{M}}$ denotes the norm used, which is based on the metric matrix $\mathcal{M}$ for B-Spline curves introduced in Section 2.1.1. The most important benefit of proceeding in that way, more than the gain of accuracy achieved by the new models generated (that is set at a feasible minimum), is the reduction in dimensionality of shape spaces. This lower dimensionality is very convenient for several reasons:

- it reduces the computational load of tracking algorithms, since less parameters have to be estimated. Thus, for a given computational time, it allows considering either a better temporal sampling of sequences, or more sofisticated processes to extract observations from images.

- it improves the specificity of the shape model, as the degrees of freedom of the shape parameterisations are reduced. This favours to obtain more robust performances.

- it improves the performance of PFs, as fewer particles will be required to achieve a desired performance.

Despite these arguments favouring the practical application of the multiple model approach, it remains to ascertain if this new proposal represents indeed an improvement in shape modelling. This subject is studied in the next section.

4.2.1 Comparison between Shape Models

To compare the multiple model proposal with the linear shape model constrained with a GMM, again we propose to use the BIC (see expression (4.4)). As the models to be compared maintain separately a shape model and a constraint model (i.e., the SVS model), we define their overall BIC value as

$$BIC(\text{Model}) \quad = \quad BIC(\text{Linear Shape Model}) + BIC(\text{SVS Model}) \ . \tag{4.5}$$

In the following, the factors required to compute the terms in (4.5) are detailed and derived.

BIC of a GMM–Constrained Linear Shape Model

Given a training set of shapes $\mathbf{q}_{1:N} = \{\mathbf{q}_i\}_{i=1}^N$ of dimension N_{d_o}, and a linear shape space $\mathcal{L}(\mathbf{W}, \bar{\mathbf{q}})$ of dimension N_{d_i} (so that $\mathbf{W}$ is a $N_{d_o} \times N_{d_i}$ matrix), the terms involved in Equation (4.4) are defined as

$$\begin{aligned} \log\big(p(\mathbf{q}_{1:N}|(\mathbf{W}, \bar{\mathbf{q}}))\big) &= \sum_{i=1}^{N} -\frac{1}{2}\|\mathbf{q}_i - (\mathbf{W}\mathbf{c}_i + \bar{\mathbf{q}})\|_{\mathcal{M}}^2 \ , \\ M &= N_{d_o}(N_{d_i} + 1) \ , \end{aligned}$$

where $\log\big(p(\mathbf{q}_{1:N}|(\mathbf{W}, \bar{\mathbf{q}}))\big)$ corresponds to the mean squared reconstruction error of the shape model, and M to the number or real numbers required to parameterise the model (that is, the total real numbers in $(\mathbf{W}, \bar{\mathbf{q}})$).

To validate the SVS model in $\mathbf{\Phi}$, it is evaluated how well the projected training set $\mathbf{c}_{1:N} = \{\mathbf{c}_i\}_{i=1}^N$ is represented by the GMM. So if the number of components in the GMM is K, the terms in the BIC expression are

$$\begin{aligned} \log\big(p(\mathbf{c}_{1:N}|\mathbf{\Phi}) &= \sum_{i=1}^{N} \log p(\mathbf{c}_i|\mathbf{\Phi}) = \sum_{i=1}^{N} \log\left(\sum_{j=1}^{K} p(\mathbf{c}_i|\mu_j, \mathbf{\Sigma}_j) w_j\right) \ , \\ M &= K\left(\frac{N_{d_i}(N_{d_i}+1)}{2} + N_{d_i} + 1\right) - 1 \ . \end{aligned}$$

The value of M computes the parameters required to maintain a GMM, which consists of K Gaussians, each one with a full covariance matrix (i.e., requiring to maintain $N_{d_i}(N_{d_i}+1)/2$ values), a mean (a vector of N_{d_i} values), and a weight. As the weights of components have to sum up to unity, just $K-1$ values have to be really maintained.

BIC of a collection of Gaussian-Constrained Linear Shape Models

For the multiple model proposal, the terms of the BIC expression are computed as

$$\begin{aligned} \log\big(p(\mathbf{q}_{1:N}|\mathbf{W}^1, \ldots, \mathbf{W}^K, \bar{\mathbf{q}}^1, \ldots, \bar{\mathbf{q}}^K)\big) &= \sum_{k=1}^{K} \sum_{\mathbf{q}_i \in \mathbf{q}1:N_k{}^k} -\frac{1}{2}\|\mathbf{q}_i - (\mathbf{W}^k\mathbf{c}_i^k + \bar{\mathbf{q}}^k)\|_{\mathcal{M}}^2, \\ M &= KM_{d_o} + \sum_{k=1}^{K} M_{d_i^k} N_{d_o} \ . \end{aligned}$$

where $N_{d_i^k}$ is the dimension of the k-th shape space, and $\mathbf{c}_i^k$ is the projection of $\mathbf{q}_i$ in $\mathcal{L}(\mathbf{W}^k, \bar{\mathbf{q}}^k)$. The value of M accounts for the total number of real values required to maintain the K linear models.

For the SVS model, the BIC terms reduce to

$$\begin{aligned}\log\left(p(\mathbf{c}_{1:N}|\mu^1,\ldots,\mu^K,\mathbf{\Sigma}^1,\ldots,\mathbf{\Sigma}^K)\right) &= \sum_{k=1}^{K}\sum_{\mathbf{c}_i\in\mathbf{c}^k_{1:N_k}} \log p(\mathbf{c}_i|\mu^k,\mathbf{\Sigma}^k)\ , \\ M &= \sum_{k=1}^{K}\frac{N_{d_i^k}(N_{d_i^k}+3)}{2}\ ,\end{aligned}$$

where now, for each model, the SVS is delimited by just a single Gaussian.

Using the presented expressions, the single and the multiple shape model approaches can be compared for a modelling problem. In the next section, this is done in a case–study application concerning the generation of a shape model for the outline of a walking pedestrian.

Experiments

Modelling the outline of a walking person is an interesting application to check our multiple model approach, since the silhouette of a pedestrian present discontinuous shape changes along time due to the legs movement, which will result in a non-linear SVS. Given a sequence of a walking pedestrian, a training set has been generated by delimiting its outline in the different frames. From the PCA of the training sequence, an initial linear shape model $\mathcal{L}(\mathbf{W}, \bar{\mathbf{q}})$ of dimension 7 is determined. Figure 4.9 shows the projection of the training set on the 3 first dimensions of the shape space, and the SVS model obtained with the method described in Section 4.1.1.

From the SVS model, the collection of linear shape models is constructed. Figure 4.10 displays the dimensionality and the mean shape $\bar{\mathbf{q}}^i$ of each model generated. Clearly, the SVS components unsupervisedly learned identify a reduced number of silhouette states which synthesise a typical pedestrian walking cycle.

To observe the benefits obtained by generating multiple shape models of low dimensionality, Figure 4.11 compares qualitatively the shape deformation modelled by each modelling method. Plots show the shape variability corresponding to some SVS component of a global model, and the variability of its corresponding local shape model. The local models are more specific in localising the zones of the silhouette that deform.

Table 4.1 shows the result of applying the BIC expression in Equation (4.5) to the different models. In order to model only the shape of the pedestrian, in terms of the BIC the single model is better. That is, although the multi–model approach represents more accurately the training set (i.e., its reconstruction error is inferior to the one of the single model approach (i.e., -909 versus -1096)), in BIC terms performs worse than the single model approach (its BIC value (-6238) is further from zero). The accuracy gained using 7 models to represent the pedestrian outline does not justify the amount of model parameters to be maintained. On the other hand, considering just the SVS model, it results that the multiple model approach is

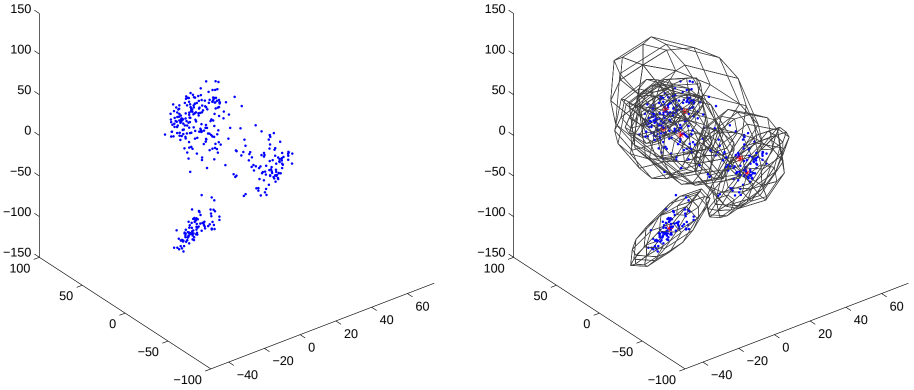


Figure 4.9: SVS of the pedestrian training set. Left: training data projected into the 3 first dimensions of a 7-dimensional Shape space. Right: the SVS model consists in a GMM of 7 components.

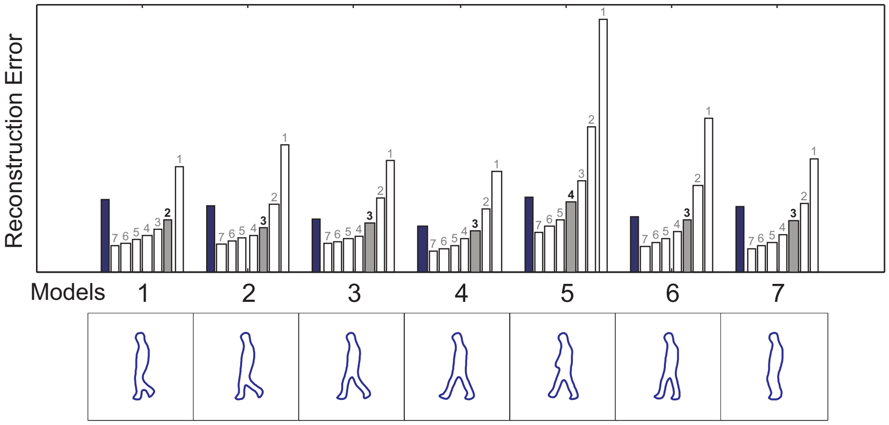


Figure 4.10: Details of the multiple models generated. Up: Mean squared reconstruction error for each new shape model generated. For each model, there is a group of bars, where the black one shows the reconstruction error of the initial 7-dimensional shape model applied to $\mathbf{q}^i_{1:N_i}$. The rest of the bars show the error of the model constructed from $\mathbf{q}^i_{1:N_i}$, considering dimensionalities from 7 (the *white bar* closer to the *black one*) till 1. The *gray bar* marks the dimensionality selected. Down: Mean shape of the generated models.

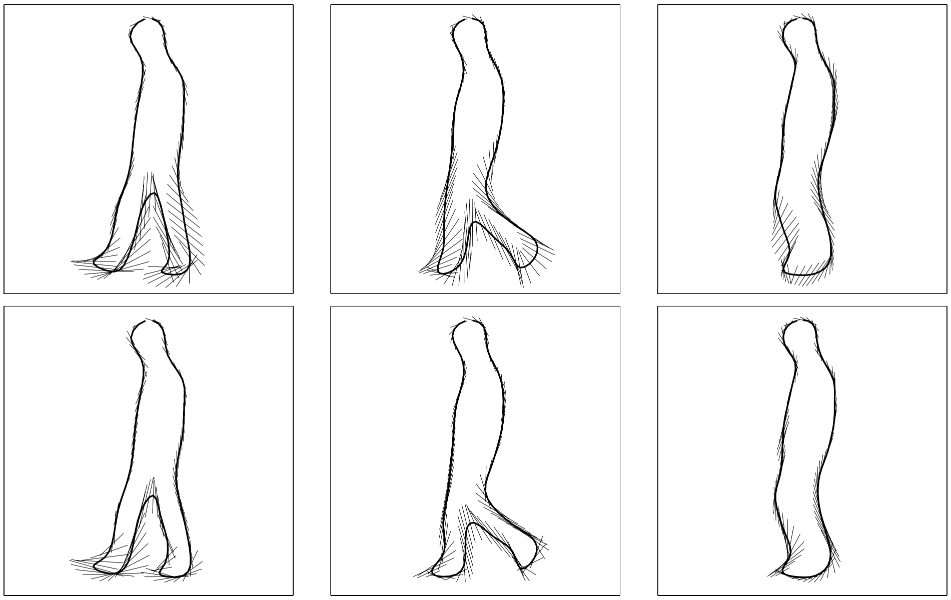


Figure 4.11: Shape variability modelled. Top: variability corresponding to the SVS components of a global shape model. Bottom: variability corresponding to the local shape models created from the SVS components.

much better (i.e., −4850 vs −9829 in the single model approach). Evaluating together shape and SVS models, in this particular application the multiple model proposal is the best option, as it has a bigger BIC value (i.e., closer to zero).

4.3 Tracking With Multiple Shape Models

Previous section has shown the benefits of using multiple models to represent the shape variability of an object. Now it is time to use them in a tracking application. First, it has to be formalised the interaction between the models in the evolution of the state of a process to be estimated. Extending the classical expressions of system and measurement processes in Kalman filters, the action of multiple models can be modelled including a discrete variable r_t identifying which model is active at each instant t. Formally, this is expressed as[4]

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}_{r_t}\mathbf{x}_{t-1} + \mathbf{w}_{r_t} \qquad \mathbf{w}_{r_t} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{r_t}) \\ \mathbf{y}_t &= \mathbf{H}_{r_t}\mathbf{x}_t + \mathbf{v}_{r_t} \qquad \mathbf{v}_{r_t} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{r_t}) \end{aligned} \tag{4.6}$$

This system is commonly referred as a *Conditional Dynamic Linear Model* or *Jump Markov Linear System* (JMLS) . It requires to specify the evolution of the model selection variable r_t, which is commonly done by means of a discrete time

[4] At this moment this expression is presented for the sake of clarity. Posteriorly, a different expression will be proposed, in order to better model the state dynamics.

Table 4.1: Quantitative results of the pedestrian modelisation. The closer to zero the BIC value (*bold* numbers), the better the model is judged.

Single Model Approach

BIC(Shape)	$\log p(\mathbf{q}_{1:N}, \Theta_{Shape})$	-$\frac{1}{2}M\log N$	Sum
	-1096	-1523	-2619
BIC(SVS)	$\log p(\mathbf{c}_{1:N}, \Theta_{SVS})$	-$\frac{1}{2}M\log N$	Sum
	-9082	-747	-9829
			$\mathbf{-12448}$

Multiple Model Approach

BIC(Shape)	$\log p(\mathbf{q}_{1:N}\vert\Theta_{Shape})$	-$\frac{1}{2}M\log N$	Sum
	-909	-5329	-6238
BIC(SVS)	$\log p(\mathbf{c}_{1:N}\vert\Theta_{SVS})$	-$\frac{1}{2}M\log N$	Sum
	-4660	-190	-4850
			$\mathbf{-11088}$

Markov chain with known transition probabilities. Basically, a JMLS can be seen as one linear system whose parameters $(\mathbf{A}_{r_t}, \mathbf{Q}_{r_t}, \mathbf{H}_{r_t}, \mathbf{R}_{r_t})$ evolve according to a finite state Markov chain r_t. The extension to non–linear and/or non–Gaussian systems is straightforward. In the following, this kind of systems are generically denoted as *Jump Markov Systems* (JMS).

The basic problem of estimating $\mathbf{x}_t$ in a JMS is that it is unknown the active model at each instant t (i.e., r_t is not observable). This brings to consider at each instant t all the feasible values of r_t, opening in that way a tree of multiple hypothesis. This has an obvious drawback: the computational demand to estimate $\mathbf{x}_t$ grows exponentially along time, making unfeasible an optimal estimation of $\mathbf{x}_t$ in practice. For that reason, *suboptimal* methods have been proposed to provide an operative solution to different practical problems.

Early contributions on this problem were strongly conditioned by the limited computational power available at that time. This forced to reduce the evaluation of the tree of feasible hypothesis to the minimal expression. Basically, given $\mathbf{x}_{t-1}$, at t the tree of hypothesis was spreaded and from the opened branches a unique estimation of $\mathbf{x}_t$ was derived. Thus, at each time step, only one hypothesis survived. Popular examples of these first approaches are the *Gaussian Pseudo Bayesian Filter* (GPBF) [1], and the *Interacting Multiple Model Filter* (IMM) [19, 84], which are restricted to linear systems. These proposals mainly differ in how the opened tree branches are processed to determine the final $\mathbf{x}_t$ estimation at instant t. A comparative study of these multiple–model algorithms and some of their extensions is provided in [91].

With the availability of more computational resources, it has been feasible to take into consideration a bigger depth in the tree of multiple hypothesis. The reason is clear; the more history is considered, the better the approximation of the optimal density is obtained. To avoid the exponential grow in the number of hypothesis considered, different methods have been proposed to consider from the full tree of hypothesis just a limited number of them. Proposed strategies are mostly based on two basic concepts: fusing tree branches which are very similar, and pruning the ones less sustained by the observations. Examples of these methods are given in [86, 81, 80].

The first operative implementations of PF opened a new perspective to solve this problem. In fact, a PF can be interpreted as a mechanism to propagate along time a constrained number of hypothesis on a given state $\mathbf{x}_t$. Each particle represents a hypothesis expanded along time, and its survival along time depends on the resampling procedure, which inherently implements a mechanism of pruning branches of a hypothesis tree. Thus, the use of PF in a multiple model system is very attractive, as allow to implement the expansion and pruning of hypothesis in a very simple way. In this chapter we propose a multimodel contour tracking derived from this perspective. In the following sections, first we present the procedure used to construct a JMS from the multiple models previously generated. Then, we propose an estimation algorithm based on the PS technique, adapted to deal with a JMS in a contour tracking application. Two versions of the algorithm are proposed, to be used in the following two situations:

- tracking a contour using a single shape model with multiple dynamical models;
- tracking a contour using multiple shape models, each one with its own dynamical model.

The first situation corresponds to the case in which a single *global* shape model is used, modelling its dynamical behaviour from the components of the GMM describing its SVS. The second one corresponds to the case in which a collection of *local* shape models are used. We evaluate the performance of both proposals experimentally, showing the benefits of using multiple shape models in contour tracking tasks.

4.3.1 Modelling the Interaction Between Models

In a multiple model tracking system, the model to be used at each instant (i.e., the value of r_t) may vary in every frame, and this behaviour has to be modelled. In most applications the transition between the different models is not arbitrary, but follows a given dynamic pattern. Thus, given a training sequence, the dynamics of the discrete variable r_t can be learned. A very simple way to model that consists in using a first order *State Transition Matrix* (STM) , which defines what is known as a Markov Chain Model. This is a discrete probability density function of the form

$$p(r_t|r_{t-1}) = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{k,1} & \cdots & \cdots & p_{k,k} \end{pmatrix} , \tag{4.7}$$

where $p_{j,k} \geq 0$ for all j, k, and $\sum_k p_{j,k} = 1$ for all j. Each value $p_{j,k}$ gives the probability of being active the j-th model at time t conditional on being active the k-th model at time $t-1$. These probabilities can be set by the following procedure.

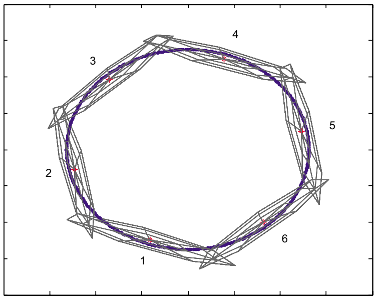
Given a training sequence $\mathbf{q}_{0:t}$ displaying one o more cycles of the dynamic behaviour to be modelled, each own of its components is projected onto the global shape space $\mathcal{L}(\mathbf{W}, \bar{\mathbf{q}})$ used in the multiple model construction. This leads to a sequence $\mathbf{c}_{0:t}$, where each one of its elements is labelled with the component g_t of the SVS model that *captures* it. As from each SVS component derives one of the multiple models learnt, the sequence of labels $g_{0:t}$ determines the sequence of models expected to be active along time. Using $g_{0:t}$, the STM can be constructed. First, elements $p_{i,j}$ are initially set to zero. Then, using each consecutive pair of labels (g_t, g_{t-1}) they are updated using the expression

$$p_{g_t,g_{t-1}} \quad = \quad p_{g_t,g_{t-1}} + 1 \ .$$

Once the whole sequence has been processed, the resultant values are normalised by

$$p_{i,j} \quad = \quad \frac{p_{i,j}}{\sum_{k=1}^{K} p_{i,k}} \ .$$

Figure 4.12 shows an example of this discrete description of dynamics.



	1	2	3	4	5	6
1	0.96	0	0	0	0	0.04
2	0.04	0.96	0	0	0	0
3	0	0.04	0.96	0	0	0
4	0	0	0.04	0.96	0	0
5	0	0	0	0.04	0.96	0
6	0	0	0	0	0.04	0.96

Figure 4.12: Dynamic learning process of a Markov Chain model. Left: a GMM models the SVS. Right: the STM computed from a training sequence, using the procedure described in [57]. It can be seen that the shapes in the training sequence progress along the SVS in clockwise order.

Provided that $g_{0:t}$ maintains one or more complete cycles of the expected periodic change between models, it can be used to compute the a priori probability of each model of being active. This corresponds just to the portion of the sequence in which each model is active. That is

$$p_i = \frac{\sum_{k=1}^{t} (g_k == i)}{t} \ . \tag{4.8}$$

Once the transitions between models are modelled, the next step is detailing the dynamical model inside each shape model $\mathcal{L}(\mathbf{W}^k, \bar{\mathbf{q}}^k)$. We propose to do that using AR(1) models constrained inside a Gaussian envelope. By using first order models, our objective is minimising the dimensionality of the state to be estimated, with the aim to favour the performance of PFs. To construct the AR(1) models, we project the training examples $\mathbf{q}_{0:N}$ onto the parameter space of their corresponding shape model, and then the distribution of this projection is approximated using a Gaussian model. Using the Gaussian parameters, a Constrained Brownian motion is established, which confines the evolution of the model parameters inside a hyperellipsoidal region.

Since the proposed multiple model system combines different shape spaces, which can be even of different dimensionality, it is also necessary to model how an state vector changes once a given model transition occurs. Since in general each model can represent a family of shapes completely different to the ones of the other models, so there will be no continuity within parameters in the different spaces, what we propose is modelling the more likely initial parameters of a model, once a given model transition occurs. This is approximated from the training sequence, by collecting the initial parameterisations of each model, for the different feasible model transitions. Being $\{\mathbf{c}_k^{i,j}\}_{k=1}^{N^{i,j}}$ the set of initial parameters of model j when a transition form the i-th model has occurred, we approximate its distribution by a Gaussian with parameters $\mathcal{N}(\mu^{i,j}, \mathbf{\Sigma}^{i,j})$. Thus, for a given active model r_t, the evolution of the estimated state $\mathbf{x}_t$ is finally defined by

$$\mathbf{x}_t \quad = \quad \begin{cases} \mathbf{A}_{r_t}\mathbf{x}_{t-1} + \mathbf{w}_{r_t} & \text{for } r_t = r_{t-1} \quad \text{with } \mathbf{w}_{r_t} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{r_t}) \\ \mu^{r_{t-1},r_t} + \mathbf{n}_{r_{t-1}-r_t} & \text{for } r_t \neq r_{t-1} \quad \text{with } \mathbf{n}_{r_{t-1},r_t} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^{r_{t-1},r_t}) \end{cases} \tag{4,9}$$

modifying slightly the general JMLS presented in Equation (4.6). Figure 4.13 sketches our proposal for modelling the dynamics of the proposed multiple model system.

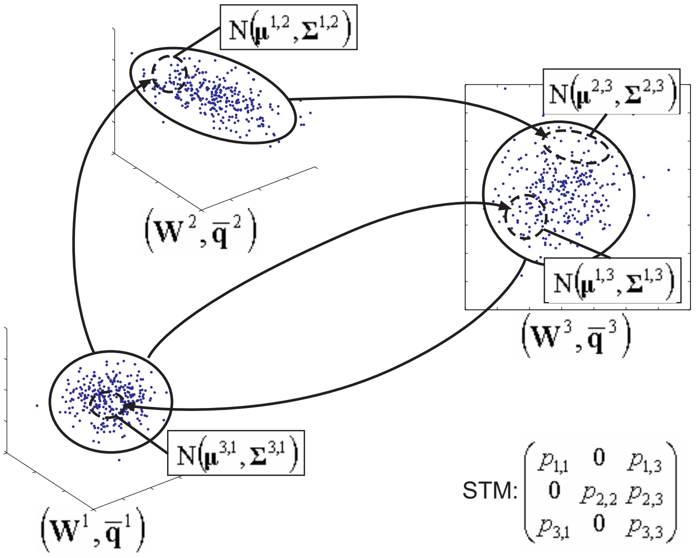

$N(\boldsymbol{\mu}^{1,2}, \mathbf{\Sigma}^{1,2})$
$N(\boldsymbol{\mu}^{2,3}, \mathbf{\Sigma}^{2,3})$
$(\mathbf{W}^2, \bar{\mathbf{q}}^2)$
$N(\boldsymbol{\mu}^{1,3}, \mathbf{\Sigma}^{1,3})$
$(\mathbf{W}^3, \bar{\mathbf{q}}^3)$
$N(\boldsymbol{\mu}^{3,1}, \mathbf{\Sigma}^{3,1})$
STM: $\begin{pmatrix} p_{1,1} & 0 & p_{1,3} \\ 0 & p_{2,2} & p_{2,3} \\ p_{3,1} & 0 & p_{3,3} \end{pmatrix}$
$(\mathbf{W}^1, \bar{\mathbf{q}}^1)$

Figure 4.13: Dynamical model attached to the proposed multiple model approach.

An Alternative Approach

The modelling methodology proposed in this chapter is based on a two–phased process: first generate a shape model, and then fit a dynamical model onto it. A different strategy could have been used to generate both models simultaneously. In [35] is proposed to learn the SVS of a given shape model by posing this problem as the parameterisation of a Hidden Markov Model (HMM) [92], given a set of training sequences. Like in the method previously proposed, the SVS is modelled by a GMM, but now the transitions between the different Gaussian components are modelled simultaneously using the Baum-Welch algorithm. We claim that our two–phased method has some advantages in practical terms. To generate the shape model just a set of significative examples has to be provided in order to model the desired shape variability, but no spatio–temporal coherence between examples is required. On the other hand, to train properly the HMM, several training sequences have to be aligned, showing exactly the same cyclical dynamic behaviour, which can be costly. This task of sequence alignment is also present in the proposed methodology to train the dynamical model, but as long as the shape variability has been already modelled, just the alignment of a few sequences is usually needed.

4.4 PFs in a Multiple Model System

The application of PFs in problems concerning JMS is direct. Just a discrete variable $r_t^{(i)}$ has to be attached to each particle $\mathbf{x}_{0:t}^{(i)}$ modelling the target distribution, which informs of the model assumed active at a given instant t. Then the SISR procedure (Algorithm 3 in Chapter 2) is applied, using commonly the following mechanism to propagate particles:

- first the current active model $r_{t-1}^{(i)}$ is propagated sampling the Markov chain model in (4.7)
$$r_t^{(i)} \sim p(r_t|r_{t-1}^{(i)}) \ ;$$
- then the particle $\mathbf{x}_{0:t-1}^{(i)}$ is propagated according to the $r_t^{(i)}$ value, using the dynamics in (4.9), which is concisely expressed as
$$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)} r_t^{(i)}) \ .$$

To reduce the number of particles required to achieve a desired performance, several variance reduction techniques can also be applied on this problem. For instance, when the multiple models involved are linear, the Rao–Blackwellised Particle Filter (RBPF) can be applied to estimate the r_t distribution by means of particles, estimating the rest of the state using a Kalman Filter [27, 42]. If models involved are non–linear, the work in [7] proposes different techniques to also increase performance efficiency. We propose to use the PS technique to manage the multiple model system in a contour tracking application, because, as shown in the previous chapter, it overcomes the performance of standard PF implementations, for an equivalent amount of particles. We check only the performance of this algorithm, since our goal in this chapter is not identifying which estimation strategy takes more profit on using multiple models, but displaying the benefits of our multiple model approach with respect to the approach of using a single shape model with a SVS constraint, as is done in [57, 35]. Hence, we evaluate the tracking performance achieved using:

- a single shape model, constrained in a SVS modelled by a GMM,
- an heterogeneous collection of simpler shape models, each one constrained inside a Gaussian envelop.

The difference between both approaches is that in the former modelisation, the estimation process consists in propagating particles in a given shape space through some kind of *wormholes*[5], while in the later particles propagate themselves through completely different spaces, which can be even of different dimensionality (Figure 4.14).

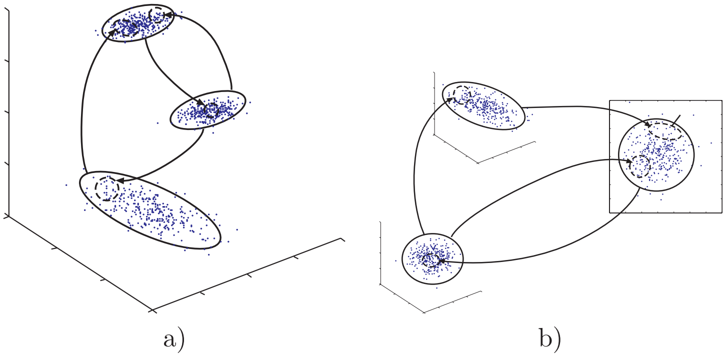


Figure 4.14: State estimation inside a *wormholed* space (left) and in a multi-spatial system (right).

4.4.1 A Case–Study: Tracking the Outline of a Pedestrian

The tracking problem analysed in this section is the estimation of the silhouette of a pedestrian, whose variation along time is due to the translation and the local deformations of its outline. The two modelisation strategies compared require to estimate, at each frame, the value of a discrete variable r that identifies:

- the active dynamical model (derived from the SVS component expected to be active), if a single shape model is used;
- the active shape model, with its associated dynamical model, if a multiple shape model is used;

Together with the value of r, for each frame we must estimate the shape parameters $\mathbf{x}^D = [\mathbf{c}^r]$ too, as well as the translation $\mathbf{x}^R = [t_x t_y]$. This task is solved using the PS algorithm (see Algorithm 10 in Chapter 3), as its performance overcomes the one of *classical* PFs.

Given the state $\mathbf{x}_t = [\mathbf{x}_t^R \, \mathbf{x}_t^D \; r_t]$, we propose to implement the weighted resampling step of the PS in terms of the state partition $[\mathbf{x}_t^R \; r_t]^T$. The reason is the following. The silhouette of a walking pedestrian presents zones with a high variability (the

[5] Term used in [57] to refer to jumps between different space regions.

outline corresponding to the legs), as well as zones which scarcely vary along time (the head and back contour regions). These practically rigid zones make possible to preevaluate the likelihood of the state part $[\mathbf{x}_t^R \ r_t]^T$ independently of the value of $\mathbf{x}_t^D$, and this is used to implement the weighted resampling step in the PS algorithm. In the two modelisation techniques compared, the methodology followed is the same. The r_t value denotes the model assumed active at instant t, but also determines a Gaussian region of valid parameterisations in a given shape space. This region of valid parameters has implicitly associated a set of valid shapes, whose variance allows us to determine the contour zones which suffer from less deformation (see Section 3.3.1). Thus, for each sample partition $[\mathbf{x}_t^{R(i)} \ r_t^{(i)}]^T$, the *rigid* contour zone associated to $r_t^{(i)}$ is used in the reweighted sampling step, using the following evaluation function

$$g([\mathbf{x}_t^R \ r_t]^T) \quad = \quad \left(\prod_{i=1}^{N_{r_t}} \left(q_{01} + \frac{q_{11}}{\sqrt{2\pi}\sigma\lambda} \exp\left(-\frac{\nu_{t_i}^2}{2\sigma^2} \right) \right) \right)^{1/N_{r_t}} \tag{4.10}$$

where ν_{t_i} is the distance to the closest edge in the image of the contour corresponding to $[r_t \ \mathbf{x}_t^R]^T$, for the i-th measurement line. As the number of measurement lines N_{r_t} for each rigid contour model can vary, the evaluation function consists in the geometric mean of the likelihood computed in each measurement line. In that way, models with a bigger rigid contour region are not penalised. Figure 4.15 shows for the two compared modelisations, the contour zones where observations are extracted and used in the weighted resampling step.

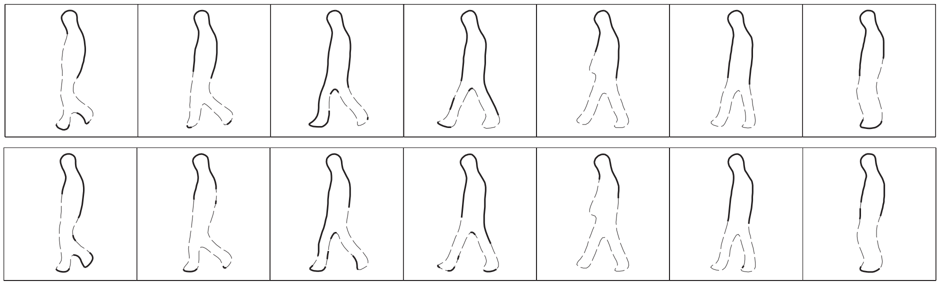
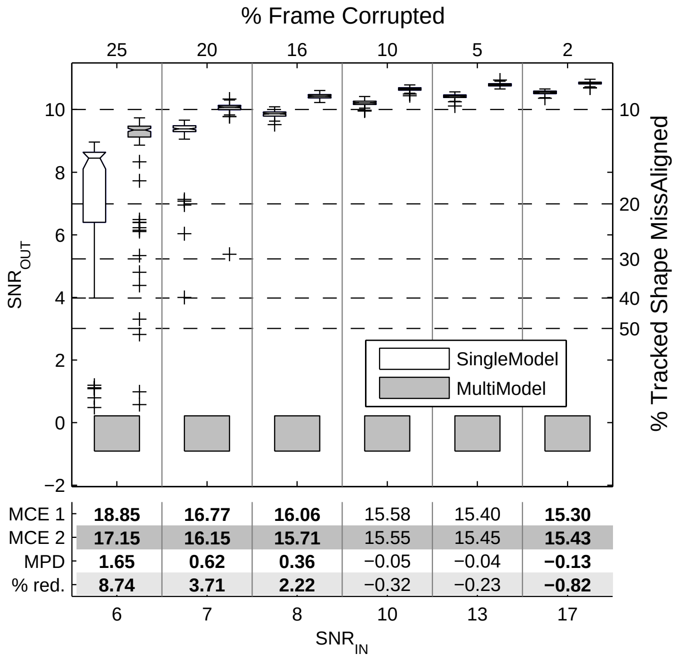


Figure 4.15: Almost rigid contour zones for the different states in a single shape model approach (top), and the different models in a multiple shape model approach (bottom).

The tracking performance achieved for the two considered shape modelisations is shown in Figure 4.16. As in the previous experiments in this thesis, the proposals are evaluated on distorted versions of a given *ground truth* sequence, analysing the results of 100 tracking performances, for each noise situation considered. Unlike in the experiments of preceding chapters, the ground truth sequence has been obtained from the manual segmentation of a real sequence. In that way, no one of the compared models is favoured. The PS algorithms implemented manage an amount of particles equivalent to 1000 in a SISR algorithm, using two thirds of the particles in the weighting resampling step, and the remaining one third in the likelihood evaluation of the complete particle state.



SNR$_{IN}$	6	7	8	10	13	17
MCE 1	**18.85**	**16.77**	**16.06**	15.58	15.40	**15.30**
MCE 2	**17.15**	**16.15**	**15.71**	15.55	15.45	**15.43**
MPD	**1.65**	**0.62**	**0.36**	−0.05	−0.04	**−0.13**
% red.	**8.74**	**3.71**	**2.22**	−0.32	−0.23	**−0.82**

Figure 4.16: Pedestrian tracking performance. PS using a single shape model and a multiple shape model approach.

Results show the significant improvement of the tracking performance provided by our multiple shape model strategy, in all the noise situations evaluated. The improvement achieved is more significant, the more the noise distorts frames. With the use of multiple shape models, the dimensionality of $\mathbf{x}^D$ decreases with respect to the single model approach, resulting in a more *dense* inspection of the corresponding shape space, for a given number of particles. An additional benefit of this dimensionality reduction, is that the computational cost of the algorithm implementation is reduced, as it depends linearly on the dimension of the estimated state.

4.5 Conclusions

In this chapter we have proposed an strategy to improve the performance of particle filters, based on modelling the target to be tracked by means of multiple low–dimensional shape models, instead of the single model approach commonly used. To fulfil this objective, we have done different proposals concerning the following topics:

- the unsupervised modelisation of the subspace of valid parameterisations of a given shape model;
- the generation of multiple shape models, from the SVS of a given shape model;
- The use of multiple models in a particle–based estimation algorithm.

Concerning the first topic, we have modelled the SVS of a given shape model using a GMM, bounding the region in the space of model parameterisations where training shapes project. We have proposed a novel greedy algorithm to parameterise the GMM, based on embedding the EM algorithm inside the procedure of classical divisive hierarchical clustering algorithms. The proposed method avoids an exhaustive search in the space of parameters of the GMM, incrementing iteratively the complexity of an initial GMM until the sample dataset is properly modelled. Results achieved are comparable to the ones obtained by an exhaustive search approach. Our proposal is very appealing to be used in practical applications, as no parameters has to be set by the user. The method displays also a qualitatively good performance in modelling complex dataset distributions typically used in the literature to check this sort of algorithms.

With regard to the second topic, we contribute with a novel methodology to generate multiple shape models, from the components of the GMM bounding the SVS of a single shape model. The proposed method replaces the initial shape model by a collection of models of lower dimensionality, which is the desired characteristic in order to improve the performance of contour trackers based on PFs. The validity of the proposed multiple model approach has been tested in the context of modelling the silhouette of a walking pedestrian. In terms of the BIC, it has been constated that our proposal provides a better shape modelisation than the one obtained using a single model approach.

Finally, we have presented a novel approach to use the multiple models generated in a contour tracking application. First, we have formalised the relationship between the different shape models using a JMS, establishing their parameters from training data. Then, we have proposed a novel contour tracking algorithm to manage the multiple models, and we have tested its performance in the problem of tracking the silhouette of a walking pedestrian. Results conclude that using the multiple model approach, we overcome the performance achieved using the traditional single shape model approach.

Chapter 5

Monocular Model–Based Vehicle 3D Tracking System

Our work detailed in preceding chapters has focused on the study of different tracking algorithms, aimed at improving the estimation of the contour of targets along time. In the study done, algorithms were manually initialised, and evaluated on sequences displaying the target of interest partially or totally in all video frames. In this chapter a more general goal is pursued: the design and implementation of a complete *target tracking system*. In addition to the tracking of targets along frames, a tracking system involves the automatic detection of target in images, providing their initial state to a tracking algorithm, and control the situations in which a tracked target is no longer of interest.

Designing and implementing a tracking system means developing this system for a given specific task. The one analysed in this chapter is the development of a vision–based on–board advanced driver assistance system devoted to detect and track other vehicles on the road ahead. Our main motivation to focus on this problem is that its resolution is a *classical* demand of intelligent transportation systems, as the main threats a driver faces are from other vehicles in the driving environment. Most applications addressed by the automobile industry (adaptive cruise control, autonomous stop & go driving, lane change assistance, etc.) rely on an accurate perception of the own vehicle surroundings [38]. To solve these tasks, the knowledge about other vehicles around is of main relevance. In essence, what these applications demand is estimating the relative 3D position and velocity of the observed vehicles, in order to predict their future location and, for instance, warn in advance about unsafe manoeuvres.

The perception of a vehicle surroundings has been investigated for automation with active sensors for several decades. Hence, the application of radar and lidar systems to the current application seems natural. However, the richness of the information that can provide a camera (texture, colour, intensity) in addition to high resolution and other aspects as low cost, low consumption, non-intrusive nature, etc., make it the most suitable in the sensing process, either working alone or in combination with active sensors. After all, vision is by far the most relevant sense in human driving.

This chapter proposes a vehicle tracking system based on monocular vision. The image acquisition is based on a monochrome camera installed on a *host* vehicle[1], mounted inside the cabin near the rear–view mirror, attached to the windshield screen and forward facing. The camera uses a 1/3-inch sensor, has a (approximately) 7 mm optics, an provides frames of 640×480 pixels. Our goal is detecting and tracking mid range and far distance vehicles (up to 70 meters), as long as they are entirely observed in images. To fulfil this objective, we propose a modular system composed of a vehicle detector, an estimator of the initial 3D state of detected vehicles, and a vehicle tracker. Figure 5.1 sketches the three main modules of the proposed system. Each module has been designed isolatedly from the rest, using a shared memory technique to establish communications among them. A supervising process controls their execution and coordinates their cooperation. An alternative to our proposed modular system could be the proposal in [128]. In this case, the tasks of the different modules are solved simultaneously within a Bayesian framework, which integrates in a single expression all the a priori system knowledge available (i.e., the vehicle appearance model, scene geometry, the assumed vehicle dynamics, etc.). However, although this formulation is very elegant, it lacks of scalability, and does not allow to check how each specific required task is being performed separately from the rest. This obscures the analysis of the system performance. A modular approach has clear advantages with respect to these points, which are very relevant to develop functional systems.

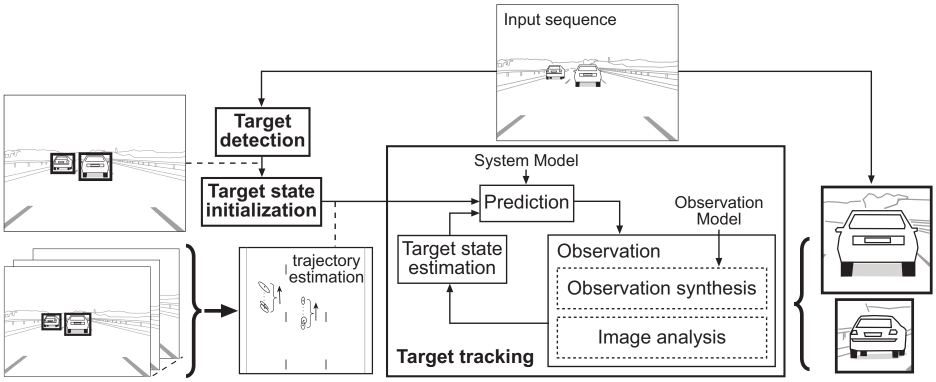


Figure 5.1: Vehicle tracking system proposed.

In a vehicle tracking system, the first task to be solved is detecting the presence of vehicles on acquired frames. A vehicle detection module is required able to carry out this task for different sorts of vehicles[2] (cars, vans, trucks, busses, etc.), from their frontal and rear view, at distances up to 70 meters, and under a wide range of illumination conditions (different daytime and weather conditions, and artificially illuminated scenarios). To generate such a detector, we propose in Section 5.1 the use of an state of the art machine learning technique (Boosting) to generate a model to robustly distinguish vehicles from other elements in the sequences. Given a big

[1] With *host* vehicle we denote the vehicle hosting the camera.

[2] Currently, motorbikes and similar are excluded.

amount of vehicle and non–vehicle examples, the Adaboost algorithm selects a subset of image features well–suited to discriminate between both training classes, and at the same time parameterises a classifier tuned for this specific application. Using this classifier, we propose to scan acquired frames with the aim to detect the observed vehicles.

Once vehicles are detected, the next step is determining which is their state (3D location and velocity), in order to initialise a tracking algorithm. This is the task of the vehicle 3D state initialisation module in Section 5.2, which requires to estimate the 3D road location corresponding to vehicle detections, and track them for a short period in order to identify their dynamics. As the camera system used is monocular, determine the 3D location of detected vehicles is an ill-posed problem, which in general can not be solved. However, for the specific system analysed, realistic assumptions about the driving environment can be done, which make this problem solvable. Specifically, if the captured road conforms to a flat surface, a direct relation can be established between image coordinates and the 3D location of objects on the road, as both (the image and the road) are two planes and, therefore, can be related by an homography. Once the 3D location of detections is computed, it is analysed their spatio–temporal behaviour in order to estimate the 3D velocity of vehicles. This task is not only useful to complete the initial state of a vehicle tracking algorithm, but also to filter out false detections (i.e., *False Positives* (FPs)) that the vehicle detector can generate. Indeed, depending on the camera viewpoint, sometimes non–vehicle regions match the appearance of some of the sorts of vehicles modelled. These regions mislead the designed detector just temporarily, because due to the movement of the vehicle holding the camera, the camera viewpoint changes, and with that the appearance of such regions, which are then correctly identified as non–vehicles. The process to estimate the 3D velocity of detected vehicles will identify false detections due to their lack of spatio–temporal coherence.

Having the initial state of a vehicle estimated, the goal is updating it efficiently in the successive frames by means of an estimation procedure. This task is solved by the vehicle tracking module detailed in Section 5.3, which has been implemented by means of a multitarget UKF, due to the non–linear relation between the vehicle 3D states and their observations, and the type of noise disturbing the measurement process. Finally, the coordination of the different modules of the system (i.e., how they are sequenced and share information), is briefly detailed in Section 5.4.

5.1 Vehicle Detection

The detection of vehicles is the most challenging task of the proposed system. On the one hand, vehicles conform a very heterogeneous class of objects, showing a wide inter–class variability with respect to their appearance and dimensions (see Figure 5.2). On the other, they have to be detected in an uncontrolled environment with a high variability too, and in a restricted processing time. This problem has been classically addressed using heuristic criteria, based on certain expectations about the vehicle appearance. Commonly, observed vehicles present

- vertical axis symmetry;

- dark wheels;
- left and right boundaries;
- dark areas beneath (their own shadows).

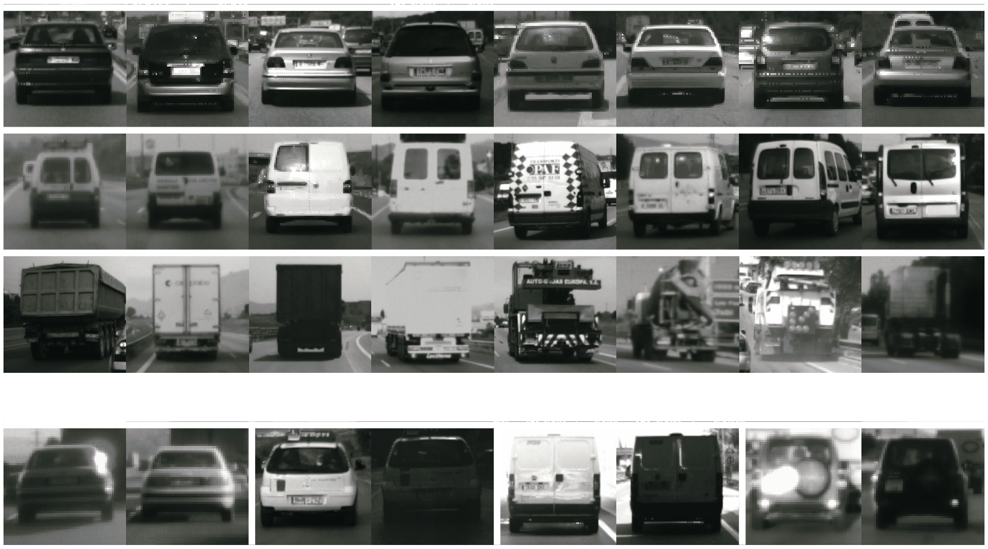


Figure 5.2: Vehicle training set examples. Top: heterogeneity of the objects to be detected, just from their rear–view. Bottom: Pairs of the same vehicle, acquired under different illumination conditions.

These and other properties have been considered by many authors, to propose the use of different visual cues to quickly hypothesise the presence of vehicles on images. Examples can be found based on shadows and lateral edges [82], symmetry [75], a combination of shadows, symmetry and edge information [24, 58], considering the movement perceived between frames [13, 70], texture descriptors [65, 112], etc. These methods are rather naïf to provide a reliable detection system, and commonly generate many false detections. However, they can help to preselect with an small computational cost, the image regions where to apply a more sofisticated detector. Other strategies can also be used with this same objective, like for instance, the presegmentation of paved regions [51, 4], the use of high level information extracted from images (for instance, use the detected road lane markings to delimit the detection region [106, 58, 125]), or the use of other available sensors (for instance, radar [49, 126, 3]).

Main proposals to verify the presence of vehicles in given image regions are posed from a classification point of view. Given an image region, a set of descriptors is computed and evaluated by a classification function, which discerns if the region appearance corresponds to a vehicle or not. Many different features have been proposed in the bibliography, with the aim to provide a robust vehicle representation: texture descriptors [67], edge orientation histograms [48], Gabor filters [108], Legendre moments [126], Haar filters [111], etc. Using a training set composed of vehicle and non–vehicle examples, their feature representation is processed by a machine learning algorithm to construct a vehicle classifier. The algorithms most commonly applied to this task are Support Vector Machines (SVMs) [111, 108, 126, 67] and Neural Networks (NNs) [65, 48, 126].

The recent review in on–road vehicle detection in [110] claims that an important drawback of some proposed vehicle classifiers is that they are based on a *generic* feature space defined a priori, which in general includes features redundant or even irrelevant for the vehicle detection problem. Ideally, the vehicle detection should be based on features that have great separability power to distinguish vehicles from other elements on their environment. In that way, learned classifiers would generalise better, requiring the computation of less features to classify image regions. The search for these discriminant features is the object of study of feature selection techniques [53]. In [109] is proposed the use of genetic algorithms to determine an optimised set of Gabor filters to discern vehicles from no–vehicles. In this thesis we adapt the approach based on the Adaboost algorithm in [117][3] to our problem, which solves simultaneously the selection of discriminant features and the construction of a vehicle classifier.

Summarising, on–road vehicle detection is commonly solved in practice in two steps: a fast preselection of image regions where to look for vehicles, and the refinement of these regions by means of a classifier. In systems based purely on vision sensors, the preselection step is needed to fulfil the response time constraints imposed by final applications. In the work developed along the thesis, we have checked some preselection methods in the literature [82, 36, 13]. Due to response time constraints, methods checked consisted basically on the application of thresholding operations on low level image features, in order to verify the presence of given image structures (shadows, edges, etc.). The performance achieved by this methods in many situations was significantly poor. For acquisition conditions deviated from the most common ones, the thresholds for image processing algorithms were no longer valid, as well as the ones used to verify the presence of relevant image features, which in some cases were no longer observable in frames. Due to this poor performance, vehicles that our vehicle classifier was able to detect, were in practice missdetected. Hence, we decided, for the moment, to desist from using a preselection process. This prevents the application of the proposed detector in a realtime scenario, but allows to measure the real detection power of the proposed vehicle classifier. We claim that before establishing a fast criterion to roughly predetect vehicles, we first need to assure that our classifier reaches a desired performance. Once we achieve that, we could take advantage of the features used by the vehicle classifier to construct a fast vehicle predetector assuring that it does not provoke missdetections, which is the most critical error of a vehicle detection system.

5.1.1 Vehicle Detection Using Adaboost

Due to the proven effectivity of the Adaboost algorithm in generating robust classifiers [117], we propose to use this technique to construct a vehicle classifier that will be the

[3] The approach was originally applied to face detection.

basis of the vehicle detection module of our system. The Adaboost is a supervised algorithm, which trains a classifier from a training set $\mathbf{t}_{1:N_r} = \{(\mathbf{h}_i, l_i)\}_{i=1}^{N_r}$ of N_r examples, where $\mathbf{h}_i = [f_j^i]_{j=1}^{N}$ is a vector of features describing the i-th example, and l_i is a boolean flag indicating if this example is positive (i.e. it is a vehicle) or not. This algorithm selects $N_f \ll N$ features $\mathbf{f}$ from $\mathbf{h}$, and estimates a *weak* classifier $r_i()$ for each one of them, such that when properly combined correctly classify the training examples. Classifiers $r_i()$ are branded as *weak* to reflect that commonly their particular classification accuracy is just slightly over 50% of the training set. We have used the confidence–rated version of the Adaboost algorithm [100], in which $r_i()$ is a decision stump on f_i that returns a positive (v_i^+) or negative (v_i^-) value according to its classification decision. That is,

$$r_i(f_i) = \begin{cases} v_i^+ & \text{if } f_i \leq (\text{or} \geq) \textit{ threshold} \ , \\ v_i^- & \text{otherwise} \ . \end{cases}$$

The Adaboost combines these learned weak rules to construct a *strong* classifier

$$s(\mathbf{f}) = \sum_{i=1}^{N_f} r_i(f_i) \ , \tag{5.1}$$

which, when applied to the features $\mathbf{f}$ of an image region, returns a value whose sign provides the classification decision on this region[4], and whose absolute value indicates the confidence on this decision.

Training examples are characterised by an over-complete set of Haar-like[5] features. We use this type of features because they can be computed very efficiently on image regions, with a constant computational cost whichever the size of the region [117]. With the aim of detecting vehicles, this is a very interesting characteristic, because vehicles are imaged in a wide range of scales (in our particular case, from 24×18 up to 334×278 pixels), and Haar-like features do not demand an explicit size normalisation of image regions. Moreover, these features can be computed to be invariant to monotonical changes on the intensity of regions, remaining thus also useful even in low contrast lighting conditions. Using these features, each example is described by a vector of 176214 Haar filter responses.

The procedure followed to learn the vehicle classifier has been the following. First, an initial vehicle classifier is trained to detect frontal/rear views of all the kinds of vehicles considered (cars, vans and trucks). The training process needs examples and counter-examples of normalised size and aspect ratio. However, the aspect ratio of cars, vans and trucks is different, thus we have just cropped out the upper par of many trucks and some vans (Figure 5.3 a)). In other words, to detect trucks and vans only their lower part is used. The learned classifier has been then applied to a testing sequence, showing a good behaviour on detecting vehicles, but a bad behaviour on discarding non–vehicle regions. This is due to the fact that, initially, it is very difficult to provide a representative set of negative examples, as the non–vehicle class is very heterogeneous. Collecting the false positive detections generated by this first

[4] A positive value means that the region is classified as vehicle.

[5] The responses of filters proposed in [117] and their absolute value are used.

classifier, a new set of negative examples is generated, from which to learn a second classifier to discard these regions. Following the proposal in in [117], this procedure is applied iteratively, until the application in cascade of the learned classifiers results in a desired detection performance. The use of a Cascade of Classifiers (COC) not only improves the detection accuracy, but also reduces the classification time: once a region is classified as non–vehicle, it has no longer to be evaluated in the rest of the COC layers. Figure 5.3 sketches the procedure to learn a COC, and how it is used to classify an image region. Following this process, an initial COC of 7 layers has been obtained. In overall, 1830 positive examples and 11820 negative examples have been processed to learn the COC, extracted from sequences acquired in high speed roads, under sunny and cloudy midday and evening conditions. Due to the symmetry of vehicles, for each example its reflection along the Y axis has been also considered, doubling in that way the amount of training examples “ad-hoc”.

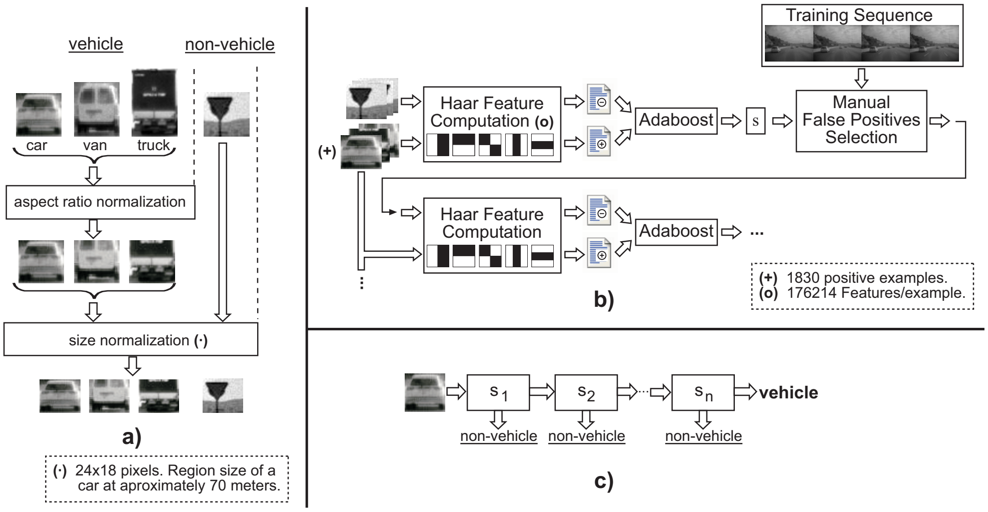

vehicle
non-vehicle
car
van
truck
aspect ratio normalization
size normalization (·)
a)
(·) 24x18 pixels. Region size of a car at aproximately 70 meters.
Training Sequence
(+)
Haar Feature Computation (o)
Adaboost
s
Manual False Positives Selection
Haar Feature Computation
Adaboost
...
b)
(+) 1830 positive examples.
(o) 176214 Features/example.
s1
s2
sn
vehicle
non-vehicle
non-vehicle
non-vehicle
c)

Figure 5.3: COC learning process. a) Training set normalisation. b) Process to construct the COC. c) Evaluation of the COC. True positives should be processed by all the COC layers.

Once we have a COC, the following step is applying it on images to detect vehicles. Next section details the systematic scanning procedure that we propose to perform this task.

5.1.2 Image Scanning Procedure

The process of scanning frames to detect vehicles requires establishing different image subregions where to apply the vehicle classifier. To properly perform this task, inspecting just zones of the image where it is likely to find vehicles, it is first necessary to detail how the 3D world projects on acquired frames. That is, modelling the camera acquisition system.

Camera Model

As previously remarked, the acquisition system is composed by a single camera installed near the rear–view mirror of the host vehicle, facing the road ahead with an slight inclination. To model how this camera relates with the 3D world, a host coordinate system is defined relative to the camera position, placed externally on the road that sustains the host vehicle (Figure 5.4). Using this reference frame, the point of view from where images are captured (i.e., the camera extrinsic parameters) is completely defined by the camera origin $\mathbf{o}^{Cam} = [0, h, 0]^T$ and the pitch angle θ describing its inclination with respect to the road surface.

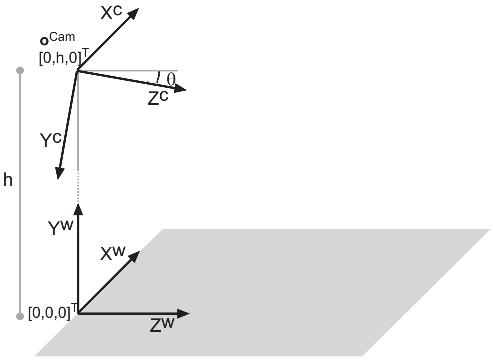


Figure 5.4: Camera coordinate system relative to the host coordinate system.

The projection of the 3D world onto 2D images is modelled as a pin-hole camera with zero-skew [55]. This requires identifying the effective camera focal length on the image X and Y direction (f^x, f^y) an its center of projection (x^0, y^0) (i.e., the camera intrinsic parameters). This has been done calibrating the camera with the software provided in [21]. Using homogeneous coordinates, the image projection of a 3D coordinate $[x\ y\ z]^T$ on images is defined by

$$s \begin{bmatrix} x^r \\ y^r \\ 1 \end{bmatrix} = \mathbf{A}\,\mathbf{S}\,\mathbf{R}\,\mathbf{T} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} , \tag{5.2}$$

where s is a scale factor, $\mathbf{A}$ the camera intrinsic matrix, $\mathbf{S}$ a reflection matrix to model the different sign of the Y axis of the camera reference system and $(\mathbf{R}, \mathbf{T})$ specify the rotation and translation that locates the camera coordinate system with respect to the host coordinate system. For the situation in Figure 5.4, these parameters are

$$
\begin{aligned}
\mathbf{A} &= \begin{bmatrix} f^x & 0 & x^0 \\ 0 & f^y & y^0 \\ 0 & 0 & 1 \end{bmatrix} , \\
\mathbf{S} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} , \\
\mathbf{R} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} , \\
\mathbf{T} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -h \\ 0 & 0 & 1 & 0 \end{bmatrix} .
\end{aligned}
$$

Scanning Process

The scanning process determines the image regions that are evaluated by the vehicle detector, from determining the rectangular image zones where vehicles located at different points on the road ahead project. These road points are defined in terms of a regular grid on the road surface, considering points up to 70 meters away from the camera (see figure 5.5). Given a grid point $[x\ 0\ z]^T$, the image coordinates $[x^r\ y^r]^T$ where it projects are determined from Equation (5.2) as

$$
x^r = x^0 + \frac{f^x\, x}{z\, \cos(\theta) - h \sin(\theta)}, \tag{5.3}
$$

$$
y^r = y^0 + \frac{f^y\, (h \cos(\theta) + z \sin(\theta))}{z\, \cos(\theta) - h \sin(\theta)}. \tag{5.4}
$$

These image coordinates determine the bottom–left corner of the image region analysed by the detector. The width and height of this region are determined from considering the presence of vehicles of different possible sizes at the road grid point. Obviously, if more than one grid point projects onto the same image pixel, only one of them is taken into account.

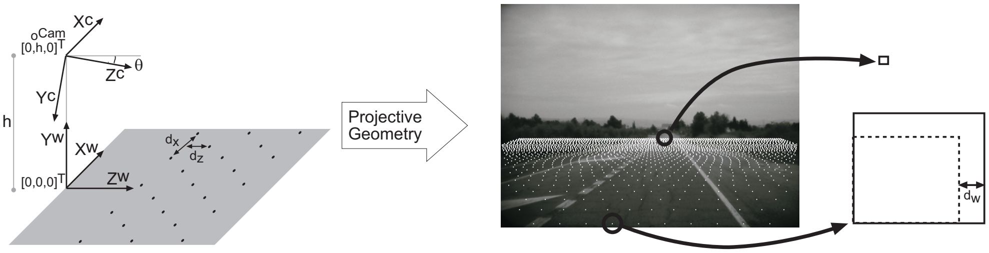


Figure 5.5: Frame Scanning Process. For each inspected road point, rectangular regions of different widths and aspect ratios are evaluated.

A problem with this scanning process is that the position of the camera with respect to the host coordinate system can vary significantly in every frame. Indeed, the effect of the suspension system of the vehicle alters the camera extrinsic parameters (see Figure 5.6). Thus, a proper scanning of frames depends on using the correct camera extrinsic parameters. We have evaluated how the projection of points from the road surface vary due to changes in the extrinsic parameters, and this has revealed that the parameter with a more significant impact on that is θ. Ignoring variations in $\mathbf{o}^{Cam}$ provokes a negligible error. However, estimating θ from acquired frames is a complex task in a monocular acquisition system. In [12] is proposed to ascertain θ by matching distant image structures observed on successive acquired frames, but in the experimental evaluation of this technique that we have done, the method shows a very poor performance in many situations (driving under bridges, in urban areas, etc.). In [107] a robust method is proposed to estimate the vehicle ego-motion from the optical flow observed in successive frames, which provides consequently information on the θ. Future work is planned to incorporate an ego–motion estimator to the current system, to assist the different modules proposed. For the moment, the problem of estimating θ from a single image has been obviated, as in a real system θ may be determined from additional sensors on the vehicle [32, 33]. In the experimental evaluations of the system, we will use (when available) a value of θ provided by ground truth data [6]; otherwise we will use its most likely value (i.e., the one corresponding to the neutral position of the car suspension system).

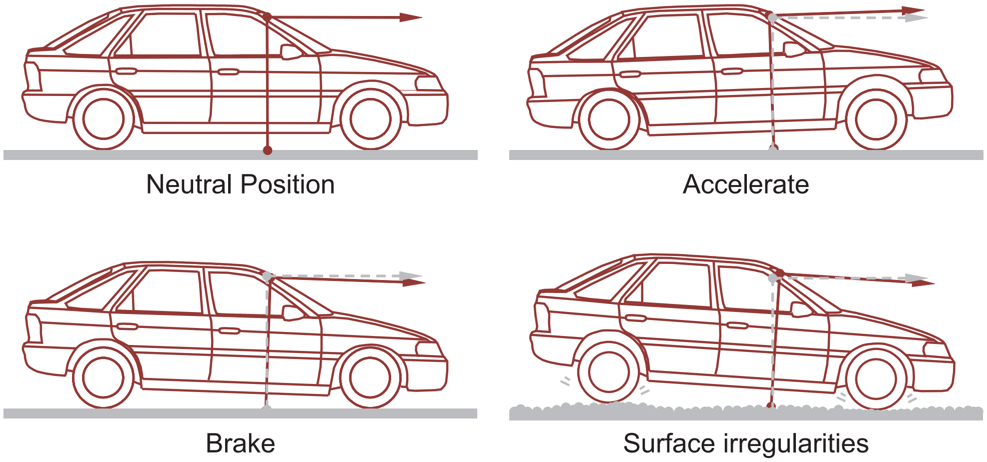


Figure 5.6: Extrinsic camera parameters variation.

Figure 5.7 exemplifies the application of the COC using the described scanning process, where the refinement of the classification decision at different levels of the cascade can be observed. Once a frame has been scanned, a list of image regions that may contain a vehicle is obtained.

[6] Given a frame, we obtain the current value of θ by manually annotating the position of the horizon. The y coordinate of the horizon (denoted as y^h) is related with θ by $\theta = \tan^{-1}\frac{(y^h - y^0)}{f^y}$.

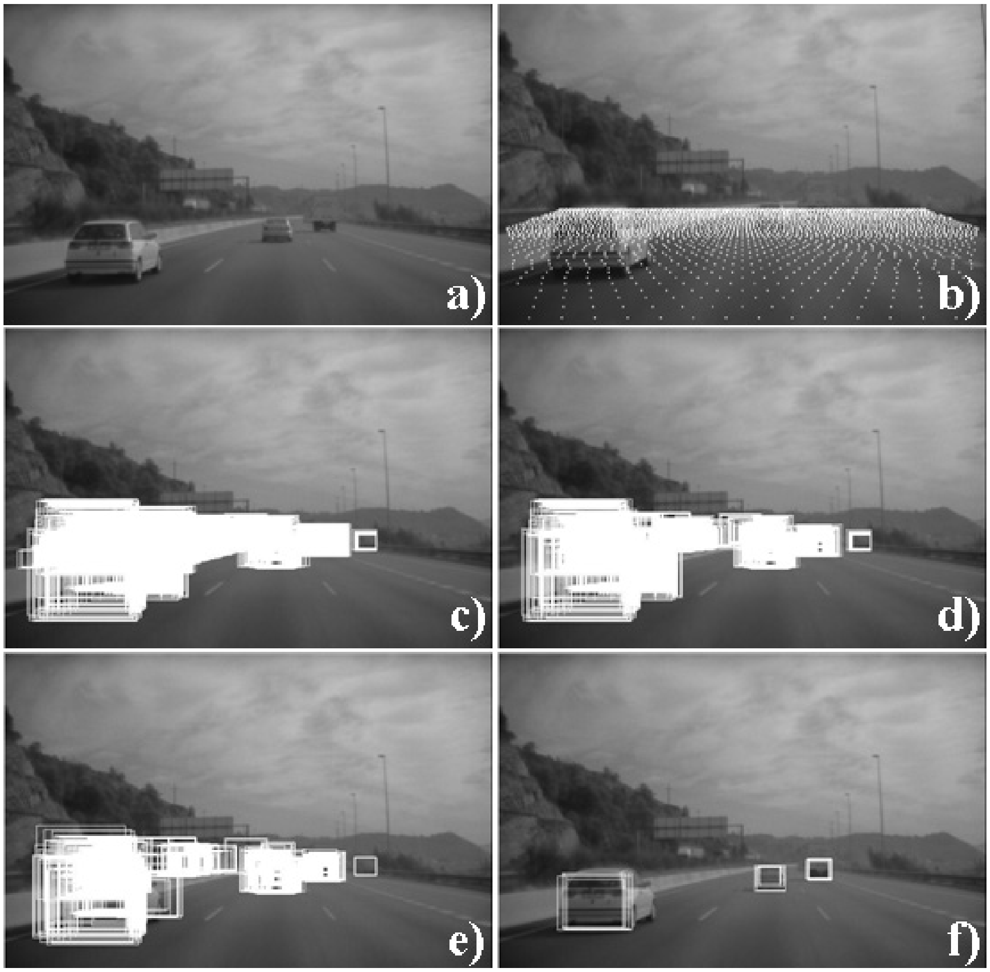


Figure 5.7: Vehicle detection example. a) Original image. b) Pixels used as bottom–left corner for candidate image regions where to look for vehicles. c)-f) Image regions that are like to be the bounding–box of the rear/front part of vehicles: c) shows the surviving regions after the first layer of the cascade (rule $s_1()$) and f) shows the regions surviving after the last layer (rule $s_N()$).

Clustering Detections

One same vehicle is usually detected in several neighbouring overlapping regions (see the final detections in Figure 5.7–f). This is undesirable, since the detector should provide a single precise detection for each vehicle. Thus, some postprocessing is necessary to fuse the detections corresponding to it. The presence of multiple detections is due to the fact that the classifier $s()$ of Equation (5.1) is sensitive to vehicles even if the scanning region is not perfectly adjusted to the location of vehicles on images. Figure 5.8 exemplifies that, detailing the response of $s()$ on subregions scanned ± 20 pixels from the optimal vehicle location (white rectangles). Notice that, in general, the higher detector responses are, the closer the subregion evaluated is to the vehicle optimal location. However, the *detection surface* computed has not a unique maximum, which difficults determining the best vehicle image location. Different strategies have been proposed in the literature to solve this problem. In [117], regions are partitioned into disjoint non–overlapping groups and each group gives a single detection located at the centroid of its associated regions. Differently, [2] proposes an strategy founded on the idea of non–maximum suppression. Provided that each detection has a confidence value, this strategy is based on identifying the one of highest confidence. This region is registered as output, and detections in a predefined neighbourhood around it are deleted. Then this procedure is repeated for the remaining regions, until all detections are either considered as output or are suppressed.

We have evaluated both strategies in the current application, as well as some variations of them, designed with the aim to improve the localisation of detected vehicles in the generated output. We have obtained the best results combining both strategies. That is, using the strategy of the non–maximum suppression mechanism to detect non–maximum detections, but in order to cluster them (instead of deleting them). Given a set of detections, first the region of maximum confidence is identified. Then, the rest of detections are intersected with this one, grouping the ones whose intersection is over the 50% of the area of the bigger of the two regions intersected. Then the weighted mean and variance of the clustered regions is computed, using the confidence detection value as weight. In that way, the different regions conforming a cluster are reduced to a Gaussian distribution of region parameters, which reflect the uncertainty about the real vehicle location. Thus, the final output of the vehicle detector is a list of normal distributions, detailing the mean and the variance of the region parameters $[x^r\ y^r\ w^r\ h^r]^T$ (i.e., the bottom-left corner coordinates of the vehicle region and its width and height).

Detection Practicalities

Having the whole vehicle detection strategy described, is time to focus on its details. In order to achieve a high detection rate using the proposed scheme, the scanning of the road ahead has to be dense. This means that a huge amount of image regions have to be evaluated by the COC generated. For instance, for a road sampling grid with parameters $d_x = d_z = 10$ cms (see Figure 5.5) it is required to classify between 350.000 and 500.000 regions per frame, depending on the pitch value θ of the acquisition system. This is a remarkably huge number of regions, especially if it is compared to the amount inspected in other application domains. For instance, in

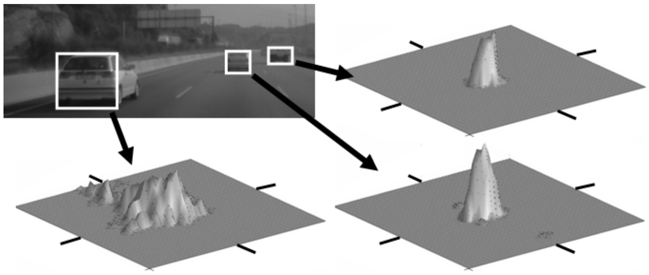


Figure 5.8: Positive detector response in subregions around the optimal vehicle location, for vehicles at near, middle and far distances.

[127] a dense scanning to detect pedestrians consists in evaluating just 12.800 regions per frame. Hence, for the described application, the classification of the scanned regions has to be done very efficiently. In the next section we propose two different techniques to reduce the cost of evaluating the vehicle classifier in frames, namely the lazy evaluation of the COC, and the optimisation of the COC, in order to reduce the average number of features required to classify an image region.

5.1.3 Lazy COC Evaluation

Scanned regions are classified using a COC. Each level of the COC is constituted by a classifier $s()$ as defined in Equation (5.1). Since classification is done in terms of the sign of the value returned by $s()$, it is not always necessary to evaluate all their weak classifiers r_i to produce the classification decision. A classification decision can be taken, as soon as the sum of the accumulated r_i responses has a magnitude bigger than the summation of responses of opposite sign in the remaining rules. More formally, given a classifier $s()$ and the set of features $\mathbf{f}$ of the region to be evaluated, the number of features n_{eff} required to establish a classification decision corresponds to the minimum n accomplishing

$$\left|\sum_{i=1}^{n} r_i(f_i)\right| > \left|\sum_{j=n+1}^{n_f} v_j^{-sign\left(\sum_{i=1}^{n} r_i(f_i)\right)}\right| \quad .$$

We propose to take advantage on this fact to minimise the computation required by the described vehicle detection process, evaluating for each classifier in the COC layer the minimum number of features necessary. To quantify the significance of this lazy evaluation scheme, we have applied the learned COC on a set of testing frames, registering for each processed region the details of the COC evaluation, namely:

- the number of COC layers evaluated to give a classification decision[7];
- the number of features evaluated at each COC layer.

Figure 5.9 displays the statistics of the obtained results, showing the percentage of processed regions that receive a final classification decision at each COC layer, and

[7] Notice that only positive regions are expected to be evaluated in all COC layers.

for each layer, the histogram of the amount of feature needed to classify processed regions. For each layer, the expectation of the number of features evaluated vs the total number of features n_f of the classifier is presented. Considering the evaluation of all the layers, it results that, on average, the standard evaluation of the COC requires computing 102.82 features per region, while the lazy evaluation requires just 76.06. This means reducing a 26% the number of computed features.

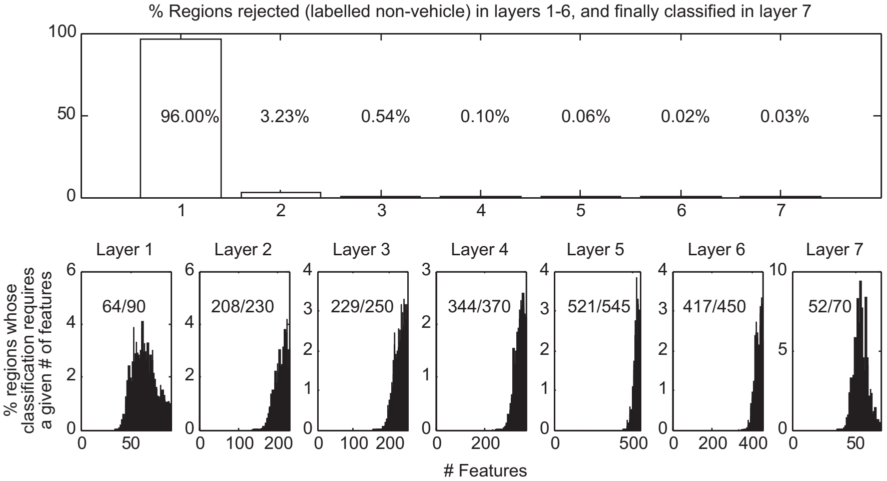


Figure 5.9: Lazy COC evaluation on a testing set. Top: statistics of the lazy evaluation of a COC. Bottom: for each layer, histogram of the number of features needed for classification, remarking with text the average number of features evaluated vs the total number of features of its classifier.

Results also show that, on average, 96% of regions are discarded (i.e., classified as non–vehicles) at the first COC layer. This comes from the fact that processed images present a large homogeneous area (the road), and the image regions croped there are easy to distinguish from vehicles. However, despite most image regions require the evaluation of just a single COC layer, this means evaluating 64.34 features, which involves a noteworthy computational cost, due to the big amount of regions that have to be inspected at every frame. In order to obtain a more efficient vehicle detector, less features should be used to discard this greater part of the analysed regions. Accordingly, we propose in the next section a methodology to further optimise the learned COC.

5.1.4 Optimising a COC

In order to implement with lower computational cost the task of a given layer of a COC, we propose to substitute its corresponding strong classifier by another COC. Ideally, this COC should achieve an equivalent classification performance, requiring the analysis of fewer features when a frame is processed. The method proposed is based on a partition of the training set $\mathbf{t}_{1:N_r}$ used to generate the layer classifier in order to obtain new classifiers of lower complexity. Let's denote $\mathbf{t}_\oplus$ and $\mathbf{t}_\ominus$ the positive and negative examples in $\mathbf{t}_{1:N_r}$ (i.e., $\mathbf{t}_{1:N_r} = \mathbf{t}_\oplus \cup \mathbf{t}_\ominus$). Using the classifier $s()$ learned from it, we classify the elements in $\mathbf{t}_\ominus$, and then we select the ones whose classification remains always negative once the first 10% of weak rules $r_i()$ in $s()$ have been evaluated. This selection groups negative examples according to how *easily* are classified, partitioning $\mathbf{t}_\ominus$ in two groups:

- one with elements easily distinguishable from positive examples ($\mathbf{t}_{\ominus_1}$);
- the other with elements more difficult to classify ($\mathbf{t}_{\ominus_2}$).

Heuristically we hope that from these two sets new classifiers will be learned, that jointly will require evaluating fewer features than the replaced COC layer. From the training set $\{\mathbf{t}_\oplus \cup \mathbf{t}_{\ominus_1}\}$, because it contains examples *clearly* negative , it seems logical to expect learning a classifier requiring less features. For $\{\mathbf{t}_\oplus \cup \mathbf{t}\ominus_2\}$ it is also logical to expect obtaining a classifier of lower complexity, because the Adaboost algorithm will select a subset of features $\mathbf{f}_2$ different to the one of the original classifier $s()$, specially dedicated to distinguish the elements in $\mathbf{t}_{\ominus_2}$ from $\mathbf{t}_\oplus$[8]. Thus, we propose to recursively apply such a divide and conquer strategy, attempting to obtain classifiers of lower complexity. Figure 5.10 sketches this idea. The subset $\mathbf{t}_{\ominus_1}$ is recursively purged using the described method, until either a classifier with a constrained maximum complexity is obtained, or the complexity of the classifier obtained does not decrease. Then, the examples discarded during this process are grouped in a new training set $\mathbf{t}\ominus'$, and the process is started again. The process is stopped when no significant improvement is achieved.

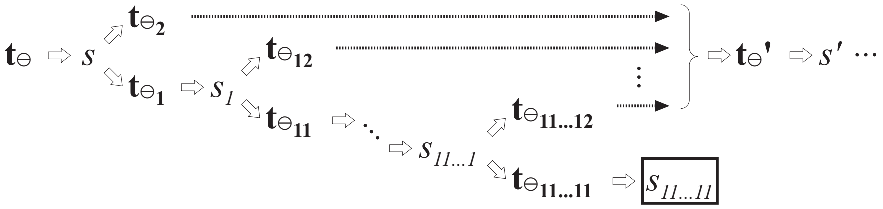


Figure 5.10: Strategy proposed to substitute a classifier $s()$ by a COC.

Using this strategy, the first level of the cascade analysed in Figure 5.9 has been replaced by 4 new sub-levels which, when applied on testing frames, present the statistics of Figure 5.11. The joint performance of these new 4 layers is compared in Figure 5.12 with the performance of the replaced layer. If in the process of scanning an image 96% of regions were discarded in the first COC layer, requiring on average computing 64 features per region, now this same amount of regions is discarded requiring just evaluating 33 features per region.

If the cost of evaluating the whole COC is considered, the average number of features required per inspected region is now 43.35, which, with respect to the 76.06 of the original COC, means a reduction of the 43%. If we make this same comparison

[8] However, in case that this would not happen, one could just use the original $s()$ for classifying $\mathbf{t}_{\ominus_2}$.

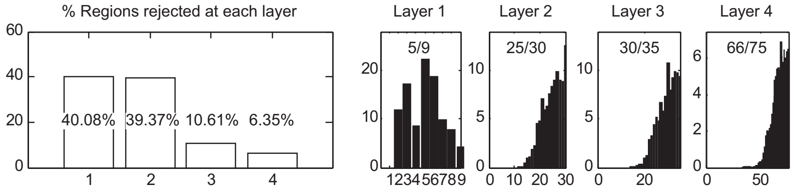


Figure 5.11: Performance of the new COC layers. Left: statistics of the layers that replace the first layer of the COC in Figure 5.9. Right: for each new layer, histogram of the number of features needed for classification, remarking with text the average number of features evaluated vs the total number of features of its classifier.

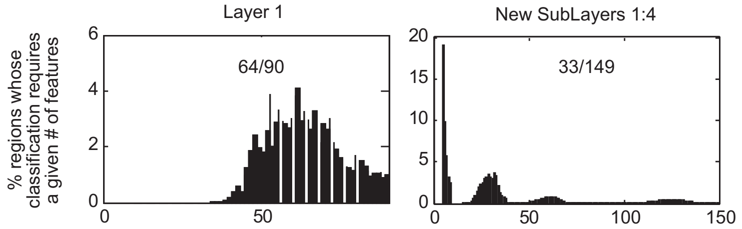


Figure 5.12: Performance of the initial COC layer vs the new learned sublayers. The new sublayers cuts the average number of features required to classify regions approximately by half (from 64 to 33).

against the original COC with standard evaluation (102.82 features where required on average), we have reduced around the 58% the number of features evaluated.

5.1.5 Performance Evaluation

To objectively evaluate the performance of the proposed vehicle detection module, the following experiment has been carried out. First, sequences different from those employed for training has been used, which were acquired by three different vehicles and video cameras. Each camera has fixed optics, and has been calibrated assuming a pin–hole camera model with zero–skew, with the software provided in [21]. The images provided by each camera are significantly different, due to their different spectral sensitivity and automatic gain control mechanism. Sequences have been acquired at different hours of the day (midday and sunset) and environmental conditions (cloudy and sunny). From them, 500 frames have been selected in order to construct a testing set to validate the system. The selection criterion has been collecting frames showing a variety of vehicles and lighting conditions (presence of shadows, specularities, under-illuminated environments, etc.). In all selected frames a planar surface is annotated approximating the observed road. This annotation is easy if parallel road structures (lane markings, road limits, etc.) are clearly observed in the image. The annotated plane provides indirectly the camera pitch angle θ to determine the frame regions to be inspected. With this information, an ideal scanning of video frames is carried out, and the best performance achievable for the proposed method can be quantified[9]. The vehicles in testing frames have also been manually annotated, being labelled as

- *detectable*, if their detection should be mandatory;
- *miss–detectable*, if they can be miss–detected due to some of the following causes:
  - present partial occlusions;
  - are farther than the maximum operative detection distance (70 meters);
  - lay in another plane different than the scanned.

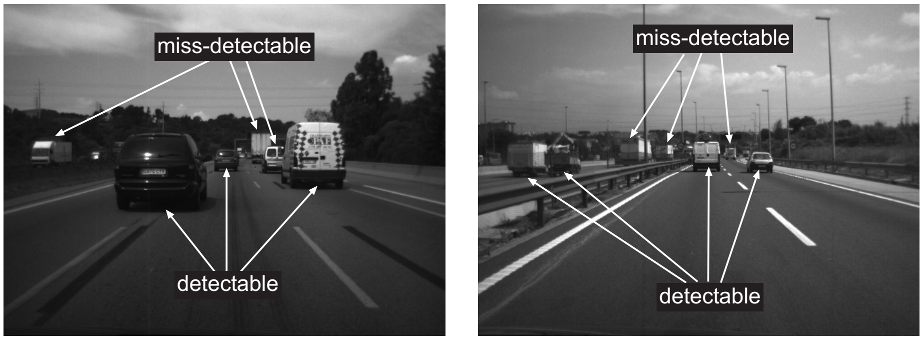


Figure 5.13: Examples of detectable and miss–detectable vehicles.

The labelling of observed vehicles in these two disjoint classes (see Figure 5.13) has been done to better quantify the detection performance (i.e., count properly the number of false positives and false negatives of the detector). The miss-detection of a *miss-detectable* vehicle does not have to be interpreted as a false negative, since the objective is not evaluating the detection performance in these challenging cases. On the other hand, *miss-detectable* vehicles, being detected or not, are counted neither as true nor false positives, in order to not distort results. Thus, classification ratios are computed taking into consideration just vehicles that *must* be detected. Table 5.1 and Figure 5.14 show the results obtained for a dense scanning[10] of testing frames, using the original and the optimised COC respectively. Using the optimised COC, a slightly lower detection rate is achieved (93.91% versus the 94.13% of the original COC), but also a lower false positive rate per evaluated region. The detection accuracy achieved is remarkable, due to the complexity of the faced problem (detection of vehicles up to 70 meters away), and the challenging conditions considered in the testing (different acquisition cameras, daytime conditions, frontal and rear vehicle views, etc.).

The detector has a better performance in detecting the rear side of vehicles, probably due to the fact that frontal views are underrepresented in the training set (they constitute less than the 10% of positive training examples). Concerning the type of vehicles, those more difficult to detect are trucks. We guess that this is due to two

[9] It would be unfair to evaluate the performance of the detector scanning unproperly frames, as missdetections could be due to the lack of inspection of regions containing vehicles.

[10] The ground plane has been inspected using a dense grid of points, with $dx = dz = 0.1$ meters.

Table 5.1: Detection results of the original (top) and optimised (bottom) COC.

Original COC - True Positives Detection rates					
	Car	Van	Truck		Acum.
Rear	547/570 (95.96%)	163/169 (96.45%)	67/78 (85.90%)	⇒	777/817 (95.10%)
Front	67/80 (83.75%)	11/12 (91.67%)	11/11 (100.00%)	⇒	89/103 (86.41%)
	⇓	⇓	⇓		⇓
Acum.	614/650 (94.46%)	174/181 (96.13%)	78/89 (87.64%)	⇒	866/920 (94.13%)

Original COC - False Positives Detection rates	
FP per Window evaluated: 1.509e-004	FP per Frame: 1.07

Optimised COC - True Positives Detection rates					
	Car	Van	Truck		Acum.
Rear	545/570 (95.61%)	162/169 (95.86%)	68/78 (87.18%)	⇒	775/817 (94.86%)
Front	67/80 (83.75%)	11/12 (91.67%)	11/11 (100.00%)	⇒	89/103 (86.41%)
	⇓	⇓	⇓		⇓
Acum.	612/650 (94.15%)	173/181 (95.58%)	79/89 (88.76%)	⇒	864/920 (93.91%)

Optimised COC - False Positives Detection rates	
FP per Window evaluated: 1.426e-004	FP per Frame: 1.02

factors. On one hand, trucks conform a class with a bigger within–class variance than other types of vehicles. On the other hand, the appearance of their rear side usually vary very significantly depending on the camera viewpoint. This happens less pronouncedly in the other types of vehicles, where the observed vehicle parts are approximately at the same distance to the camera. No parallax is appreciated, and for this reason their appearance scarcely varies with the camera viewpoint. Another point worth to mention is the number of FPs. On average around 1 FP per frame is generated, which is significantly big. However, in practice this does not represent a major problem, as in real sequences FPs present no spatio–temporal coherence, and can be distinguished from true detections. The suppression of spurious false detections is solved by the system module described in the next section, devoted to estimate automatically the 3D state of detected vehicles, in order to initialise the state of a vehicle tracking algorithm.

5.2 3D Vehicle Initial State Estimation

The output of the vehicle detector in the preceding section provides the statistics (mean and variance) of the rectangular regions where vehicles have been detected. These detections have to be reexpressed in terms of their corresponding 3D world coordinates, in order to be useful for ADAS applications. This section details our proposal to carry out this task, and presents an algorithm to construct trajectories from the generated 3D coordinates, which allows to estimate the relative 3D velocity of vehicles, as well as to discard false detections.

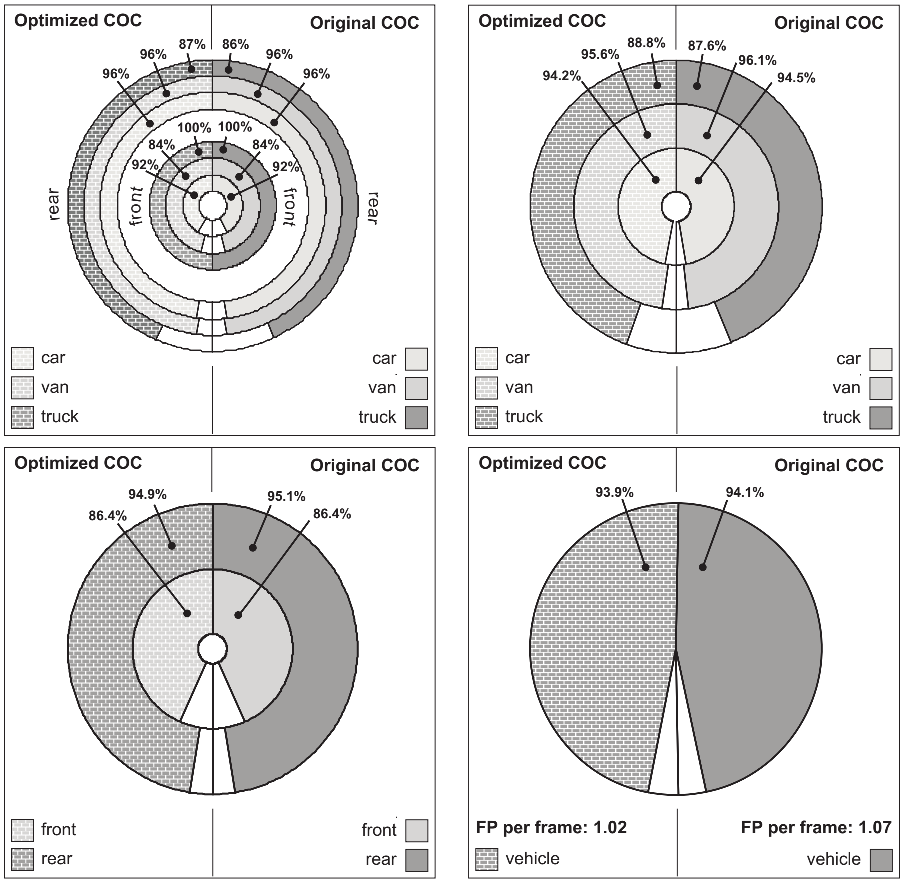


Figure 5.14: Detection rates of the original and the optimised COC, using different grouping criteria.

5.2.1 From 2D Vehicle Detections to 3D Vehicle Locations

Each detection is specified by a Gaussian distribution $\mathcal{N}(\mu^r, \Sigma^r)$, where μ^r details the 2D region parameters $[x^r \; y^r \; w^r \; h^r]^T$, and Σ^r their uncertainty. To extract the 3D vehicle location from this observation, $\mathcal{N}(\mu^r, \Sigma^r)$ has to be backprojected onto their corresponding 3D road coordinates. In general, this backprojection can not be solved, but by assuming that the road conforms to a flat surface (which is realistic for motorways and high speed roads), and that the camera position with respect to it is known, this is feasible. Using the model of camera projection presented in (5.2), the bottom–left corner coordinates $[x^r \; y^r]^T$ of detected regions are related with the 3D road coordinates $[x \; 0 \; z]^T$ by

$$x = \frac{f^y \, h \, \left(x^0 - x^r\right)}{f^x \left(\left(y^0 - y^r\right) \cos(\theta) + f^y \sin(\theta)\right)} \;, \tag{5.5}$$

$$z = -\left(\frac{h \, \left(f^y \cos(\theta) - \left(y^0 - y^r\right) \sin(\theta)\right)}{\left(y^0 - y^r\right) \cos(\theta) + f^y \sin(\theta)}\right) \;. \tag{5.6}$$

These expressions require the knowledge of the camera extrinsic parameters (h, θ), which as previously remarked, can vary at every frame due to the action of the host vehicle suspension system. The variation of h can be ignored, as it provokes a negligible error on x and z estimations, but a correct θ value is essential to estimate them accurately. With the objective of developing a method independent of θ, we exploit the following relation: given a vehicle detection, the value of θ can be estimated if the width w of the observed vehicle and its yaw angle φ relative to the Y–axis (see Figure 5.15) is known. At ground level, the value $w' = w \cos(\varphi)$ projects onto w^r image pixels according to

$$w^r \sim \frac{f^x \, w'}{z \cos(\theta) - h \sin(\theta)} \;. \tag{5.7}$$

Combining Equations (5.6) and (5.7), we found that

$$\theta = 2 \arctan\left(\frac{1 - \sqrt{1 + \left(\frac{y^0 - y^r}{f^y}\right)^2 - \left(\frac{w^r h}{w' f^x}\right)^2}}{\frac{y^0 - y^r}{f^y} - \frac{w^r h}{w' f^x}}\right) \;, \tag{5.8}$$

and this makes feasible using Equations (5.5) and (5.6) to hypothesise the 3D vehicle location. Another interesting advantage of knowing w is that the estimation of x does not require anymore the value of θ. Combining (5.3) with (5.7), it follows that

$$x = (x^r - x^0) \frac{w}{w^r} \;. \tag{5.9}$$

Unfortunately, to estimate θ the values of w and φ of observed vehicles are required, which are also unknown a priori. In practice, the value of φ is usually zero, except in some specific driving manoeuvres, as for instance, vehicle overtaking. However, whichever its value, in a short time interval[11] usually it can be correctly assumed

[11] As the one considered in the process of constructing a trajectory from consecutive vehicle detections, in order to compute the 3D state of the vehicle (see Section 5.2.2).

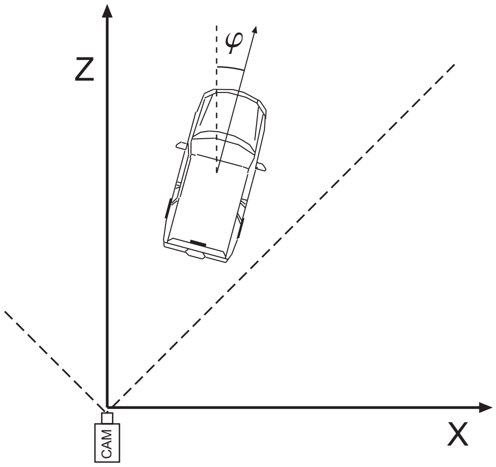


Figure 5.15: Top view of the road coordinate system, detailing the vehicle orientation angle φ.

as constant. With respect to w, the range of feasible values is quite wide, since a detected vehicle may correspond to a small car ($w \sim 1.5$ meters), or to a big truck ($w \sim 3$ meters). Thus, a priori one may think that there is no advantage on using our proposal to estimate x and z; if first we needed to guess θ, now we need to guess φ and w. However, there is an important difference between both situations. Given the detections of a vehicle at successive frames, the accuracy of their road location obtained by guessing directly θ will vary at each frame, depending on how close this value is to the real camera pitch angle. On the other hand, road locations derived by assuming certain (φ, w) values will present the same systematic error in all frames, induced by the wrong assumed values. Due to that, locations extracted in consecutive frames, although erroneous, will manifest an spatio–temporal coherence, which will easily allow to construct trajectories from them. These trajectories will be incorrectly localised on the road, but if afterwards a more precise guess on the (φ, w) values is available, their localisation can be corrected. Details are provided in Section 5.2.5.

Backprojecting Vehicle Detections

The output of the vehicle detector does not provide simple 2D regions, but normal distributions $\mathcal{N}(\mu^r, \Sigma^r)$ of parameters (x^r, y^r, w^r) (h^r is discarded because is irrelevant in the proposed backprojection scheme). Backprojecting these parameter distributions onto their corresponding 3D road locations is not straightforward, as backprojection Equations (5.5) and (5.6) are non–linear. To solve that, we propose to use the Unscented Transform [64] mechanism, described previously in Section 2.3.1. In short, this transform involves the following procedure: a set of samples (sigma–points) are deterministically chosen from the detection distribution, which jointly match its mean and covariance. These samples are backprojected into road coordinates with Equations (5.5) and (5.6), using the value of θ computed from Equation (5.8), for an assumed vehicle width and orientation w, φ parameters. Computing the sample mean

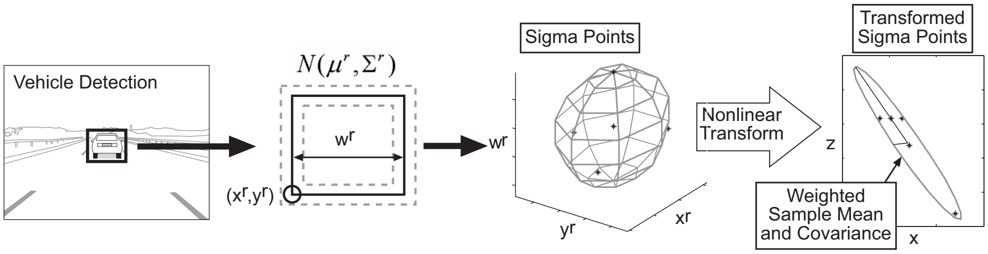


Figure 5.16: Transformation of 2D detections into 3D road coordinates.

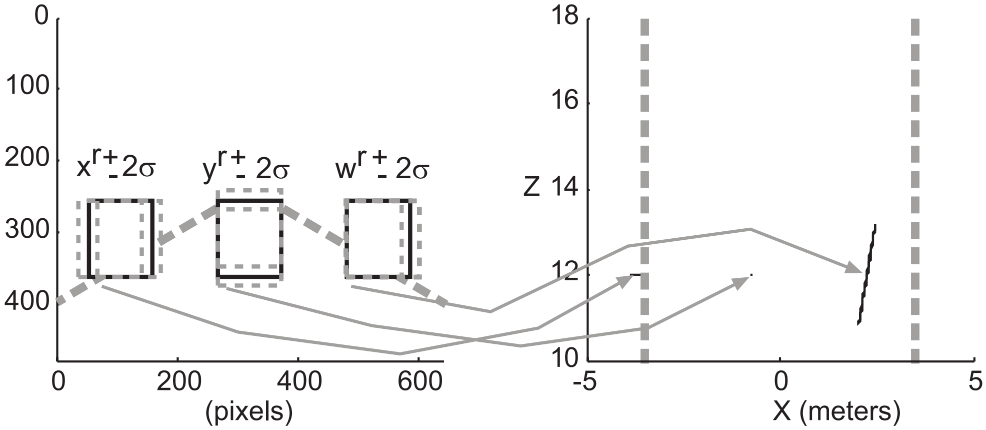


Figure 5.17: Backprojection procedure. Left: distributions of 2D region parameters (in pixels). Right: the corresponding distribution on road coordinates (in meters). In both plots, bold dashed lines represent the road plane. It can be seen that, given a detection with parameters (x^r, y^r, w^r), a fixed value of uncertainty ($\sigma = 5$ pixels) is further more critical in w^r (rightmost 2D region) than on the other parameters.

and covariance of backprojected samples (properly weighted with defined factors), the Gaussian distribution fitting the transformed distribution is characterised (see Figure 5.16). The uncertainty of the final vehicle location distribution depends on the particular uncertainty in each detection parameter, being w^r the one that affects 3D locations the most (see Figure 5.17).

5.2.2 From 3D Vehicle Locations to 3D Vehicle Trajectories

The procedure described up to this point analyses sequences in a frame-by-frame basis, and it can not provide information about the dynamics of detected vehicles. Given the output of the vehicle detector, it generates a list of 3D location distributions where vehicles can lie. These distributions may correspond to real vehicles or not, so some mechanism is required to deal with the presence of false alarms, and estimate the trajectory of genuine detections. There are two different philosophies to perform this task.

One option is to use the *Track–Before–Detect* (TBD) methodology. This approach is applied in situations where the signal is noisy, and targets can have a very weak response to its detector (i.e., the sensor has a low SNR). In order to detect them, algorithms must apply a very low detection threshold, generating an output with many false positives. To avoid these false detections, TBD methods propose to detect targets not analysing the sensor signal frame by frame, but analysing a sequence of consecutive frames simultaneously. The idea is taking advantage that true targets should have well behaved trajectories, while false alarms are uncorrelated and unlikely to form *reasonable* trajectories over time. Thus, they check the spatio-temporal signal with a model of expected target dynamics, that when it matches, verifies the presence of a real target, and at the same time determines its dynamics. It that way, the generation of false detections is prevented. Examples of this strategy can be found in [121, 20], which essentially implement an integration of the target energy through a sequence considering a set of potential trajectories.

Another possibility to deal with this problem is using a *Detect–Before–Track* (DBT) strategy. In essence this approach uses the same spatio–temporal criterion as TBD methods, but this time oriented to discard false detections instead of avoiding its generation. The idea is to connect consecutive detections along time, constructing target trajectories or *tracks*. As new observations are collected, they may verify the correctness of trajectories currently considered, as well as starting new ones. This process of trajectory construction is not always trivial, as several detections may be coherently assigned to a given trajectory at a given instant (this situation is commonly referred as *data association* problem). In general, in case of ambiguous observation assignments, the best option is to consider all the feasible hypothesis, and for this reason Multiple Hypothesis Tracking (MHT) methods [16] are commonly applied on this problem. Similarly to the multiple model algorithms introduced in Chapter 4, at each time step a tree of hypothesis is expanded for active trajectories, but this time with the purpose of evaluating the different possible associations between trajectories and observations. Each considered trajectory is managed by an estimation method (commonly a Kalman–based filter) that integrates the information of assigned detections in an state being estimated. Trajectories started from false detections are discarded in a few frames, as future detections can not be coherently assigned to them. On the other hand, *true* trajectories collect observations in successive frames, confirming the presence of a real vehicle, and simultaneously estimating its dynamics. Examples of these methods can be found in [25, 34].

For the system being developed, the DBT strategy has been chosen, as the vehicle detector used has a high SNR. There exist many different MHT algorithms in the literature that could be used to implement a solution. These algorithms basically differ in how they deal with the data association problem. In general, the tree of trajectories can grow exponentially with the observations. If occasional miss-detections must also be considered, the growth in the number of hypothesis is even more severe. Hence, in order to fulfil computational requirements, existing algorithms propose different strategies to consider only a (presumably good) subset of all the possible trajectories. Because of that, they are usually referred as *suboptimal* algorithms. In this thesis we use the more conventional MHT algorithm, based on solving the data association problem using the Global Nearest Neighbour (GNN) approach. It determines the most likely assignment of input observations to existing trajectories, under the constrain that an observation can be associated with at most one track. Unassigned observations initiate new trajectories. Inherently, it considers that an observation is produced by a single target, which happens in the studied application. The fact that in the vehicle tracking problem targets are widely spaced and false alarms occur uncorrelatedly also makes the GNN approach a good a priori choice. Next section details our implementation of this algorithm, based on the multiple target Kalman tracker described in [34].

5.2.3 Trajectory Construction

Given a frame, its analysis provides a list of N observations, which are Gaussian distributions $\mathcal{N}(\mathbf{y}_t^i, \mathbf{R}_t^i)$ on the road location where vehicles may be present. $\mathbf{y}_t^i = [x_t^i \; z_t^i]^T$ specify the road coordinates of the i-th vehicle, and $\mathbf{R}_t^i$ is a covariance matrix denoting their uncertainty. Initially, for each observation $(\mathbf{y}_t^i, \mathbf{R}_t^i)$, a first-order KF is initialised, with state

$$\mathbf{x}_t^k = \mathbf{y}_t^i \;, \qquad \Sigma_t^k = \mathbf{R}_t^i \;,$$

where k is a label identifying the different KFs that simultaneously are active. The dynamics and the observation process of the KFs are assumed linear, defined as

$$\begin{aligned} \mathbf{x}_t^k &= \mathbf{A}\mathbf{x}_{t-1}^k + \mathbf{v}_t \;, \\ \mathbf{y}_t^k &= \mathbf{x}_t^k + \mathbf{w}_t \;, \end{aligned} \tag{5.10}$$

where $\mathbf{A}$ is the dynamic matrix, and $\mathbf{v}_t$ and $\mathbf{w}_t$ are perturbations with distributions $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$, $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$. The first time that a KF is initialised, as no information on the target dynamics is available, it is assumed that its state remains constant, except for a given Gaussian distortion; That is, $\mathbf{A}$ is an identity matrix, and $\mathbf{Q}_t$ takes into consideration the maximum variation that the position of a vehicle can experiment between consecutive frames. Using Equation (5.10) the vehicle state at t is predicted, delimiting a region of interest on the road (a *validation gate*), that constraints which observations at instant t can be assigned to the KF. This validation gate is defined in terms of a Mahalanobis distance between the prediction and the observations. Given a k-th KF, the j-th observation is at distance $d(k,j)$ from its prediction, given by

$$d(k,j) = \sqrt{(\mathbf{x}_{t|t-1}^k - \mathbf{y}_t^j)^T (\Sigma_{t|t-1}^k)^{-1} (\mathbf{x}_{t|t-1}^k - \mathbf{y}_t^j)} \;.$$

Using this distance, an observation $\mathbf{y}_t^j$ is assigned to the k-th KF iff

1. lies inside a region of interest delimited by thr, i.e.,

$$d(k,j) < thr \;;$$

2. it is the closest observation to the k-th prediction, i.e.,

$$d(k,j) < d(k,i) \qquad \forall\, i = 1, \ldots, N \; i \neq j \;;$$

3. from the set of active KFs (trajectories) which has no yet received a observation, there is no other l-th KF prediction with $l \neq k$ closer to this measurement, i.e.,

$$d(k,j) < d(l,j) \qquad \forall\, l = 1, \ldots, K \text{ with}\, l \neq k \ .$$

The first time a KF of first order (KF1) receives an observation, it does not reestimate its state, but upgrades it to an state of second order. Thus, the KF state is set to

$$\begin{aligned} \mathbf{x}_t^k &= [\ \mathbf{y}_t^i \quad \mathbf{y}_{t-1}^j\ ]^T \ , \\ \Sigma_t^k &= \begin{bmatrix} \mathbf{R}_t^i & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{t-1}^j \end{bmatrix} \ , \end{aligned}$$

where here $\mathbf{0}$ is a 2×2 zero matrix. The state transition matrix $\mathbf{A}$ is updated to express a model of constant velocity, which has been found a good approximation of the behaviour of vehicles in the analysed application. It is given by

$$\mathbf{A} = \begin{bmatrix} \frac{1+\Delta_t}{\Delta_t} & 0 & -\frac{1}{\Delta_t} & 0 \\ 0 & \frac{1+\Delta_t}{\Delta_t} & 0 & -\frac{1}{\Delta_t} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \ ,$$

where Δ_t is the time interval between acquired images. The noise process $\mathbf{Q}_t$ is readjusted too, as more accurate predictions now will be made. Future observations assigned to this second order KF (KF2) will be used to reestimate the vehicle state using the well–known Kalman update equations (see Appendix B).

If in a given frame, some observations are not assigned to any existing KF, they are used to start new first–order KFs. On the other hand, if a KF receives no observation, it persists active over some frames trusting in its state predictions, being robust in that way to occasional vehicle miss-detections. At each frame, a rule–based procedure checks active trajectories to verify the detection of vehicles. Once a KF has collected a given number of observations, the presence of a vehicle is confirmed. On the contrary, when a KF does not receive observations over several frames, it is discarded and erased from the list of KFs. This happens for example in KFs started from false detections, or when a vehicle is occluded or leaves the scene. Figure 5.18 sketches the described process.

5.2.4 Trajectory Characterisation

Once a vehicle detection is confirmed, the next step is characterising the trajectory that it has described, making maximal use of the available information. Using the KF, the distribution of the feasible vehicle states at each instant t, using observations available up to this same instant has been estimated (i.e., $p(\mathbf{x}_t|\mathbf{y}_{1:t})$). A maximal use of the N collected observations can be done, by considering all of them to estimate the vehicle state at each instant t. Formally, this corresponds to estimate $p(\mathbf{x}_t|\mathbf{y}_{1:N})$ where $N \geq t$. In estimation theory terms, this procedure corresponds to the *smoothing* of the trajectory. To do that, we have used a Kalman Smoother (KS) based on the

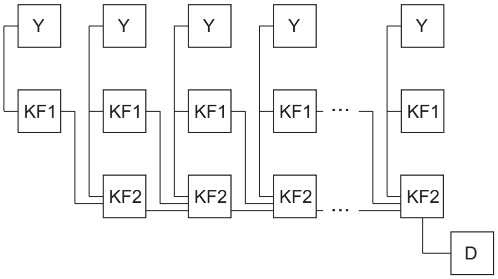


Figure 5.18: Sketch of the trajectory construction procedure. For each frame several observations Y are obtained, which are assigned (if proper) to active first and second–order trackers KF1 and KF2. The non-assigned observations initialise new KF1s. After a KF2 successfully collects a given amount of observations along time, the detection D of a vehicle is confirmed.

Rauch–Tung–Striebel fixed–interval optimal smoother [47]. This algorithm estimates a smoothed trajectory, from where we then computed the forward velocity v_t and orientation φ_t of the vehicle. First, we project the states of the smoothed trajectory to the space of Euclidean velocities. Given an state $\mathcal{N}(\mathbf{x}_t^k, \Sigma_t^k)$, the distribution of its Euclidean velocity $[{v_x}_t^k \ {v_z}_t^k]^T$ corresponds to $\mathcal{N}(\mathbf{E}\mathbf{x}_t^k, \mathbf{E}\Sigma_t^k\mathbf{E}^T)$, where

$$\mathbf{E} = \left[\begin{array}{cccc} \frac{1}{\Delta_t} & 0 & -\frac{1}{\Delta_t} & 0 \\ 0 & \frac{1}{\Delta_t} & 0 & -\frac{1}{\Delta_t} \end{array} \right] .$$

Then, we reexpress the Euclidean velocity distribution in polar coordinates $[v_t^k \ \varphi_t^k]^T$. We do that because in polar coordinates velocity parameters can be interpreted separately and more directly, and this benefits posterior processes that use them. For instance, in order to estimate their values in a tracking algorithm, the dynamics of $[v_t^k \ \varphi_t^k]^T$ can be modelled very accurately using a simple first order AR model. The transformation from Euclidean to polar velocities is done using again the UT mechanism (see Section 5.2.1), now with non–linear expressions

$$\begin{aligned} v_t^k &= \sqrt{({v_x}_t^k)^2 + ({v_z}_t^k)^2} \ , \\ \varphi_t^k &= \arctan({v_x}_t^k, {v_z}_t^k) \ . \end{aligned}$$

Figure 5.19 sketches the proposed procedure. The distribution finally obtained codifies the direction of movement of a target integrally in φ. This is unconvenient for some posterior applications like vehicle trackers, as in some situation the evolution of φ may present sudden changes along time. For instance, imagine that a vehicle overtakes the host, and after the host vehicle accelerates approximating to it. In this case, the first computed φ will have a value around zero, which will change abruptly to around $\pm\pi$ when the host vehicle begins to approximate to the target. To deal with these common situations, is more convenient to express the direction of movement of a target in the sign of the forward velocity v, maintaining in φ the relative orientation of the vehicle with respect to the host. Given a polar velocity (v_t^k, φ_t^k), this means restrict $\varphi_t^k \in [-\pi/2, \pi/2]$, changing the sign of the forward velocity v_t^k if required.

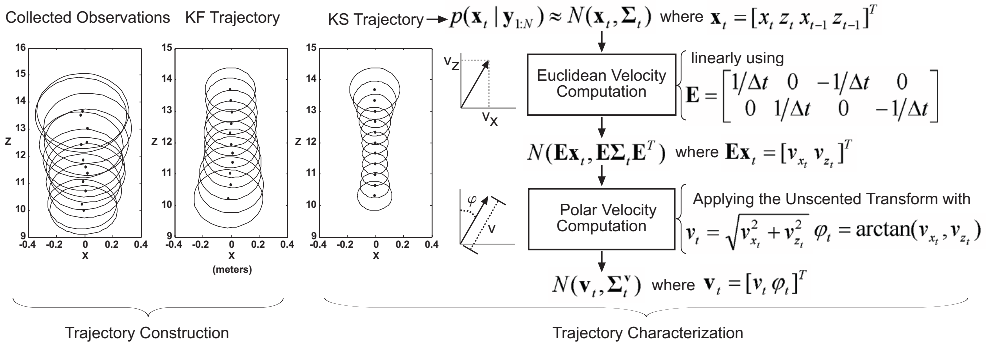

Collected Observations
KF Trajectory
KS Trajectory
$p(\mathbf{x}_t \mid \mathbf{y}_{1:N}) \approx N(\mathbf{x}_t, \boldsymbol{\Sigma}_t)$ where $\mathbf{x}_t = [x_t\ z_t\ x_{t-1}\ z_{t-1}]^T$
Euclidean Velocity Computation
linearly using
$\mathbf{E} = \begin{bmatrix} 1/\Delta t & 0 & -1/\Delta t & 0 \\ 0 & 1/\Delta t & 0 & -1/\Delta t \end{bmatrix}$
$N(\mathbf{E}\mathbf{x}_t, \mathbf{E}\boldsymbol{\Sigma}_t\mathbf{E}^T)$ where $\mathbf{E}\mathbf{x}_t = [v_{x_t}\ v_{z_t}]^T$
Polar Velocity Computation
Applying the Unscented Transform with
$v_t = \sqrt{v_{x_t}^2 + v_{z_t}^2}\ \varphi_t = \arctan(v_{x_t}, v_{z_t})$
$N(\mathbf{v}_t, \boldsymbol{\Sigma}_t^{\mathbf{v}})$ where $\mathbf{v}_t = [v_t\ \varphi_t]^T$
vz
vx
φ
v
z
x
(meters)
16
15
14
13
12
11
10
9
-0.4
-0.2
0
0.2
0.4
Trajectory Construction
Trajectory Characterization

Figure 5.19: Process to estimate the 3D velocity vehicle.

Initial Tracking State Determination

Once the vehicle trajectory has been characterized, the initial tracking state for the vehicle tracking module is determined. Since the trajectory has been constructed from vehicle locations estimated using an arbitrary hypothesis [12] of the vehicle width and height, first it has to be corrected according to a more plausible hypothesis (i.e., a hypothesis sustained by some kind of evidence). Using the information on the camera θ value that available at this instant (as will be shown posteriourly, the tracking module can provide a guess on this value), and using the width w^r of the last detection added to the trajectory, a more confident hypothesis on the vehicle width w is determined from the relation in Equation (5.7)[13]. With this new hypothesis on w, the trajectory can be corrected properly (details are given in Section 5.2.5). Using the corrected trajectory, the initial tracking state of the vehicle is determined, which consists in the following parameters (see Section 5.3):

(x, z) : the road location of the vehicle, obtained from the last trajectory state,

(v, φ) : the vehicle velocity in polar coordinates, determined from the average polar velocity along the whole smoothed trajectory,

$d\varphi$: the yaw rate, that is, the velocity in which the vehicle changes its orientation φ. This value will be assumed zero.

This values are given to the tracking module, together with a hypothesis on the vehicle width and height (w, h). We determine the value of h simply from the value of w estimated, according to the average aspect ratio of the detection regions processed to construct the trajectory.

[12] In the sense that it has been fixed before hand, without using any available information to establishing it.

[13] Here we need also the value of the vehicle orientation. We take $\varphi = 0$, which holds in most cases. However, it is shown in Section 5.2.5 that for near vehicles, φ could be estimated from the trajectory to be corrected if the noise in the locations conforming the trajectory were negligible.

5.2.5 Performance Evaluation

The presented proposal shows qualitatively a very good performance on real sequences, specially in discarding false vehicle detections. However, a quantitative analysis is necessary to validate it objectively. Due to the lack of ground truth information in the available real sequences (i.e, knowledge about the real 3D location and velocity of vehicles in all frames), the system has been evaluated using synthetic data. Its accuracy depends on many factors (the 3D position of vehicles, their velocities, the noise perturbing detections, the fulfilment of assumptions made, etc.). Making an exhaustive study of all possible situations is unfeasible. However, a selected set of situations has been considered. In the case of errors derived from breaking assumptions done, we provide with expressions to correct estimated trajectories, if at some point information to correct the assumptions done is available.

The effect of noise

Our first study done has focused on evaluating the system performance in situations where assumptions hold, but vehicles are noisely detected and occasionaly missdetected. Synthetic vehicle trajectories have been generated, and the parameters of their corresponding 2D image regions computed. An artificial variance has been given to these parameters, emulating that of real detections. Three different situations have been analyzed, showing a vehicle at three different starting points, with a velocity relative to the host of 6 Km/h. The vehicle detection parameters (x^r, y^r, w^r) have been randomly distorted by different amounts of noise, corresponding to disturbances of the real vehicle position and width between $[-0.15, 0.15]$ meters. Missdetection events with probabilities between 0 and 40% have been considered. Figure 5.20 gives some examples of the type of trajectories processed.

For each considered noisy situation, 100 different random sequences have been generated and processed, computing the disparity between the real and recovered trajectory, forward velocity, and orientation. The performance criterion used is the average of the Mean Square Error (MSE) between the estimated and the ground truth trajectory in the 100 experiments. The presence of vehicles is confirmed, when their corresponding trajectories collect 12 observations, which has been found sufficient to filter out false positives. This means that in a real–time implementation, a correct vehicle detection will be verified approximately in a second. Figure 5.21 shows results obtained.

With respect to the disparity between the real and the recovered trajectory, the system performs similarly wherever the trajectory starts. Noise on detections does not increase significantly the disparity, while on the other hand, the presence of missdetections degradates notably the achieved accuracy.

In the estimation of the vehicle velocity $[v, \varphi]^T$, results achieved depend on the degree of overlap between the 3D road regions connected to conform a trajectory. The best results are obtained for close vehicles, as the uncertainty in the 3D road locations composing the trajectory is small. Concerning the lateral position of vehicles (x coordinate), centered ones provide a more accurate estimation of their orientation φ, while the estimation of the module of their velocity v is a little bit more imprecise. Notice that the factor that affects more critically the estimation of the vehicle velocity

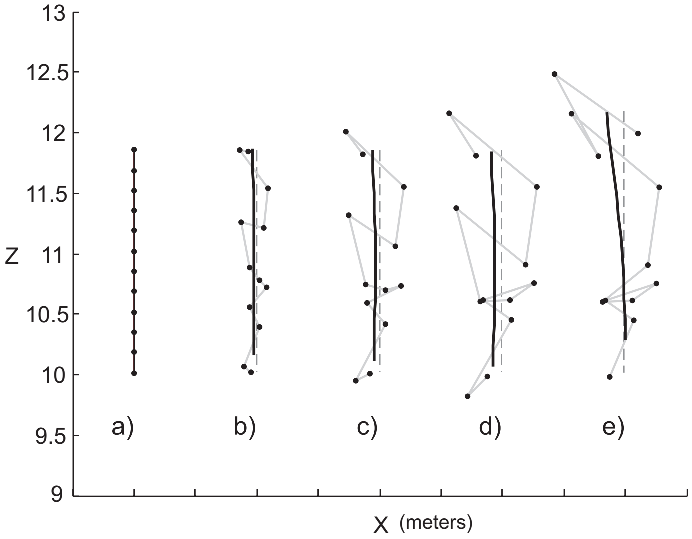


Figure 5.20: Example of noisy trajectories evaluated (light gray lines): a) No noise. From b) to d), trajectories with incremental added noise to the 2D detected image region. e) Same as d), but with some missing observations due to occlusion. Dark lines show recovered trajectories with the proposed algorithm. Dashed lines show the ideal trajectory.

is the noise perturbing observations, while unexpectely, miss-detections clearly favour its better estimation. This results from the fact that, due to miss-detections, it takes a longer time to obtain the N observations that compose a *complete* trajectory, and this favours having observations more distant (in spatial terms) between each other, which will present an inferior degree of overlapping. This attenuates the effect of the uncertainty of 3D locations in the velocity estimation (Figure 5.22).

In a different experiment, the presence of false positives in the observations has also been evaluated. In each frame, false detections has been added, distributed randomly according to a uniform distribution. Results obtained are practically identical to the ones in Figure 5.21, showing the great capacity of our proposal to discard uncorrelated false detections.

The significance of violated assumptions

Our proposal to construct vehicle trajectories assumes a vehicle with width w^a, moving at constant velocity with an orientation $\varphi = 0$. Assuming constant velocity is usually not a problem, as holds for the small period of time required to confirm a vehicle detection. Furthermore, the stochastic term of the dynamic model in (5.10) accounts for slight deviations on this assumption. However, assumptions on w^a and φ are more arbitrary. Given a vehicle, the only a priori knowledge available on its width w is a range of its feasible values (commonly $w \in [1.5, 3]$ meters). So its very likely to use a w^a deviated from the real w, provoking inaccuracies on the 3D estimations. With respect to φ, its value is in most cases really around zero, but in situations like

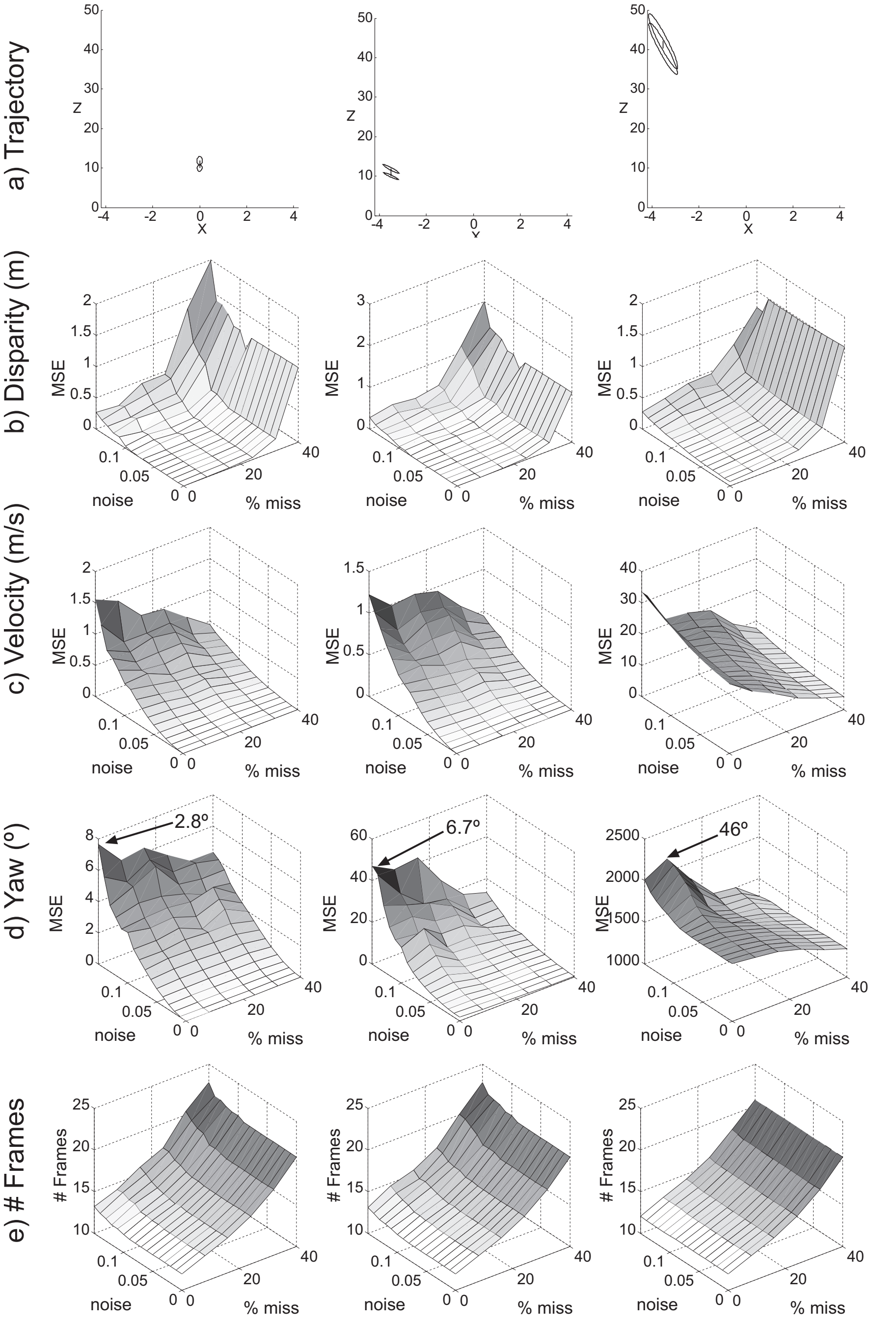


Figure 5.21: Trajectory construction performance. a) Extremes of the three types of trajectories considered. From b) to d): average MSE between real and recovered 3D trajectory locations, forward velocities , and orientations, in 100 different experiments done. e) Average number of frames required to confirm a vehicle detection.

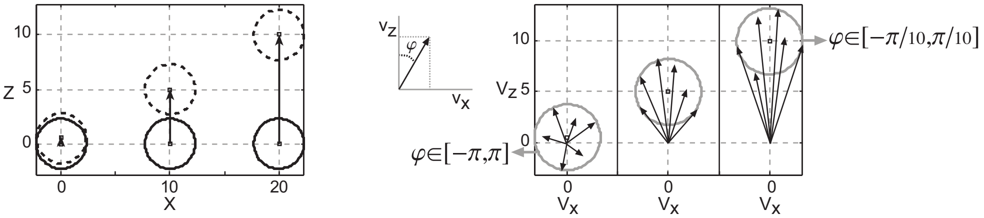


Figure 5.22: Velocity estimation. Left: consecutive observations of three trajectories. Right: distribution of their corresponding velocity. The more distant observations are, the less uncertain φ is.

lane change or taking a curve, it can deviate considerably from this value. The effect of violating these two assumptions is evaluated on the next sections, where we also present a mechanism to a posteriori correct estimated trajectories.

Wrong vehicle width hypothesis ($w^a \neq w$) The effect of this error on the estimation of the location of vehicles has a different behaviour for the x and z coordinates. Provided that the assumed vehicle width w^a is deviated from its real value a given scale factor α (so that, $w^a = \alpha w$), then, from (5.9), the estimations of x are deviated from their corresponding real location by this same scale factor. Hence, if at any point the real vehicle width is available, x coordinates can be corrected just by balancing the factor w/w^a.

The error in z is not so easily characterized, basically because its estimation depends on the θ obtained in (5.8) using w^a. If at each frame the value of θ used to construct the trajectories has been saved, the following procedure can be carried out to correct the estimated z:

1. Given the smoothed trajectory coordinates x, z, use w^a and θ in each frame to obtain their theoretic detection parameters x^r, y^r, w^r using (5.3), (5.4) and (5.7).

2. Use (5.8) to estimate θ coherent with y^r, w^r and the new w value.

3. Use (5.6) to reestimate z from y^r and the new θ.

However, if instead of doing this procedure, z coordinates are corrected with the same factor used for x coordinates, the resultant trajectory is relatively close to the ideally correct one. Figure 5.23 shows the difference between the correction factor used (i.e., w/w^a), and the one that theoretically should be used for z, given different w^a inaccuracies and z values. Results show that this simplified correction method is less accurate, the smaller the z coordinates are. However, for the situation of largest error evaluated ($z = 10$ meters), notice that the error is inferior to 15 cm, what can be acceptable for many applications.

Wrong vehicle orientation hypothesis ($\varphi \neq 0$) To evaluate the impact of the violation of this assumption, a synthetic experiment has been carried out, evaluating

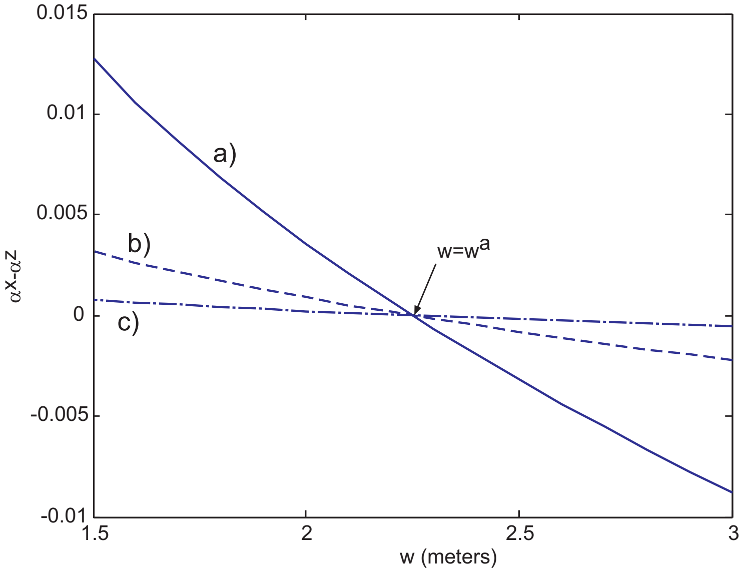


Figure 5.23: Trajectory correction factors. Difference between the factor to correct the x and z coordinates, for an assumption $w^a = 2.25$ meters and real w values in $[1.5, 3]$ meters. Plots a)–c) corresponds respectively to points at $z \in \{10, 20, 40\}$ meters.

for some selected trajectories with different angles φ, which is the trajectory recovered by assuming $\varphi = 0$. Figure 5.24 displays the ground truth trajectories versus the recovered ones, showing that in some cases the disparity between both is quite significant.

In general, it is seen that the more distant the x coordinates of trajectories from zero, the worse located the recovered trajectories. This error increases slightly with z, and is bigger for trajectories moving forward towards the road center. This asymmetrical behavior with respect to the sign of φ (i.e., the vehicle orientation) is due to the fact that depending on it, the projection of the vehicle width w^r differs more significantly from the one corresponding to $\varphi = 0$ (see Figure 5.25). This provokes that the incorrectness of (5.8) to estimate θ is bigger, and the 3D vehicle locations are worse estimated.

However, results in Figure 5.24 show that although recovered trajectories can be localized quite unaccurately, their length (i.e., the velocity module v_t) and orientation (φ) are estimated quite accurately. So advantage on that can be taken a posteriori, to correct the localisation of recovered trajectories. In order to do that, we propose the following procedure:

1. given the smoothed trajectory coordinates x, z, use w^a and θ in each frame to obtain its theoretic detection parameters x^r, y^r, w^r using (5.3), (5.4) and (5.7). Notice that θ has been estimated in the construction of the trajectory assuming $\varphi = 0$, and this was not true. Having no other way to estimate it, we take at any rate this value in the required computations;

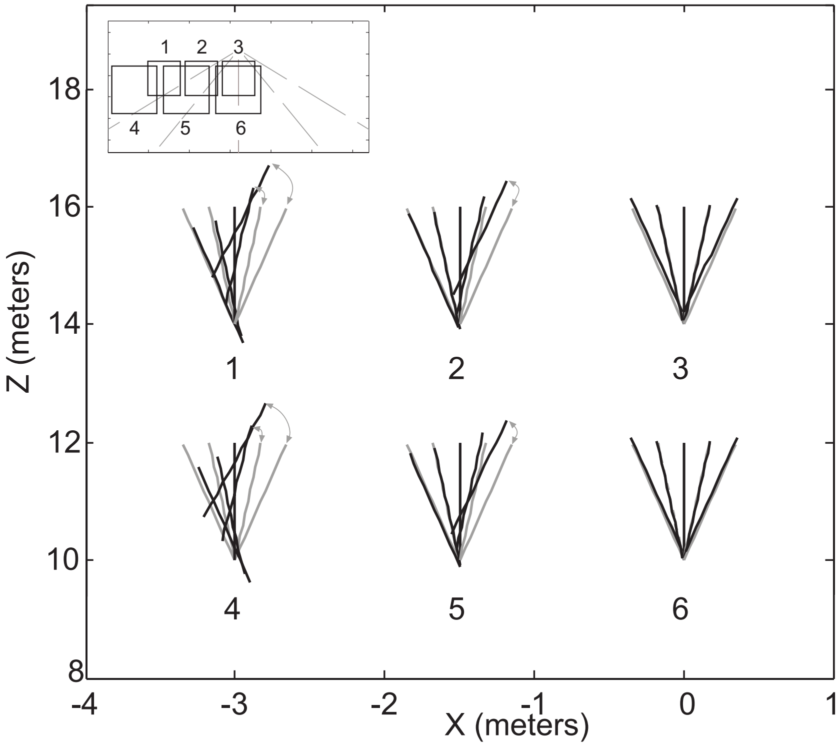


Figure 5.24: Different real trajectories studied with $\varphi \in \{-10^\circ, -5^\circ, 0^\circ, 5^\circ, 10^\circ\}$ (gray lines), and the trajectories recovered using the described method (black lines). An orientation $\varphi = \pm 10^\circ$ accounts for situations in which a vehicle performs a lane change maneouvre.

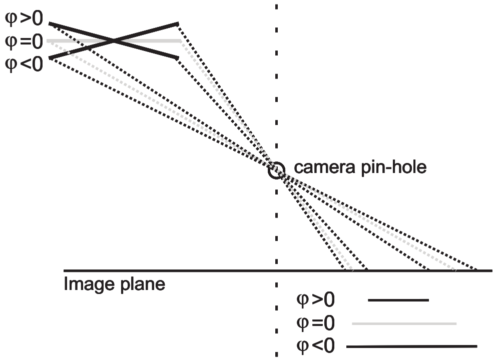


Figure 5.25: Top view of the projection of the rear of a vehicle onto the image plane. It is seen how symmetric orientations provoke different deviations of w^r with respect to $\varphi = 0$, which result in different trajectory deviations.

2. the lateral edges of the rear of a vehicle with orientation φ lay on the world coordinates given by x and $x + w^a\cos(\varphi)$. Using these coordinates, the width of the image region where the rear of a vehicle projects can be computed. Using (5.3) and after some algebraic manipulation, it is obtained that

$$w^r = x^0 - x^r + \frac{f^x(x + w^a\cos(\varphi))}{\frac{f^x x}{x^r - x^0} - w^a\cos(\theta)\sin(\varphi)} \ . \tag{5.11}$$

3. Isolating x from Equation (5.11), the x coordinate considering now the estimated vehicle orientation φ is obtained, given by

$$x = -\frac{w^a(x^0 - x^r)(f^x\cos(\varphi) + (w^r - x^0 + x^r)\cos(\theta)\sin(\varphi))}{f^x w^r} \ . \tag{5.12}$$

4. To obtain the corresponding z coordinate, just (5.12) has to be combined with (5.3), and then isolate z, obtaining

$$z = \frac{f^x w^a\cos(\varphi)\sec(\theta) + w^a(w^r - x^0 + x^r)\sin(\varphi)}{w^r} + h\tan(\theta) \ . \tag{5.13}$$

To demonstrate how the proposed procedure improves estimated trajectories, Figure 5.26 shows the result of an experiment carried out. Different trajectories with $\varphi = 10\,^\circ$ at different world coordinates are first recovered assuming $\varphi = 0$. Then, the orientation φ of computed trajectories is used to correct their (x, z) coordinates, applying (5.12) and (5.13). The accuracy improvement is very significant, specially for trajectories where φ is more precisely estimated. Figure 5.27 quantitatively evaluates the benefits of the trajectory correction in this experiment.

The methodology proposed to *relocate* estimated trajectories has a very remarkable performance, but relies on having a good φ estimation. Unfortunately, as has been shown in Figure 5.21, estimating correctly φ requires observations scarcely distorted by noise. Thus, the trajectory correction scheme will be of practical use if a low distortion of observations can be assured, and mainly for correcting trajectories near to the *host* vehicle, as in these cases the φ estimation error is small. Otherwise its application can be counterproductive.

5.3 Multiple Vehicle 3D Tracking

Having vehicles detected and their initial 3D state estimated, the next step is updating their state along time. This task could be done using the same MHT algorithm applied to estimate their initial state, but at this point, as the presence of real vehicles in frames is in some way asserted, we take advantage on that to carry out this task more efficiently, modelling more accurately the vehicle tracking problem. On the one hand, to detect vehicles, it is now available their state at the previous frame. This allows

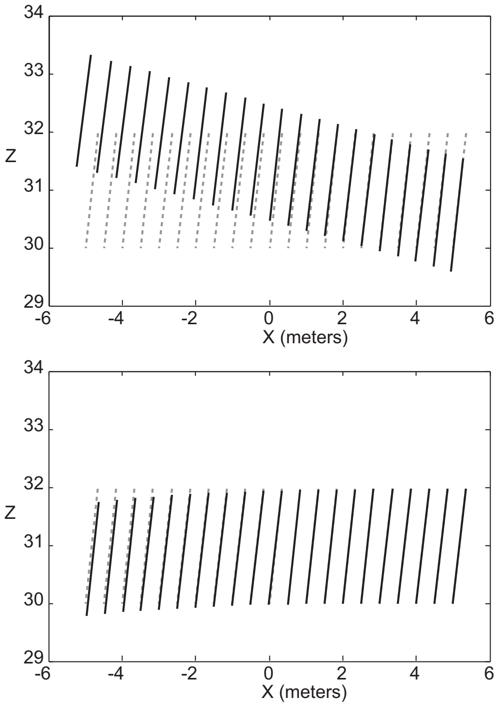


Figure 5.26: Trajectory correction results. Top: Real (dashed) and recovered (solid) trajectories for $\varphi = 0$ (solid). Bottom: Real trajectories (dashed) and trajectories corrected (solid) using the orientation angle φ of the trajectories previously computed assuming $\varphi = 0$.

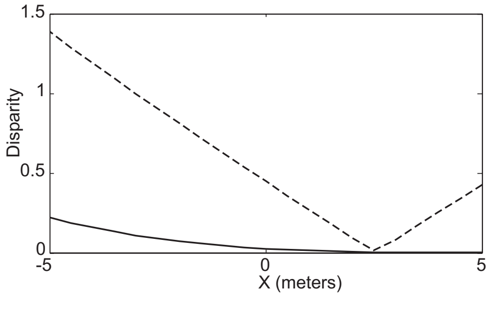


Figure 5.27: Mean disparity in meters between real and estimated trajectories, for the initially estimated trajectories (dashed) and the corrected ones (solid).

to define an specific measurement process to extract vehicle observations, localising vehicles more accurately than the general vehicle detector can do, and requiring less computational effort. On the other hand, the knowledge that real vehicles are observed in frames allows also to model more accurately the observation process. Given a sequence, the variation observed between frames will be on account of the change of position of:

- the observed vehicles (the tracked *targets*),
- the host vehicle (which changes the camera viewpoint).

To correctly model the information obtained from frames, both have to be considered in the state to be estimated. By assuming that vehicles progress following a constant velocity model, and that the road conforms to a planar surface, the movement of vehicles is represented by a velocity vector parallel to the road plane. Extending the proposal in [36] to consider multiple targets, our proposed system state $\mathbf{x}$ maintains

- the velocity describing the change of position and orientation of the host vehicle between frames $(v^h, d\varphi^h)$;
- the position (x^i, z^i) and orientation (φ^i) with respect to the host of each i-th tracked vehicle, and its corresponding velocity $(v^i, d\varphi^i)$.

So, if N vehicles are being tracked, $\mathbf{x}$ corresponds to

$$\mathbf{x} = [(v^h, d\varphi^h), (x^i, z^i, \varphi^i, v^i, d\varphi^i)_{i=1}^N]^T$$

Figure 5.28 illustrates the meaning of the state parameters in an example. Two different reference frames are used to describe the state vector parameters:

- $(v, d\varphi)$ (forward velocity and yaw rate) of host and tracked vehicles are expressed in terms of a fixed global 3D reference frame;
- the position (x^i, z^i) and orientation φ^i of targets are specified in the host coordinate system.

Using these two different reference frames, each parameter is represented in the coordinate system that allows to model its dynamical behaviour more accurately and using a simple expression. Considering $(v, d\varphi)$ under a global reference frame allows to assume that their values remain constant between frames, an assumption that would be unrealistic if a relative coordinate frame like the host coordinate system were used. On the other hand, to describe the pose of targets (x^i, z^i, φ^i) it is better to use a coordinate system relative to the host position. This is really the information of interest, and in that way it is avoided to estimate the position and orientation of the host vehicle with respect to a global coordinate system. Considering that, next section details the evolution of the system state parameters.

5.3.1 System Model

The dynamics of parameters in $\mathbf{x}$ is expressed in the following way:

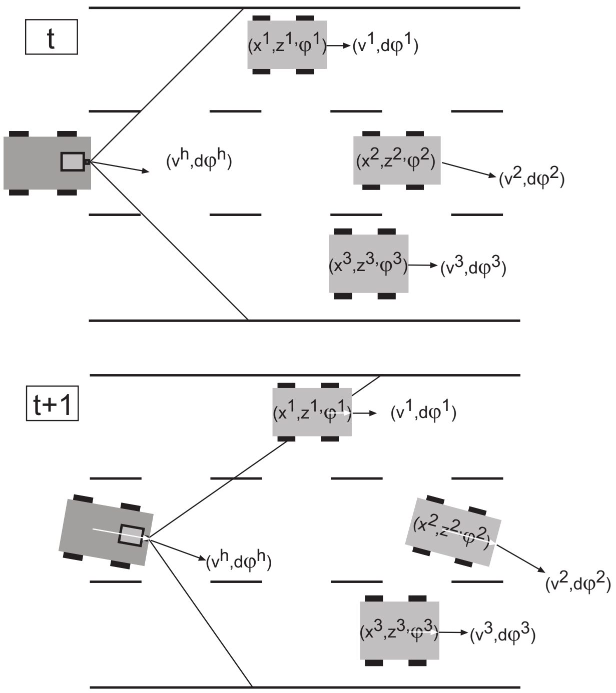


Figure 5.28: Sketch showing system parameters at consecutive time instants.

$(v, d\varphi)$ **dynamics** The behaviour along time of vehicles strongly depends on the road type where they move. In the current study, vehicles driving on motorways or high speed roads are considered. In this context the ground plane assumption holds most of the time, and the behaviour of vehicles is usually regular and smooth. So, it is expected that vehicles maintain approximately their velocities $(v, d\varphi)$ along time, experimenting just smooth changes frame–by–frame. To model this behaviour, a Constrained Brownian motion model has been used (see Appendix A), taking advantage that for the current problem constraints can be imposed on the $(v, d\varphi)$ evolution:

- the range of values that they can feasibly take;
- their expected variation between frames.

(x^i, z^i, φ^i) **dynamics** The modelisation of the relative pose of targets at each frame has been done using a deterministic linear model. Predicting the target pose at each frame requires to:

- reexpress their pose at instant $t-1$ according to the position of the host coordinate frame at instant t;
- update their pose given their current velocities.

For the i-th target, this process (illustrated in Figure 5.29) reduces to:

$$\begin{aligned} \varphi_t^i &= \varphi_{t-1}^i + d\varphi_t^i - d\varphi_t^h \ , \\ \begin{bmatrix} x_t^i \\ z_t^i \end{bmatrix} &= \mathbf{R}(d\varphi_t^h) \begin{bmatrix} x_{t-1}^i \\ z_{t-1}^i \end{bmatrix} - \begin{bmatrix} 0 \\ v_t^h \end{bmatrix} + \\ & \quad + \mathbf{R}(\varphi_t^i) \begin{bmatrix} 0 \\ v_t^i \end{bmatrix} \ , \end{aligned}$$

where $\mathbf{R}$ is a rotation matrix with respect to the Y axis of the host coordinate system.

5.3.2 Observation Model

Initially, our first approach to track vehicles was based on the 3D vehicle wire–frame approach used in works like [101, 43, 44]. Vehicles where localised by looking for image contours arranged similarly to the edges of a 3D wire–frame model projected on the image. However, due to the uncontrollability of the acquisition conditions and the low dynamic range of the available camera, in some sequences the saliency of contours in images was very poor, and a bad performance was achieved. Due to that, we finally decided to use the same model used to detect vehicles (which is more robust to these challenging situations), to also track them along time.

From the use of the vehicle detection model, it follows that the observation model has to relate the 3D state of vehicles with the 2D regions on images where these vehicles are expected to be detected. This is a non–linear relation, which results from the synthesis of the following procedure. Given the state prediction $\hat{\mathbf{x}}_{t|t-1}$, we first parameterise for each tracked vehicle a bounding box corresponding to its front/rear[14]

[14] The one visible from the camera point of view.

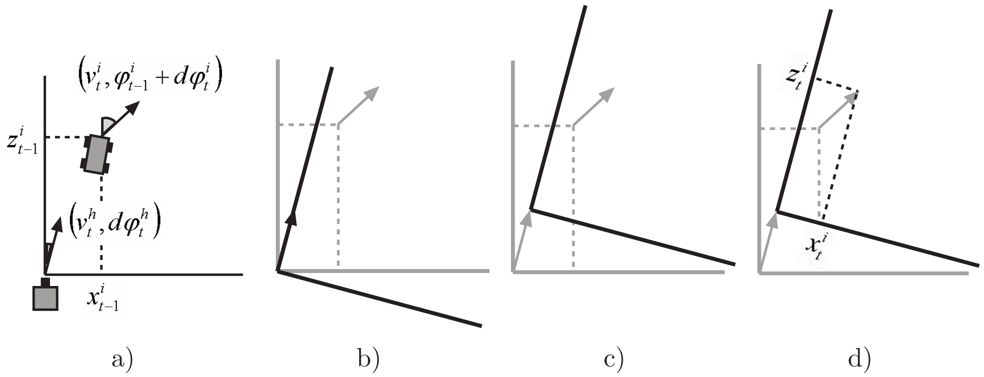


Figure 5.29: Prediction of a target new pose. a) Previous host and target state, and current velocities. b) & c) New position of the host coordinate frame. d) New coordinates of the tracked vehicle.

side. This bounding box is determined from the road location and orientation of each vehicle, and the value of its width and height estimated when the vehicle has been initialised in the tracking module[15]. Then, we project the generated bounding boxes on image coordinates, determining the predicted observation vector $\hat{\mathbf{y}}_{t|t-1}$ (i.e., the 2D regions expected to be observed in images by the measurement process). To perform this task we need to know the pitch angle θ at the current instant. We propose a procedure to estimate its value by means of an exhaustive search on the range of its possible values, determining θ as the one that projects tracked vehicles closer to the regions where they are detected.

Given the state prediction $\hat{\mathbf{x}}_{t|t-1}$, the value of its corresponding observation $\hat{\mathbf{y}}_{t|t-1}$ is computed for $\theta \in [\theta_{min}, \theta_{max}]$, which is the range of values that θ can take according to the possible states of the host suspension system. Joining the different computed observations, a group of 2D regions where to look for vehicles is obtained. This set is then augmented by the slight translation and scaling of its members, and analysed by a vehicle classifier. The classifier judges if the sub-image in each 2D region matches the appearance of a vehicle, generating as result a list of *positive* 2D regions. The positive regions of each target in $\hat{\mathbf{x}}_{t|t-1}$ are then clustered together, finally setting $\mathbf{y}_t$ with the average region of each cluster. Once $\mathbf{y}_t$ has been obtained, an iterative search procedure is started , to determine the camera pitch θ_{best} that generates the projection $\hat{\mathbf{y}}_{t|t-1}$ closer to it. This θ_{best} value is considered the current camera pitch and determines the observation model finally considered (see Figure 5.30). However, being rigorous, the estimated θ by this procedure may not correspond to the real θ, as this depends on the validity of the width and height values (w, h) assumed for each vehicle. These values are established when a vehicle is added to a tracker, using the estimation of θ at that moment. Therefore, the accuracy of the proposed method depends on the correctness of the θ considered the first time a vehicle is added to the

[15] Given the rectangular frame region where a vehicle is observed, we determine the width and height of the vehicle by backprojecting this region using the values of the camera pitch θ and the vehicle orientation φ more likely at that instant.

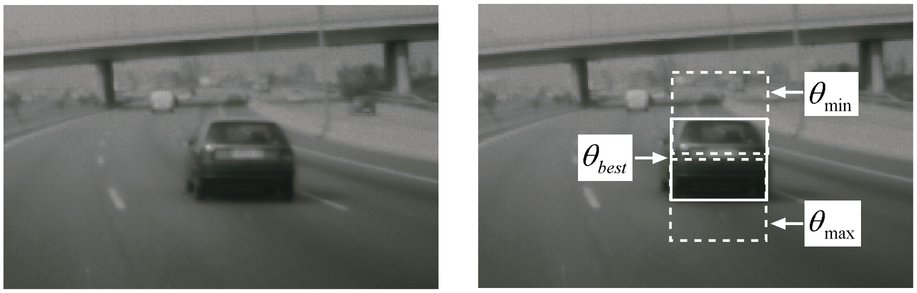


Figure 5.30: Vehicle measurement process. Dashed lines show the projection of the rear face of the vehicle, for the bound values of the θ range. The solid rectangle shows the measurement obtained, which determines θ_{best} for this frame.

tracker.

5.3.3 Estimation Algorithm

Once described the system and observation models, it is now time to address the algorithm to combine them with observations, in order to update the state of vehicles. Considering that

- we have constated experimentally that noise distorting observations can be assumed as Gaussian distributed;
- we require a method with a low computational cost;

the UKF has been found the more appropriate method to perform this task. At the end of each estimation cycle, the state provided by this algorithm is checked to control if the locations of tracked vehicles reflect some of the following events:

- a vehicle is being occluded by another target;
- a vehicle has moved outside the field of view of the acquisition system;
- a vehicle does not receive observations during a given period of time (missdetection situations);
- a vehicle moves too far away from the camera for being properly tracked (i.e., it is more than 70 meters away).

In any of these situations, target involved are eliminated from $\mathbf{x}$, as they are no longer observable.

A point to be remarked is that, as the current system uniquely relies on the location of vehicles in images, the state in $\mathbf{x}$ can only be trusted in relative terms. That is, any situation of a host vehicle with $v^h = n$ km/h observing a vehicle with $v^1 = n + 20$ km/h is identical in terms of their estimation, whichever the value of n is. Thus, the value estimated of v^h is of use just relatively to v^1. To avoid this problem, additional sensors in the host vehicle could provide information about its egomotion.

In the experimental work done, a qualitatively robust performance of the vehicle tracking module is observed. A quantitative evaluation of its performance has not been possible, because unfortunately we lack of ground truth information providing the location of vehicles in testing sequences, as well as the egomotion information.

5.4 System Modules Cooperation

Finally, we describe how all proposed modules cooperate to implement the whole tracking system. Providing a detailed explanation of the system coordination would be considerably extensive, requiring to provide details on implementation decision taken that may obscure the essential ideas. For this reason, just an schematic view of the overall system is provided. Figure 5.31 details the flow of acquired images through the system modules for the cases in which the system is currently tracking vehicles or not.

Results obtained from the cooperation of all the modules are encouraging, showing a reliable performance in the simultaneous tracking of different types of vehicles, even when acquisition conditions are challenging. However, as previously remarked, this performance can only be evaluated at this moment qualitatively. Figure 5.32 gives an example of the performance of the system in a short testing sequence.

5.5 Conclusions

This chapter has focused on the design and implementation of a complete target tracking system, devoted to detect and track vehicles observed from a camera mounted in a mobile platform. A modular approach has been followed, integrated by:

- a vehicle detector, to identify the presence of cars, vans and trucks in frames;
- a vehicle 3D state initialiser, to estimate the 3D location and velocity of detected vehicles, discarding at the same time false detections;
- a vehicle tracker, to estimate efficiently their 3D state along a sequence.

We have designed the vehicle detection module following a classifier–based approach. Using the Adaboost algorithm, first we have trained a COC from training data. These classifiers are used to detect vehicles on frames, by evaluating them at different images regions determined using the projective geometry of the acquisition system. As the amount of regions to be classified is huge, two proposals has been done to increase the efficiency in the COC evaluation: an optimisation of the first COC learned, and its lazy evaluation. Thanks to this two proposals, the number of image features to be computed per evaluated region has been reduced around a 58%. The performance of the proposed detection system has been analysed on testing frames, correctly identifying around the 94% of detectable vehicles, generating on average a false positive detection per frame. This high false positive rate does not suppose a problem to the system developed, as false positives usually do not present spatio–temporal coherence along time, and this allows to identify and discard them posteriorly.

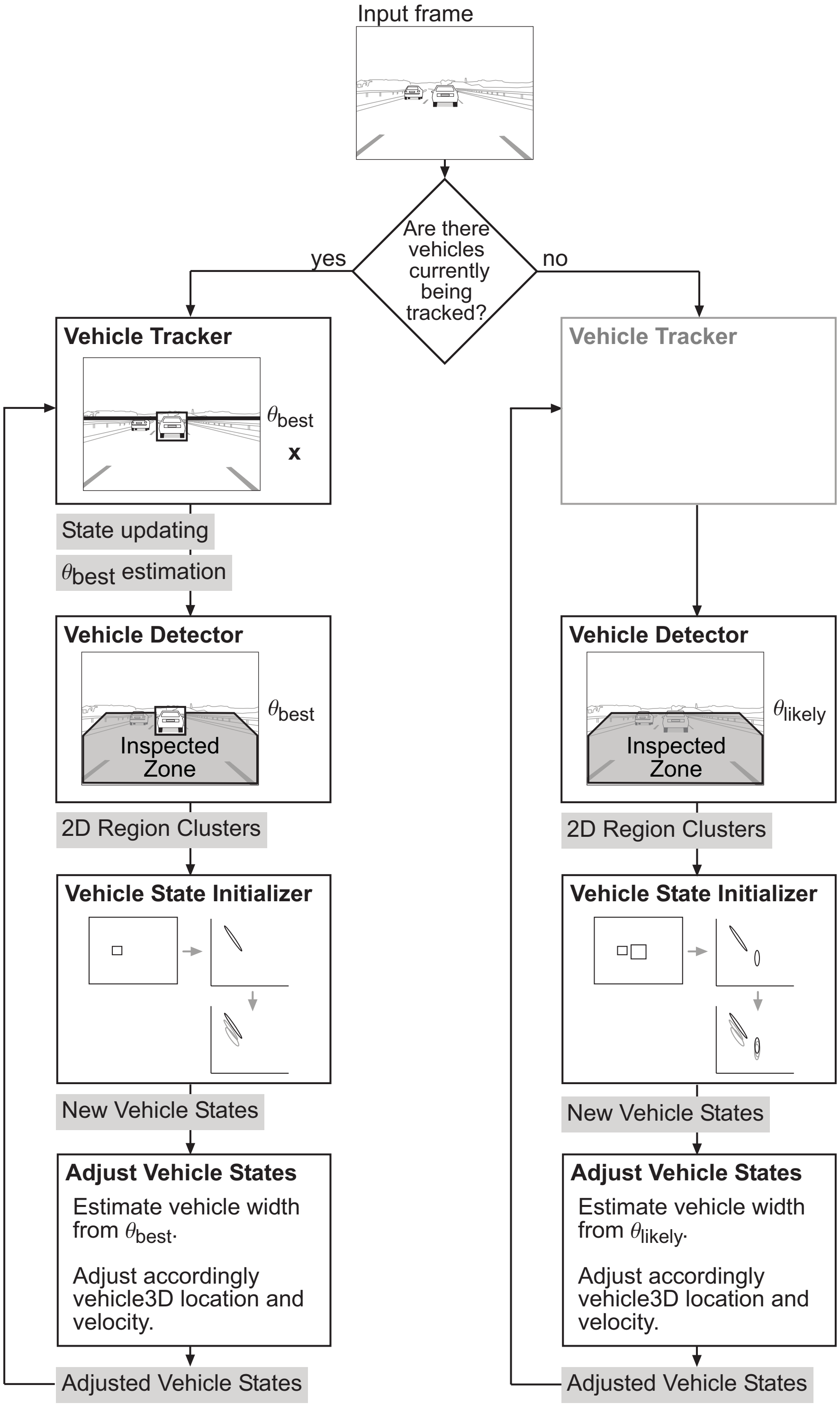


Figure 5.31: Cooperation between the vehicle tracking system modules, when there are vehicles being tracked (left) or not (right).

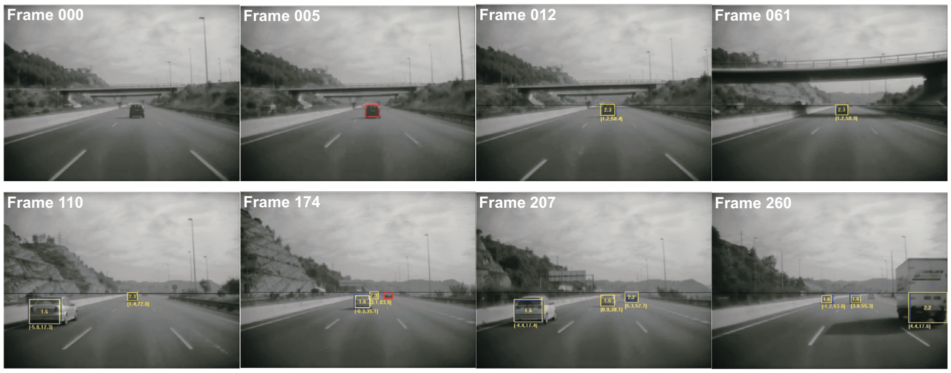


Figure 5.32: Detection–tracking procedure. Frame 000 is the first frame of the sequence. After several consecutive detections (frame 005), a target is added to the tracking module (frame 012). Frame 061 shows that the tracker is robust to sudden changes in the lightening conditions. At frame 110 a new target is added to the tracking module. Frame 174 shows the situation previous to the elimination of one target, due to the occlusion it suffers. Also it is shown how a distant truck is detected. Frame 207 shows the simultaneous tracking of two cars and one truck. Frame 260 shows the situation previous to the elimination of one target that leaves the camera field of view.

Our proposal to estimate the initial 3D state of a detected vehicle is based on a method that constructs vehicle trajectories from series of successive vehicle detections. First, we propose to estimate the 3D road locations corresponding to detections by combining a model of projective geometry, the unscented transform, and assumptions on the width and orientation of detected vehicles. With respect to other approaches, the advantage of our proposal is that

- we estimate a whole distribution of the feasible vehicle locations;
- the error in the estimated vehicle location, derived from the violation of assumptions done, is systematic.

Having just a systematic error is very interesting because the spatio–temporal coherence of successive detections is maintained, and this makes easier the task of estimating vehicle trajectories. Moreover, a systematic error is always easier to correct. In our proposal, the localisation errors are mainly due to the use of wrong hypothesis of the vehicle width and orientation. Conscious of that, we provide with expressions to correct locations if more accurate assumptions on this values are later available.

In order to construct trajectories from successive vehicle locations, we propose a multiple hypothesis tracking algorithm based on the KF, using the GNN to associate locations between frames. Generated trajectories allow to solve two different tasks: the verification of the detection of real vehicles (false positives are spurious and do not allow to construct coherent trajectories), and the characterisation of their dynamic behaviour. The performance of the proposed methodology has been estimated using synthetic data, in order to identify how the different sources of error threating the proposal affect the results achieved. Results show that trajectories of real vehicles are recovered, even in the presence of significant noise and missdetection events. Hence, the task of verifying the presence of vehicles and discard false detections is correctly fulfilled. With respect to the 3D localisation of trajectories, accuracy decreases when the noise disturbing observations and the amount of missdetections increases. Concerning the estimation of the 3D vehicle velocity, the accuracy depends on the overlap between the 3D road regions that conform a trajectory, which is bigger the more distant the vehicles are.

The information of constructed trajectories is used to initialise the state of a vehicle tracker. Our vehicle tracking module is integrated by a multiple target tracking algorithm based on the UKF, and a supervision process that control when tracked targets are no longer of interest. The state estimated along time includes parameters concerning tracked targets, as well as the host vehicle holding the camera. In that way, observations extracted from frames are more accurately interpreted. Results achieved in testing sequences are satisfying in qualitative terms. A quantitative performance evaluation has not been possible yet, due to the lack of ground truth information in testing sequences.

Chapter 6

Conclusions and Future Work

In this thesis we have mainly focused on the application of estimation techniques to the problem of quantitative analysis of video sequences. In particular, our work has concentrated on the study of two different application contexts: contour tracking, and vehicle detection and tracking. We have studied different problems concerning these topics, and made contributions to each of them.

6.1 Contour Tracking

Our first task developed in this part of the thesis has been the review of the contour tracking state of the art, following the model–based approach proposed in the Active Contours formalism. In particular, different approaches to extract contour observations from frames have been reviewed, as well as Kalman–based and Particle–based algorithms to track contours from these observations. An in-depth study of the performance achieved by the different proposals has been done. Performance has been quantified in terms of the overlap between the tracked object and their estimated shape, as well as in terms of the point–to–point disparity between the estimated contour and an ideal ground truth contour. Results obtained allow to compare the accuracy of tracking algorithm in different noise situations, providing insight about the implementation issues that lead to their best performance.

Regarding the measurement process, Kalman–based and Particle–based algorithms have a different behaviour. The performance observed in KFs shows that:

- the adaptive control of the length of contour measurement lines is a very relevant issue in order to achieve a good tracking performance. With respect to the fixed length approach, it minimises the disturbance that noise artifacts provoke in the measurement process;

- the best tracking performances are achieved using measurement lines normal to the predicted contour. With this result, we refute the conclusions in [10]. We claim that this is due to the method to establish the length of measurement lines used in that work, which is inappropriate when the normal direction is used.

On the other hand, the performance observed in PFs[1] regarding the measurement process lead to the following conclusions:

- there is no advantage on making and adaptive control of the length of measurement lines;
- determining the orientation of measurement lines according to the method in [10] improves the performance in noisy situations, but at a high cost. In practice is more productive just considering a higher number of particles using normal measurement lines.

A comparative study of the different estimation methods evaluated has allowed us to reach the following conclusions:

- the well–known superiority of PFs with respect to Kalman–based algorithm is corroborated specially in high noisy situations. However, in low–noise sequences, they require a big amount of particles to match their accuracy; Hence, in this cases a Kalman–based algorithm is preferred.
- the *standard* PF implementation (i.e., Condensation) is very sensitive to a bad modelisation of the target dynamics, while Kalman–based approaches are notably robust to that;
- in non–linear contour tracking problems, the performance achieved by the proposed adaptations of the EKF and the UKF is practically the same;
- in presence of noise artifacts correlated in time[2], the performance of Kalman–based algorithms degrades very significantly, while PFs basically present the same performance no matters whether artifacts are correlated or not.

The next topic of research studied in this part of the thesis has been the improvement of the performance of PF in general contour tracking applications. This applications require the estimation of rigid and non–rigid transformations of a contour, which commonly implies estimating a high dimensional contour state. This challenges significantly the performance of standard PFs. With the aim to improve their performance in this cases, we have novelty adapted three different techniques to the contour tracking problem: the UKPF, the RBPF, and the PS technique.

The UKPF is an algorithm that tries to improve tracking performance of PFs by estimating, at each time step, a linear approximation of the optimal importance

[1] Experiments have been done using the Condensation algorithm.

[2] This corresponds, for instance, to situations where the target of interest is occluded by other objects.

function used to propagate samples. Our aim in using this technique has been to reduce the need of a major number of particles by propagating them *better* (i.e., closer to the ground truth state of the tracked contour). Results achieved are noteworthy:

- with a reduced number of particles this algorithm overcomes the performance of Kalman–based algorithms, even in low–noise situations. To achieve the same result with the usual PF implementation, a very big amount of particles is required;
- with respect to the classical PF implementation, a significative better performance is achieved in low noise situations, which degrades as long as the noise increases. Just in extremely noisy sequences, the performance achieved is inferior to the one of the classical PF. In this case, the approximation of the optimal importance function done by the UKPF is completely distorted, and using it to propagate samples is worse than using the dynamical model of the system with this purpose.

The RBPF is a technique that deals with the curse of dimensionality of PFs by estimating one part of the state analytically (i.e., using Kalman expression), and the other part by means of particles. As state partitions have compulsorily a dimension inferior to the one of the original state, the demand on the number of particles decreases. The adaptation of this technique to the contour tracking problem has revealed the following performance:

- in low–noise situations this technique successfully overcomes the classical PF performance. However, it performs worse than an UKF. Results conclude that in these situations is better to estimate the whole state analytically, instead of just part of it;
- as sequences present more noise artifacts, the RBPF turns to overcome the performance of the UKF, but underperforms the one achieved by classical PFs. This reflects the problem of the Kalman estimation equations when the noise disturbing observations is mainly non–Gaussian[3]. In this situation, estimating the whole or part of the state analytically is bad posed, and the best option is estimating the whole contour state using particles.

We conclude from results that in the context of contour tracking, the RBPF *averages* the pros and cons of Kalman and Particle–based filters at each noisy situations. Hence, if an specific contour tracking application has a very well defined noise situation, it will not be clearly the best choice. On the other hand, in an application where images could alternate high and low noise situations, we think that it would be possible that the average performance of the RBPF could improve the one achieved by the UKF and the standard PF.

[3] Remember that the Kalman equations are derived assuming observations perturbed by Gaussian noise.

Our third proposal to improve the contour tracking performance of PFs is based on the PS technique, which provides a well–founded to implement our following premonition: if it were possible to estimate the rigid transformation of a contour independently from its local deformations, this estimation would provide very valuable to improve the estimation of the whole contour state. Hence, first we have proposed a novel method to estimate contour rigid transformations in spite of the local deformations that can be present. After quantitatively evaluating this proposal, which has proven to be reliable in different noisy situations, we have used this method to implement the reweighted sampling step of a PS algorithm. The resultant contour tracker has shown the following performance:

- it overcomes the performance of UKFs, except in low noise sequences, where is still better to estimate the contour state analytically;
- with respect to the classical PF implementation, a better tracking performance is achieved in all situations. However, in highly noisy sequences, the performance of both methods is very similar.

From the three proposals done, we observe that the proposal more regular in improving the PF performance is the PS algorithm, as overcomes the PF in all the noise situations. However, in noise situations with a SNR equal or higher than 8 db, the UKPF is clearly the best performing method, overcoming even the performance of the UKF. Thus, in many practical application this should be the chosen method.

Our last contribution regarding contour tracking has also been devoted to address the curse of dimensionality of PFs, but from a completely different perspective. Instead of proposing alternative algorithms to deal more robustly with high–dimensional states, now we propose a method to directly reduce the dimension of the state to be estimated. Our proposal is replacing the shape model used by a tracking algorithm, by means of a collection of models of lower dimensionality. In that way, for any shape model being active at each time step, the contour state will have a lower dimensionality. To make this idea operative, novel contributions have been done in the following topics:

- the unsupervised characterisation of the SVS of a given shape model;
- the generation of multiple shape models, using the model of the SVS constraining a given shape model;
- the estimation of the contour of an object using the multiple models generated.

Concerning the first topic, we have developed a novel algorithm to model the SVS of a given shape model using a GMM. Our proposal is based on embedding the EM algorithm inside a divisive hierarchical clustering technique. Using a greedy search strategy, we have achieved a performance comparable to the one obtained using an exhaustive search strategy. Our proposal is very appealing to be used in practical applications, as no parameters has to be set by the user.

Using the GMM delimiting the SVS of a given shape model, we have proposed an original method to generate a collection of multiple models from it. Generated models represent training shapes more accurately than the original *global* shape model, requiring in overall an inferior number of parameters. The validity of the proposed multiple model approach has been tested in the context of modelling the silhouette of a walking pedestrian. In terms of the BIC, it has been constated that our proposal provides a better shape modelisation than the one obtained using a single model approach.

Finally, we have proposed a novel algorithm to track contours using the generated multiple shape models. First, the dynamics of each model, as well as the interaction between them, has been modelled using a JMS. Then, an algorithm based on the PS technique has been presented. We have evaluated the performance of our algorithm experimentally, showing that in all noise situations, the multiple shape models approach overcomes the performance achieved by the traditional single shape model approach.

6.2 Vehicle Detection and Tracking

This part of the thesis has been devoted to the development of a complete vehicle tracking system, with the aim to extract information of interest for ADAS applications. Our goal has been developing a system to detect and track vehicles observed from a camera mounted in a mobile platform. A modular approach has been followed, integrated by a vehicle detector, a vehicle 3D state initialiser, and a vehicle tracker.

We have designed the vehicle detection module following a classifier–based approach. Using the Adaboost algorithm, first we have trained a COC from training data. These classifiers are used to detect vehicles on frames, by evaluating them at different images regions determined using the projective geometry of the acquisition system. As the amount of regions to be classified is huge, we have contributed with two proposals to increase the efficiency in the COC evaluation, namely a COC optimisation, and its lazy evaluation, reducing around a 58% the number of image features to be computed per evaluated region. The proposed detection system detects the 94% of the *detectable* vehicles in a set of testing frames, generating on average a false positive detection per frame. This false positive rate does not suppose a problem to the system developed, as false positives usually are uncorrelated, and this allows to identify and discard them posteriorly.

Our proposal to estimate the initial 3D state of a detected vehicle is based on a method that constructs vehicle trajectories from series of successive vehicle detections. First, we have proposed a novel method to estimate the 3D road locations corresponding to the positive image regions returned by the vehicle detector. With respect to other approaches, the advantage of our proposal is that

- we estimate a whole distribution of the feasible vehicle locations;
- the error in the estimated vehicle location, derived from the violation of as-

sumptions done, is systematic.

Having just a systematic error is very interesting because the spatio–temporal coherence of successive detections is maintained, and this makes easier the task of estimating vehicle trajectories. In our proposal, the localisation errors are mainly due to incorrect hypothesis done on the width and orientation of the detected vehicle. Conscious of that, we have provided with expressions to correct locations if more accurate assumptions on this values are later available.

In order to construct trajectories from successive vehicle locations, we have proposed a multiple hypothesis tracking algorithm based on the KF, using the GNN to associate locations between frames. Generated trajectories allow to solve two different tasks: the verification of the detection of real vehicles (false positives are spurious and do not allow to construct coherent trajectories), and the characterisation of their dynamic behaviour. The performance of the proposed methodology has been estimated using synthetic data. Results show that trajectories of real vehicles are recovered, even in the presence of significant noise and missdetection events. Hence, the task of verifying the presence of vehicles and discard false detections is correctly fulfilled. With respect to the 3D localisation of trajectories, accuracy decreases when the noise disturbing observations and the amount of missdetections increases. Concerning the estimation of the 3D vehicle velocity, the accuracy depends on the overlap between the 3D road regions that conform a trajectory, which is bigger for the more distant vehicles.

From the information collected in the constructed trajectories, we have proposed a method to establish the initial 3D state of vehicles, to be provided to the vehicle tracking module. The proposed vehicle tracking module is integrated by a multiple target tracking algorithm based on the UKF, and a supervision process that control when tracked targets are no longer of interest. The proposed approach is based on an extension of the formulation in [36] devoted to manage multiple targets. Results achieved in testing sequences are satisfying in qualitative terms. A quantitative performance evaluation has not been possible yet, due to the lack of ground truth information in testing sequences.

6.3 Future Research Directions

During the work developed in this thesis, different possible lines of continuation have been identified:

Concerning the work developed in contour tracking, we claim that the proposed multiple shape model tracking approach can be further extended:

- with respect to the generation of multiple shape models, and the description of their dynamics, alternative algorithms could be considered (for instance, based on auto–regressive HMMs [92]);
- in the use of multiple models for contour tracking, other estimation algorithms

could be also considered. For instance the RBPF could be used, estimating the distribution of the discrete variable identifying the active model by means of particles, and the distribution of the parameters of each model using a KF.

With respect to the proposed vehicle tracking system, next steps could be done in the following directions:

- acquiring a collection of testing sequences synchronised with the output of other sensors installed in the host vehicle (accelerometers, radar, lidar, stereo cameras, etc.). These sensors will provide additional information on acquired frames, that could be used as ground truth to evaluate quantitatively the performance of the different modules of the system.

- improving the vehicle detection module, by improving the performance of the classifier used to detect vehicles. Concerning that point, there are a lot of complementary alternatives to study:

  - using a different machine learning technique to construct the classifier (SVMs, NNs, etc.);
  - using different sorts of features to describe vehicles;
  - turning the proposed binary classification problem (i.e., vehicle/non–vehicle) to a multiclass classification problem (i.e., car/van/truck/road/guard rail/etc.);
  - learning a multipart vehicle model, in order to gain robustness to occlusions, glitters and shadows, taking advantage of the spatial relationship between the considered vehicle parts;
  - training different classifiers optimised to detect vehicles at specific ranges of distances.

- adding an egomotion estimation module to the current system, which will provide valuable information to increase the accuracy on the 3D information extracted from sequences. For instance, we could start by checking the egomotion estimator proposed in [107], as authors claim to achieve a remarkable robust performance.

- using alternative vehicle models in the vehicle tracking module. Different topics may be considered:

  - using appearance models generated on–line for detected vehicles, having in that way an ad–hoc model for each vehicle being tracked;
  - using 3D wire–frame vehicle models, adapting them to vehicle contours in order to better estimate their 3D pose. For a robust performance of this approach, images should be acquired using a camera with a high dynamic range, in order to guarantee the saliency of vehicle contours even in challenging illumination conditions.

Appendix A

Models of Dynamics

The dynamics of many physical processes can be approximated by first, second and sometimes higher–order ordinary linear differential equations in time domain. A general expression of a second order differential equation is

$$a_2 \frac{\partial^2 x(t)}{\partial t^2} + a_1 \frac{\partial x(t)}{\partial t} + a_0 x(t) \quad = \quad z(t) \ , \tag{A.1}$$

where $x(t)$ is the modelled function, and $z(t)$ some external forcing function, independent of $x(t)$. Although this models are very usual in general tracking applications, concerning visual tracking is more common the use of models based on discrete time series. For a second order motion model, expressions of the following form are used

$$x_t \quad = \quad \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \xi_t \ , \tag{A.2}$$

which describes the value of a function at instant t (denoted as x_t) by means of a weighted sum of its previous values. This expression can be easily related to (A.1) through time discretisation. For instance, applying the Euler method on (A.1) with a discretisation constant τ, parameters in (A.2) are determined as

$$\begin{aligned} \alpha_1 &= \frac{2a_2 + a_1\tau}{a_0 + a_1\tau + a_2\tau^2} \ , \\ \alpha_2 &= \frac{-a_2}{a_0 + a_1\tau + a_2\tau^2} \ , \\ \xi_t &= \frac{1}{a_0 + a_1\tau + a_2\tau^2} z_t \ . \end{aligned}$$

If the forcing function ξ_t is a Gaussian white noise process, then (A.2) defines an auto-regressive process of second order (AR(2)). These kind of processes are also known as Markov Processes, and describe a stochastic dynamic process using expressions of the form

$$x_t \quad = \quad v + \sum_{k=1}^{n} \alpha_k x_{t-k} + b_0 w_t \ , \tag{A.3}$$

where v and α_k are real constants, being $\alpha_k \neq 0$, and $b_0 w_t$ corresponds to a Gaussian white noise process with parameters $\mathcal{N}(0, b_0 b_0^T)$.

For mathematical and notation convenience, dynamics are commonly expressed in the vector–matrix form proposed by the state–space notation. For instance, (A.3) when $v = 0$ is expressed as

$$\begin{bmatrix} x_t \\ x_{t-1} \\ x_{t-2} \\ \vdots \\ x_{t-(n-1)} \end{bmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} & \alpha_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ x_{t-2} \\ x_{t-3} \\ \vdots \\ x_{t-n} \end{bmatrix} + \begin{bmatrix} b_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} w_t \tag{A.4}$$

This is called the companion form of Equation (A.3), and it describes the n-th order dynamics of x_t as a first–order Markov model in state–space. Usually it is expressed more compactly as follows

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{w}_t \ , \tag{A.5}$$

where terms equals one to one with vectors and matrices in Equation (2.9), except for the part of the stochastic error term ξ_t. Here it is alternatively expressed using a $n \times n$ $\mathbf{B}$ matrix given by

$$\mathbf{B} = \begin{bmatrix} b_0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} ,$$

and a $n \times 1$ Gaussian white noise vector $\mathbf{w}_t$.

A third parameter $\bar{\mathbf{x}}$ can be added to fix a desired *mean* dynamical behaviour, leading to the following final expression of dynamics

$$\mathbf{x}_t - \bar{\mathbf{x}} = \mathbf{A}(\mathbf{x}_{t-1} - \bar{\mathbf{x}}) + \mathbf{B}\mathbf{w}_t \ .$$

There exist several proposals to establish the dynamic parameters $\mathbf{A}$, $\mathbf{B}$, $\bar{\mathbf{x}}$ of AR processes from training sequences. Relevant works are the ones in [93, 123, 89]. Although learning techniques provide quite accurate descriptions of the behaviour observed in training sequences, their use in practical applications may require providing very long and complete training examples. Otherwise, a model too specific to the given training data is obtained, which is obviously counterproductive. In general, in many applications the a priori knowledge on the dynamics of a process is quite loose, as its evolution can perform a wide spectra of variations. In this cases, with a single model only a very general description of this behaviour is possible, and an operative way to model dynamics is selecting a generic model from available a priori knowledge, and then use training sequences for a fine tuning of parameters involved. Next section describes generic models typically used in visual tracking, and how to parameterise them to obtain a desired constrained behaviour. Expressions developed assume $\bar{\mathbf{x}} = \mathbf{0}$, but the term $\bar{\mathbf{x}}$ can be easily incorporated by replacing in the expressions $\mathbf{x}_t$ by $(\mathbf{x}_t - \bar{\mathbf{x}})$.

A.1 AR(1) Processes

An AR(1) process is a first-order process, meaning that only the immediately previous state value has a direct effect on the current value. For modelling the evolution along time of a single parameter c, the terms in equation (A.5) will correspond to

$$\begin{aligned}\mathbf{x}_t &= [c_t] \ , \\ \mathbf{A} &= [\alpha] \ , \\ \mathbf{B} &= [b_0] \ .\end{aligned}$$

A typical form of AR(1) (known as *Brownian motion* (BM)) corresponds to setting $\alpha = 1$. It models the assumption that c_t maintain the same value than in the previous instant, except for a given disturbance $\mathcal{N}(\mathbf{0}, \mathbf{B}\mathbf{B}^T)$. This displays a random evolution of $\mathbf{x}_t$, which mimics for example the one observed in microscopic particles immersed in a fluid, or less scientifically, in the walk of a very drunken person. $\mathbf{x}_t$ evolves unbounded, what is a problem in practice, as parameters are usually meaningful inside a given range of feasible values. Considering that, what can be useful in practice is a Markov process showing the random–walk behaviour on a short–time scale, but constrained in a given subspace. For the cases where a Gaussian envelope is a satisfying approximation of this subspace, this can be achieved by just using $0 < \alpha < 1$, modelling what is known as *Constrained Brownian Motion* (CBM). If $\alpha^2 = 1 - \epsilon$ with $0 < \epsilon \ll 1$ then, on a small time–scale, the dynamic process is almost indistinguishable from the case $\alpha = 1$. However, in the long term can be seen that the $\mathbf{x}_t$ is constrained in a Gaussian envelope given by $\mathcal{N}(0, \frac{1}{\epsilon}\mathbf{B}\mathbf{B}^T)$ (see chapter 9 in [18]). Figure A.1 display the evolution of a system following respectively BM and CBM. Under same initial conditions and equivalent disturbances, it is clear than CBM maintains the evolution of the process in a range determined by a Gaussian envelope.

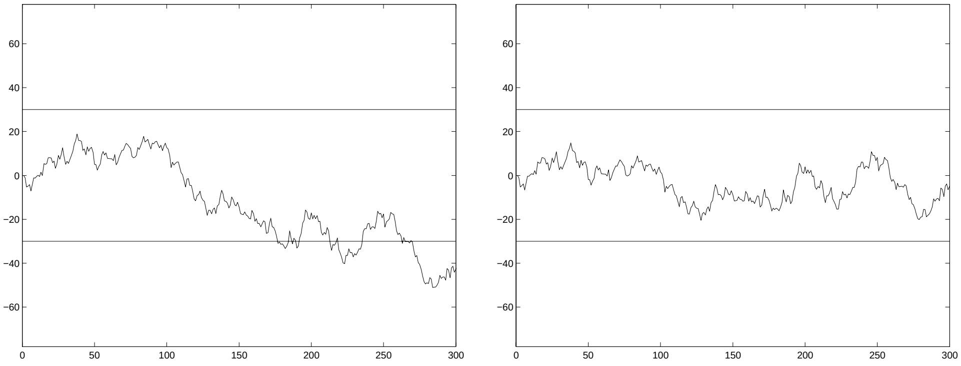


Figure A.1: Evolution of a process following BM (left) or CBM(right). Horizontal lines mark the constrained imposed in the CBM.

A.1.1 Parameterising a Constrained Brownian Motion

In this section is detailed an original proposal to establish the parameters of a CBM to obtain a desired behaviour. As described previously, a CBM constraints values in a Gaussian envelope given by $(0, \frac{1}{\epsilon} b_0 b_0^T)$, being ϵ a parameter that determines the AR(1) process by the relation $\alpha^2 = 1 - \epsilon$. Given the AR(1) of a parameter x_t as

$$x_t = \alpha x_{t-1} + b_0 w_t \tag{A.6}$$

the idea is take profit of the properties of Gaussian distributions to establish parameters (α, b_0) in a way that

- C1: constraints x_t in a desired range $[-l, l]$,
- C2: forces the term $b_0 w_t$ to take an average magnitude equivalent to m (the most likely disturbance expected).

This is achieved in the following way. It is well known that a random variable with Normal distribution $\mathcal{N}(0, \sigma^2)$ is constrained with a 99.73% probability in the range $[-3\sigma, 3\sigma]$. From this property, the Gaussian envelope in a CBM establishes consequently a range of more likely x_t values, that it can be adjusted to have desired bound values. Thus, a CBM that fulfils constraint C1 requires that

$$l = 3\frac{b_0}{\sqrt{\epsilon}}. \tag{A.7}$$

Another property of random variables with Normal distribution $\mathcal{N}(0, \sigma^2)$ is that the expectation of their absolute value corresponds to $\sigma\sqrt{2/\pi}$. As from expression A.6 the disturbance term $b_0 w_t$ follows a distribution $\mathcal{N}(0, b_0 b_0^T)$, this property can be used to tune the CBM to fulfil C2. This requires a b_0 value given by

$$b_0 = \sqrt{\frac{\pi}{2}} m. \tag{A.8}$$

Thus, the more likely disturbance expected m determines the value of b_0. Once b_0 is established, then combining (A.7) and (A.8) ϵ is determined as

$$\epsilon \ = \ \frac{9\pi}{2}\left(\frac{m}{l}\right)^2 \ .$$

and the parameter α of the desired AR(1) is obtained as $\alpha = \sqrt{1 - \epsilon}$.

A.2 AR(2) Processes

Second–order Markov Processes are popularly used because model dynamic behaviours less random than the ones of AR(1). For modelling the evolution of a model parameter c, the terms in equation (A.5) will correspond to

$$\mathbf{x}_t = \left[\begin{array}{c} c_t \\ c_{t-1} \end{array}\right] , \tag{A.9}$$

$$\mathbf{A} = \left[\begin{array}{cc} \alpha_1 & \alpha_2 \\ 1 & 0 \end{array}\right] , \tag{A.10}$$

$$\mathbf{B} = \left[\begin{array}{cc} b_0 & 0 \\ 0 & 0 \end{array}\right] . \tag{A.11}$$

A typical AR(2) used for translational motion is the *Constant Velocity* model, obtained with $\alpha_1 = 2$ and $\alpha_2 = -1$. However, with this model the value of $\mathbf{x}_t$ evolves unbounded for any $b_0 > 0$, which as explained before is undesirable. There exist however alternative parameterisations of $\mathbf{A}$ that effectively impose value constraints. A usual approach is establish α_1 and α_2 to mimic the behaviour of an harmonic oscillator. In differential equation terms, an harmonic oscillator is expressed as

$$\frac{\partial^2 x(t)}{\partial t^2} + 2\beta\frac{\partial x(t)}{\partial t} + w_0^2 x(t) = z(t) , \tag{A.12}$$

where β and w_0 are respectively the damping constant and the natural frequency of the oscillator. In terms of an AR(2), (A.12) is reproduced using the following factors

$$\begin{aligned} \alpha 1 &= 2\exp(-\beta\tau)\cos(-iw_d\tau) , \\ \alpha 2 &= -\exp(-2\beta\tau) , \end{aligned}$$

where $w_d = \sqrt{\beta^2 - w_0^2}$ is usually denoted as the damped frequency of the oscillator. Depending on the value of β and w_0 and the relation between them, b_t evolves according to the typology shown in figure A.2.

The usefulness of an harmonic oscillator model is quite obvious for objects presenting a periodic movement (for example, the outline of a beating heart). w_d establishes the desired frequency of oscillation, while β controls the magnitude of the reaction to perturbations . The value of b_0 acts as an scale factor of the whole process evolution, and determines the amplitude of the range of values that x_t may take (see figures A.3 and A.4). It can be shown (chapter 9 in [18]) that this range is delimited by a Gaussian envelope given by $\mathcal{N}(0, \frac{1}{\epsilon} b_0 b_0^T)$, where

$$\epsilon = 1 - \alpha_2^2 - \alpha_1^2 - 2\frac{\alpha_2\alpha_1^2}{1-\alpha_2} . \tag{A.13}$$

By establishing $w_d = 0$ it is possible to model constrained non–oscillatory behaviours, which are more useful for practical problems than oscillations. Next section details our proposal to parameterise this behaviours imposing a desired constrain.

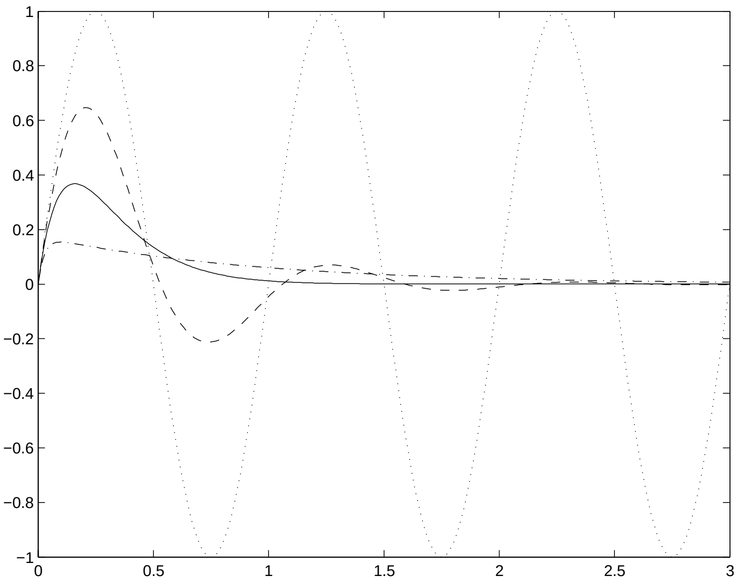


Figure A.2: Response of an Harmonic Oscillator to an impulse signal (i.e. $z(t) = \delta(t)$ in (A.12)). Dotted: undamped ($\beta = 0$). Dashed: under-damped ($\beta < w_0$). Solid: critically damped ($\beta = w_0$). Dash–dotted: over-damped ($\beta > w_0$).

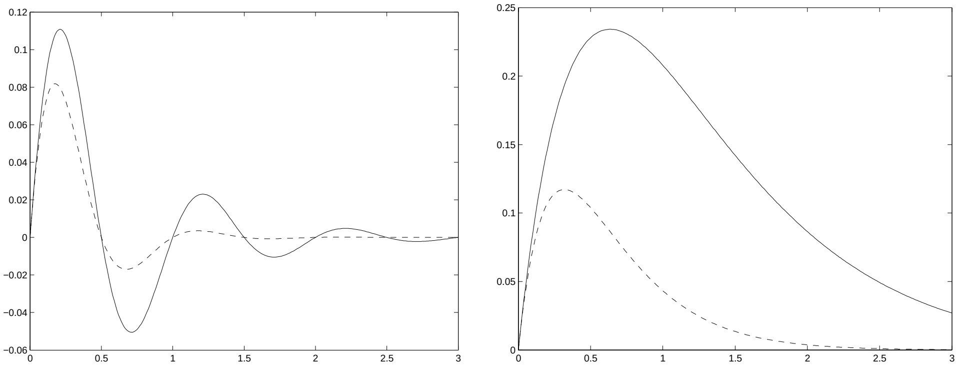


Figure A.3: Response of an under-damped (left) and a critically damped (right) oscillator to an impulse signal. The damped frequency wd is fixed, while different damping values are considered: β (solid line) and 2β (dashed line). Notice how β controls the reaction to perturbations along time (both magnitude and temporal influence).

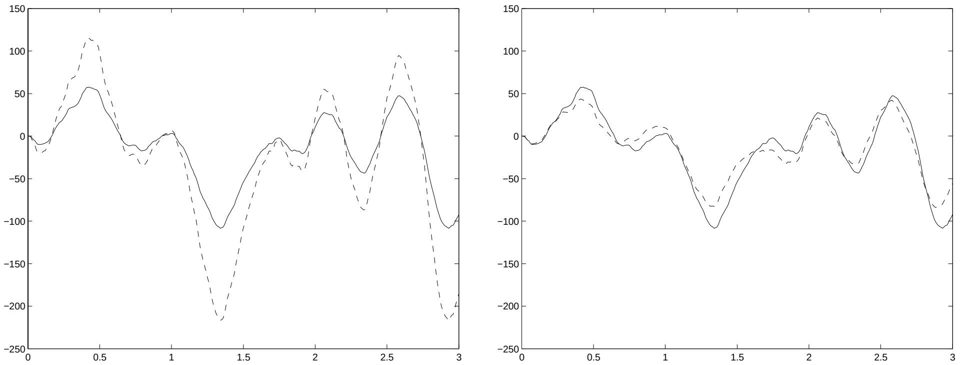


Figure A.4: Evolution of a AR(2) process of fixed $wd = 2\pi$. On the left, plots for (β, b_0) (solid) and $(\beta, 2b_0)$ (dashed), showing that b_0 can be interpreted as an scaling factor. On the right, plots for (β, b_0) (solid) and $(2\beta, b_0)$ (dashed). The bigger the β, the smoother the reaction to perturbations.

A.2.1 Parameterising a Critically Damped Oscillation

As noted before, a valid model to be applied in a real practical application many times require to be a very loose and generic one. In many cases, the only a priori knowledge that can be exploited is the range of values that a process may take, and some notion of the process evolution along time. When this evolution is quite erratic and noisy–like, CBM can be a good and computationally cheap option. When the state evolution is more smooth and has more inertia, the use of an AR(2) provides a better description. Interpreting the parameters (α_1, α_2) as coefficients of an Harmonic Oscillator, a non–oscillatory state evolution can be obtained by imposing $w_d = 0$ (i.e, parameterising a Critically Damped Oscillation (CDO)). Due to this constrain, it holds that $-\alpha_1^2 = 4\alpha_2$. If it is computed the variance of state values as $t \to \infty$, it can be shown that it tends to $\frac{1}{\epsilon} b_0 b_0^T$, where now

$$\epsilon = 1 - \alpha_2^2 + 4\alpha_2 + \frac{8\alpha_2^2}{1 - \alpha_2} \tag{A.14}$$

Using the properties of Gaussian distributions, a shape parameter b_t can be constrained in a desired range $[-l, l]$ by imposing the relation

$$l = 3\frac{b_0}{\sqrt{\epsilon}} \ .$$

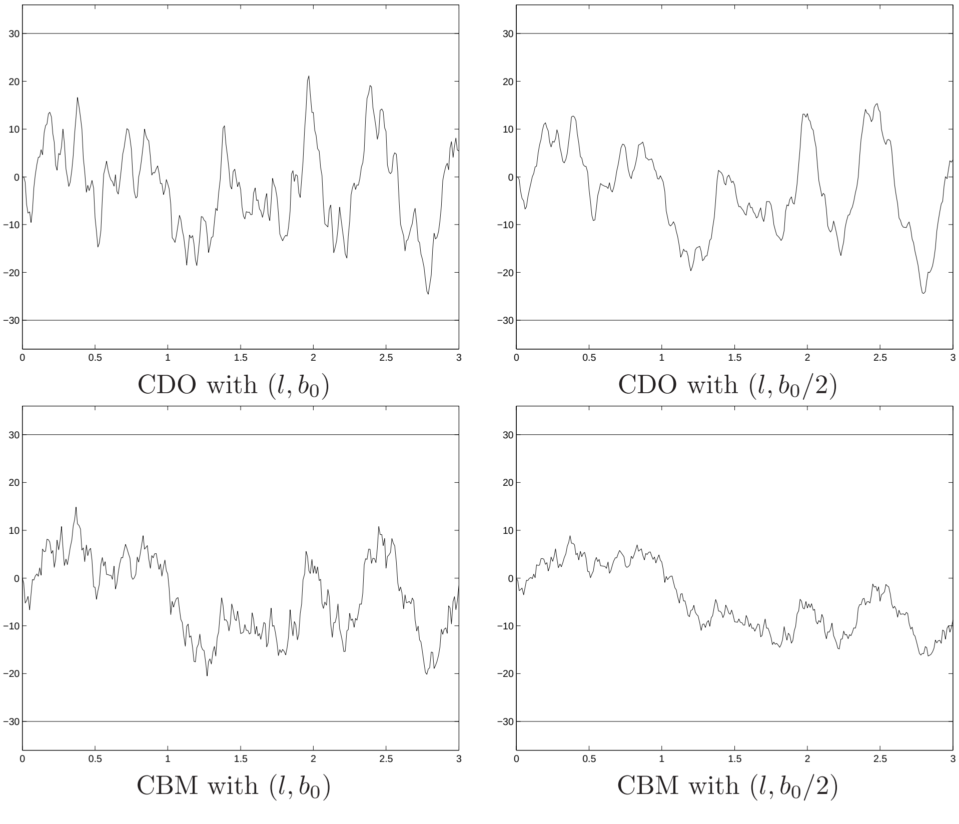
Given b_0, this expression determines ϵ as $\epsilon = \left(\frac{3b_0}{l}\right)^2$. Thus, having ϵ, the value of α_2 can be isolated from (A.14). The solution that determines α_2 is not unique, but imposing $\alpha_2 \in \mathbb{R}$, it is found that

$$\alpha_2 = -1 + \frac{\epsilon}{3^{1/3}(-9\epsilon + \sqrt{3}\sqrt{27\epsilon^2 + \epsilon^3})^{1/3}} - \frac{(-9\epsilon + \sqrt{3}\sqrt{27\epsilon^2 + \epsilon^3})^{1/3}}{3^{2/3}} \ ,$$

and from α_2, the value of β is determined as

$$\beta = -\frac{\log(-\alpha_2)}{2\tau} \ .$$

From the presented procedure, parameterising a desired constraint requires specifying 2 parameters (l, b_0). Figure A.5 shows the effect of b_0 for a fixed maximum range value l, for dynamics modelled respectively by a CDO and a CBM. For CDOs, it can be seen that the smaller b_0, the smoother the trajectory (β is also smaller). However, the average rate in which the range of feasible values is explored is basically maintained. This is a behaviour completely different to the one of CBMs. In CBMs, the value of b_0 determines the average velocity of dynamics, so the range of feasible values is explored more slowly. At long term both models share the same statistics, but CDOs allow moving inside the range of feasible values with a fast and smooth movement, while CBMs only can model this fast movement by means of a very abrupt and erratic evolution. The behaviour of both models is similar just when the feasible range of valid values is up to an order of magnitude bigger than assumed average perturbations.



CDO with (l, b_0)

CDO with $(l, b_0/2)$

CBM with (l, b_0)

CBM with $(l, b_0/2)$

Figure A.5: Evolution of a CDO (top) and a CBM (bottom) for two given parameterisation.

Appendix B

Probabilistic Origins of the Kalman Filter

The objective of this appendix is document the derivation of the KF equations from a probabilistic point of view, complementing in that way the explanation commonly given in the literature, derived from a computational point of view [9, 122, 94]. For the sake of readability and completeness, the appendix re-examine briefly some key topics already described in the thesis chapters.

The KF addresses the problem of estimating recursively the state $\mathbf{x}_t$ of a given continuous Markov process, from its observations $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \ldots, \mathbf{y}_t\}$ obtained along time. The process to be estimated is assumed to be governed by a linear stochastic difference equation

$$\mathbf{x}_t \quad = \quad \mathbf{A}\mathbf{x}_{t-1} + \mathbf{V}\mathbf{v}_t \ . \tag{B.1}$$

This expression describe the evolution of $\mathbf{x}_t$ by the summation of a deterministic and a stochastic term. The deterministic term models the expected evolution of $\mathbf{x}_t$ by means of a *system transition matrix* $\mathbf{A}$. This matrix established a linear auto–regressive relation between successive states of $\mathbf{x}_t$. However, no mathematical model represent perfectly the evolution of a process. Furthermore, the dynamics of a process can also be driven by disturbances that can be neither controlled nor modelled deterministically. For this reason, an stochastic term is also considered. This term is governed by a random variable $\mathbf{v}_t$ denoting an independent Gaussian white noise sequence $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The matrix $\mathbf{V}$ multiplying $\mathbf{v}_t$ transforms the noise sequence to mimic the distribution of $\mathcal{N}(\mathbf{0}, \mathbf{Q})$, where $\mathbf{Q} = \mathbf{V}\mathbf{V}^T$. The matrix $\mathbf{Q}$ is referred as the *process noise covariance* and represents the inaccuracy in the deterministic dynamic model used.

Observations $\mathbf{y}_{1:t}$ are collected by a measurement process, which is modelled by a linear stochastic equation given by

$$\mathbf{y}_t \quad = \quad \mathbf{H}\mathbf{x}_t + \mathbf{W}\mathbf{w}_t \ . \tag{B.2}$$

$\mathbf{H}$ is the *measurement matrix*, and describe the deterministic linear relation between the state and its observations. The sthocastic part of (B.2) represents the disturbances corrupting measurements. It is governed by a random variable $\mathbf{w}_t$ denoting an independent Gaussian white noise sequence $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This sequence is modified to be distributed as $\mathcal{N}(\mathbf{0}, \mathbf{R})$ by means of the matrix $\mathbf{W}$. The matrix $\mathbf{R} = \mathbf{W}\mathbf{W}^T$ is referred as the *measurement noise covariance*.

From a Bayesian point of view, the task carried by the KF is the estimation of the conditional probability density of $\mathbf{x}_t$, conditioned on the data $\mathbf{y}_{1:t}$. Formally, this means characterise the density

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \ .$$

Applying the Bayes' theorem, this can be developed as

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:t}) &= \frac{p(\mathbf{x}_t\mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t})} \ , \\ &= \frac{p(\mathbf{x}_t\mathbf{y}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_{1:t-1})}{p(\mathbf{y}_{1:t})} \ , \\ &= \frac{p(\mathbf{x}_t\mathbf{y}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_{1:t-1})} \ , \\ &= \frac{p(\mathbf{x}_t\mathbf{y}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \ . \end{aligned} \tag{B.3}$$

For the modelisation of the problem done, given an initial Gaussian distribution of the process state $\mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0, \boldsymbol{\Sigma}_0^{\mathbf{xx}})$, this is propagated along time through stochastic linear equations with white Gaussian noise. A consequence of that is that the distribution of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is Gaussian, too. An important implication of that is that the right side terms in (B.3) are necessarily Gaussian. This fact is important, as allows to take advantage of the relation between the joint and conditional distribution of Gaussian random variables, in order to parameterise $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. The following section describes this relation, and next it is used to characterise $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

B.1 Joint and Conditional Gaussian Random Variables

Two random vectors $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian if the stacked vector $\mathbf{z} = [\mathbf{x}\ \mathbf{y}]^T$ is Gaussian, with parameters

$$\mathcal{N}(\hat{\mathbf{z}}, \boldsymbol{\Sigma}^{\mathbf{zz}}) = \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}^{\mathbf{xx}} & \boldsymbol{\Sigma}^{\mathbf{xy}} \\ \boldsymbol{\Sigma}^{\mathbf{yx}} & \boldsymbol{\Sigma}^{\mathbf{yy}} \end{bmatrix} \right) \ ,$$

where

$$\begin{aligned} \boldsymbol{\Sigma}^{\mathbf{xx}} &= E[(\mathbf{x}-\hat{\mathbf{x}})(\mathbf{x}-\hat{\mathbf{x}})^T] \ , \\ \boldsymbol{\Sigma}^{\mathbf{yy}} &= E[(\mathbf{y}-\hat{\mathbf{y}})(\mathbf{y}-\hat{\mathbf{y}})^T] \ , \\ \boldsymbol{\Sigma}^{\mathbf{xy}} &= E[(\mathbf{x}-\hat{\mathbf{x}})(\mathbf{y}-\hat{\mathbf{y}})^T] = (\boldsymbol{\Sigma}^{\mathbf{yx}})^T \ , \end{aligned}$$

are the blocks of the partitioned covariance matrix.

If $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian, they are also marginally Gaussian. That is,

$$\begin{aligned} p(\mathbf{x}) &= \int p(\mathbf{x},\mathbf{y})d\mathbf{y} = \mathcal{N}(\hat{\mathbf{x}}, \boldsymbol{\Sigma}^{\mathbf{xx}}) \ , \\ p(\mathbf{y}) &= \int p(\mathbf{x},\mathbf{y})d\mathbf{x} = \mathcal{N}(\hat{\mathbf{y}}, \boldsymbol{\Sigma}^{\mathbf{yy}}) \ . \end{aligned}$$

Taking that into account, the conditional density $p(\mathbf{x}|\mathbf{y})$ can be analytically computed. Using Bayes this is expressed as

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &= \frac{p(\mathbf{x}\ \mathbf{y})}{p(\mathbf{y})} \ , \qquad \text{(B.4)} \\ &= \frac{|2\pi\boldsymbol{\Sigma}^{\mathbf{zz}}|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{z}-\hat{\mathbf{z}})^T(\boldsymbol{\Sigma}^{\mathbf{zz}})^{-1}(\mathbf{z}-\hat{\mathbf{z}})\right)}{|2\pi\boldsymbol{\Sigma}^{\mathbf{yy}}|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{y}-\hat{\mathbf{y}})^T(\boldsymbol{\Sigma}^{\mathbf{yy}})^{-1}(\mathbf{y}-\hat{\mathbf{y}})\right)} \ , \\ &= \frac{|2\pi\boldsymbol{\Sigma}^{\mathbf{zz}}|^{-1/2}}{|2\pi\boldsymbol{\Sigma}^{\mathbf{yy}}|^{-1/2}}\exp\left(-\frac{1}{2}\underbrace{\left(\tilde{\mathbf{z}}^T(\boldsymbol{\Sigma}^{\mathbf{zz}})^{-1}\tilde{\mathbf{z}} - \tilde{\mathbf{y}}^T(\boldsymbol{\Sigma}^{\mathbf{yy}})^{-1}\tilde{\mathbf{y}}\right)}_{q}\right) \ , \qquad \text{(B.5)} \end{aligned}$$

where

$$\begin{aligned} \tilde{\mathbf{z}} &= \mathbf{z} - \hat{\mathbf{z}} \ , \\ \tilde{\mathbf{y}} &= \mathbf{y} - \hat{\mathbf{y}} \ . \end{aligned}$$

The subexpression in (B.5) denoted as q can be expanded as

$$\begin{aligned} q &= \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Sigma}^{\mathbf{xx}} & \boldsymbol{\Sigma}^{\mathbf{xy}} \\ \boldsymbol{\Sigma}^{\mathbf{yx}} & \boldsymbol{\Sigma}^{\mathbf{yy}} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix} - \tilde{\mathbf{y}}^T(\boldsymbol{\Sigma}^{\mathbf{yy}})^{-1}\tilde{\mathbf{y}} \ , \\ &= \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix}^T \underbrace{\begin{bmatrix} \mathbf{S}^{\mathbf{xx}} & \mathbf{S}^{\mathbf{xy}} \\ \mathbf{S}^{\mathbf{yx}} & \mathbf{S}^{\mathbf{yy}} \end{bmatrix}}_{\mathbf{S}^{\mathbf{zz}}} \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix} - \tilde{\mathbf{y}}^T(\boldsymbol{\Sigma}^{\mathbf{yy}})^{-1}\tilde{\mathbf{y}} \ , \qquad \text{(B.6)} \end{aligned}$$

where the matrix $\mathbf{S}^{\mathbf{zz}} = (\boldsymbol{\Sigma}^{\mathbf{zz}})^{-1}$ can be easily determined by solving

$$\begin{bmatrix} \boldsymbol{\Sigma}^{\mathbf{xx}} & \boldsymbol{\Sigma}^{\mathbf{xy}} \\ \boldsymbol{\Sigma}^{\mathbf{yx}} & \boldsymbol{\Sigma}^{\mathbf{yy}} \end{bmatrix} \begin{bmatrix} \mathbf{S}^{\mathbf{xx}} & \mathbf{S}^{\mathbf{xy}} \\ \mathbf{S}^{\mathbf{yx}} & \mathbf{S}^{\mathbf{yy}} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \ . \qquad \text{(B.7)}$$

Expanding deeper expression (B.6) it is obtained that

$$q = \tilde{\mathbf{x}}^T\mathbf{S}^{\mathbf{xx}}\tilde{\mathbf{x}} + \tilde{\mathbf{x}}^T\mathbf{S}^{\mathbf{xy}}\tilde{\mathbf{y}} + \tilde{\mathbf{y}}^T\mathbf{S}^{\mathbf{yx}}\tilde{\mathbf{x}} + \tilde{\mathbf{y}}^T\mathbf{S}^{\mathbf{yy}}\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^T(\boldsymbol{\Sigma}^{\mathbf{yy}})^{-1}\tilde{\mathbf{y}} \ ,$$

which rearranging terms, it can be equivalently expressed as

$$\begin{aligned} q = &\ (\tilde{\mathbf{x}} + (\mathbf{S}^{\mathbf{xx}})^{-1}\mathbf{S}^{\mathbf{xy}}\tilde{\mathbf{y}})^T\mathbf{S}^{\mathbf{xx}}(\tilde{\mathbf{x}} + (\mathbf{S}^{\mathbf{xx}})^{-1}\mathbf{S}^{\mathbf{xy}}\tilde{\mathbf{y}}) + \\ &+ \tilde{\mathbf{y}}^T(\mathbf{S}^{\mathbf{yy}} - \mathbf{S}^{\mathbf{yx}}(\mathbf{S}^{\mathbf{xx}})^{-1}\mathbf{S}^{\mathbf{xy}})\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^T(\boldsymbol{\Sigma}^{\mathbf{yy}})^{-1}\tilde{\mathbf{y}} \ . \qquad \text{(B.8)} \end{aligned}$$

The resolution of (B.7) states that

$$(\mathbf{\Sigma}^{\mathbf{yy}})^{-1} \quad = \quad \mathbf{S}^{\mathbf{yy}} - \mathbf{S}^{\mathbf{yx}}(\mathbf{S}^{\mathbf{xx}})^{-1}S^{\mathbf{xy}} \ ,$$

and applying that result on B.8 it is obtained a quadratic form that characterises the Gaussian density $p(\mathbf{x}|\mathbf{y})$, given by

$$q \quad = \quad (\tilde{\mathbf{x}} + (\mathbf{S}^{\mathbf{xx}})^{-1}\mathbf{S}^{\mathbf{xy}}\tilde{\mathbf{y}})^T\mathbf{S}^{\mathbf{xx}}(\tilde{\mathbf{x}} + (\mathbf{S}^{\mathbf{xx}})^{-1}\mathbf{S}^{\mathbf{xy}}\tilde{\mathbf{y}}) \ . \tag{B.9}$$

From this expression, the mean and covariance of this density can be easily identified. The mean is obtained by solving the following equality

$$\begin{aligned} \mathbf{x} - E[\mathbf{x}|\mathbf{y}] &= \tilde{\mathbf{x}} + (\mathbf{S}^{\mathbf{xx}})^{-1}S^{\mathbf{xy}}\tilde{\mathbf{y}} \ , \\ \mathbf{x} - E[\mathbf{x}|\mathbf{y}] &= \mathbf{x} - \hat{\mathbf{x}} + (\mathbf{S}^{\mathbf{xx}})^{-1}\mathbf{S}^{\mathbf{xy}}(\mathbf{y} - \hat{\mathbf{y}}) \ , \\ E[\mathbf{x}|\mathbf{y}] &= \hat{\mathbf{x}} - (\mathbf{S}^{\mathbf{xx}})^{-1}\mathbf{S}^{\mathbf{xy}}(\mathbf{y} - \hat{\mathbf{y}}) \ . \end{aligned} \tag{B.10}$$

The covariance of $p(\mathbf{x}|\mathbf{y})$ must correspond simply to

$$Cov[\mathbf{x}|\mathbf{y}] \quad = \quad (\mathbf{S}^{\mathbf{xx}})^{-1} \ .$$

Making the proper substitutions using the terms obtained by solving (B.7), the final expressions of the first and second statistics of $p(\mathbf{x}|\mathbf{y})$ are given by

$$\begin{aligned} E[\mathbf{x}|\mathbf{y}] &= \hat{\mathbf{x}} + \mathbf{\Sigma}^{\mathbf{xy}}(\mathbf{\Sigma}^{\mathbf{yy}})^{-1}(\mathbf{y} - \hat{\mathbf{y}}) \ , \\ Cov[\mathbf{x}|\mathbf{y}] &= \mathbf{\Sigma}^{\mathbf{xx}} - \mathbf{\Sigma}^{\mathbf{xy}}(\mathbf{\Sigma}^{\mathbf{yy}})^{-1}\mathbf{\Sigma}^{\mathbf{yx}} \ . \end{aligned}$$

Results obtained conclude that given two vectors $\mathbf{x}$ and $\mathbf{y}$ jointly Gaussian, the mean and covariance of the conditional density $p(\mathbf{x}|\mathbf{y})$ can be determined by a linear combination of the parameters of

- $p(\mathbf{x}) \sim \mathcal{N}(\hat{\mathbf{x}}, \mathbf{\Sigma}^{\mathbf{xx}})$.
- $p(\mathbf{y}) \sim \mathcal{N}(\hat{\mathbf{y}}, \mathbf{\Sigma}^{\mathbf{yy}})$.
- $\mathbf{\Sigma}^{\mathbf{xy}}$.

B.2 Solving the Kalman Estimation Problem

The problem to be solved by the KF presents a clear analogy with the one in the previous section. The expression to be characterised is

$$p(\mathbf{x}_t|\mathbf{y}_t\mathbf{y}_{1:t-1}) \quad = \quad \frac{p(\mathbf{x}_t\mathbf{y}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \ ,$$

which in fact present exactly the same relations as (B.4), but now the variables $\mathbf{x}_t$ and $\mathbf{y}_t$ are conditioned on a third variable $\mathbf{y}_{1:t-1}$. As the assumptions made imply that $\mathbf{x}_t$ and $\mathbf{y}_t$ are jointly Gaussian, the resolution scheme is exactly the same that for $p(\mathbf{x}|\mathbf{y})$. By analogy with the previous result, the first and second statistics of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ are determined from a linear combination of the parameters of

- $p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \sim \mathcal{N}(\hat{\mathbf{x}}_{t|t-1}, \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1})$.
- $p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) \sim \mathcal{N}(\hat{\mathbf{y}}_{t|t-1}, \mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})$.
- $\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}$.

B.3 Characterising $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$

$p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ is a Gaussian density denoted as *state prediction* distribution. Characterising it implies determining its first and second order statistics $E[\mathbf{x}_t|\mathbf{y}_{1:t-1}]$ and $Cov[\mathbf{x}_t|\mathbf{y}_{1:t-1}]$, commonly expressed respectively as $\hat{\mathbf{x}}_{t|t-1}$ and $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}$. $\hat{\mathbf{x}}_{t|t-1}$ is determined by applying the system process equation (B.1) in the corresponding expectation computation. From the properties of the expectation computation (see table B.1), it follows that

$$\begin{aligned} E[\mathbf{x}_t|\mathbf{y}_{1:t-1}] &= E[\mathbf{A}\mathbf{x}_{t-1} + \mathbf{V}\mathbf{v}_t|\mathbf{y}_{1:t-1}] \ , \\ &= \mathbf{A}E[\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}] + \mathbf{V}E[\mathbf{v}_t|\mathbf{y}_{1:t-1}] \ . \end{aligned}$$

As by definition $\mathbf{v}_t$ is an independent variable, with $E[\mathbf{v}_t] = 0$, it results that

$$\begin{aligned} E[\mathbf{x}_t|\mathbf{y}_{1:t-1}] &= \mathbf{A}E[\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}] + \mathbf{V}E[\mathbf{v}_t] \ , \\ &= \mathbf{A}\hat{\mathbf{x}}_{t-1|t-1} \ . \end{aligned}$$

To characterise $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}$, it is useful to first define the *state prediction error* $\tilde{\mathbf{x}}_{t|t-1}$, which, by making similar substitutions to the ones to compute $\hat{\mathbf{x}}_{t|t-1}$, corresponds to

$$\begin{aligned} \tilde{\mathbf{x}}_{t|t-1} &\triangleq \mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1} \ , \\ &= \mathbf{x}_t - \mathbf{A}\hat{\mathbf{x}}_{t-1|t-1} \ , \\ &= \mathbf{A}\mathbf{x}_{t-1} + \mathbf{V}\mathbf{v}_t - \mathbf{A}\hat{\mathbf{x}}_{t-1|t-1} \ , \\ &= \mathbf{A}(\mathbf{x}_{t-1} - \hat{\mathbf{x}}_{t-1|t-1}) + \mathbf{V}\mathbf{v}_t \ , \\ &= \mathbf{A}\tilde{\mathbf{x}}_{t-1|t-1} + \mathbf{V}\mathbf{v}_t \ . \end{aligned}$$

This simplifies the derivation of *state prediction covariance* expression. By definition, it corresponds to

$$\begin{aligned} \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} &= E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})^T|\mathbf{y}_{1:t-1}] \ , \\ &= E[\tilde{\mathbf{x}}_{t|t-1}\tilde{\mathbf{x}}^T_{t|t-1}|\mathbf{y}_{1:t-1}] \ , \\ &= E[(\mathbf{A}\tilde{\mathbf{x}}_{t-1|t-1} + \mathbf{V}\mathbf{v}_t)(\mathbf{A}\tilde{\mathbf{x}}_{t-1|t-1} + \mathbf{V}\mathbf{v}_t)^T|\mathbf{y}_{1:t-1}] \ . \end{aligned} \tag{B.11}$$

From the properties of the expectation computation, and provided that $\mathbf{v}_t$ is a noise term independent from $\mathbf{y}_{1:t-1}$, it results that

$$\begin{aligned} \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} = {} & \mathbf{A}E[\tilde{\mathbf{x}}_{t-1|t-1}\tilde{\mathbf{x}}^T_{t-1|t-1}|\mathbf{y}_{1:t-1}]\mathbf{A}^T + \mathbf{V}E[\mathbf{v}_t\mathbf{v}^T_t]\mathbf{V}^T + \\ & + \mathbf{A}E[\tilde{\mathbf{x}}_{t-1|t-1}\mathbf{v}^T_t]\mathbf{V}^T + \mathbf{V}E[\mathbf{v}_t\tilde{\mathbf{x}}^T_{t-1|t-1}]\mathbf{A}^T \ . \end{aligned}$$

As the noise in the system process is assumed uncorrelated (i.e. $E[\tilde{\mathbf{x}}_{t-1|t-1}\mathbf{v}_t^T] = E[\mathbf{v}_t\tilde{\mathbf{x}}_{t-1|t-1}^T] = 0$), previous expression finally reduces to

$$\mathbf{\Sigma}_{t|t-1}^{\mathbf{xx}} = \mathbf{A}\mathbf{\Sigma}_{t-1|t-1}^{\mathbf{xx}}\mathbf{A}^T + \mathbf{Q} \ .$$

Thus, summarising, the state prediction distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ corresponds to a Gaussian distribution with parameters

$$\mathcal{N}(\hat{\mathbf{x}}_{t|t-1}, \mathbf{\Sigma}_{t|t-1}^{\mathbf{xx}}) = \mathcal{N}(\mathbf{A}\hat{\mathbf{x}}_{t-1|t-1},\ \mathbf{A}\mathbf{\Sigma}_{t-1|t-1}^{\mathbf{xx}}\mathbf{A}^T + \mathbf{Q}) \ .$$

B.4 Characterising $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$

$p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ is a Gaussian density denoted as *measurement prediction* distribution, whose statistics $E[\mathbf{y}_t|\mathbf{y}_{1:t-1}]$ and $Cov[\mathbf{y}_t|\mathbf{y}_{1:t-1}]$ are commonly expressed as $\hat{\mathbf{y}}_{t|t-1}$ and $\mathbf{\Sigma}_{t|t-1}^{\mathbf{yy}}$.

$$\begin{aligned}
\hat{\mathbf{y}}_{t|t-1} &\triangleq E[\mathbf{y}_t|\mathbf{y}_{1:t-1}] \ , \\
\mathbf{\Sigma}_{t|t-1}^{\mathbf{yy}} &\triangleq Cov[\mathbf{y}_t|\mathbf{y}_{1:t-1}] = E[(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})^T|\mathbf{y}_{1:t-1}] \ .
\end{aligned}$$

To obtain $\hat{\mathbf{y}}_{t|t-1}$, equation (B.2) is substituted in the corresponding expectation computation, obtaining

$$\begin{aligned}
E[\mathbf{y}_t|\mathbf{y}_{1:t-1}] &= E[\mathbf{H}\mathbf{x}_t + \mathbf{W}\mathbf{w}_t|\mathbf{y}_{1:t-1}] \ , \\
&= \mathbf{H}E[\mathbf{x}_t|\mathbf{y}_{1:t-1}] + \mathbf{W}E[\mathbf{w}_t|\mathbf{y}_{1:t-1}] \ .
\end{aligned}$$

As $\mathbf{w}_t$ is an independent variable with $E[\mathbf{w}_t] = 0$, it results that

$$\begin{aligned}
E[\mathbf{y}_t|\mathbf{y}_{1:t-1}] &= \mathbf{H}E[\mathbf{x}_t|\mathbf{y}_{1:t-1}] + \mathbf{W}E[\mathbf{w}_t] \ , \\
&= \mathbf{H}\hat{\mathbf{x}}_{t|t-1} \ .
\end{aligned}$$

To compute $\mathbf{\Sigma}_{t|t-1}^{\mathbf{yy}}$ it is useful to first define the *measurement prediction error* $\tilde{\mathbf{y}}_{t|t-1}$, which, by making similar substitutions to the ones to compute $\hat{\mathbf{y}}_{t|t-1}$, corresponds to

$$\begin{aligned}
\tilde{\mathbf{y}}_{t|t-1} &\triangleq \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1} \ , \\
&= \mathbf{y}_t - \mathbf{H}\hat{\mathbf{x}}_{t|t-1} \ , \\
&= \mathbf{H}\mathbf{x}_t + \mathbf{W}\mathbf{w}_t - \mathbf{H}\hat{\mathbf{x}}_{t|t-1} \ , \\
&= \mathbf{H}(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1}) + \mathbf{W}\mathbf{w}_t \ , \\
&= \mathbf{H}\tilde{\mathbf{x}}_{t|t-1} + \mathbf{W}\mathbf{w}_t \ . \qquad \text{(B.12)}
\end{aligned}$$

Using that result, the *measurement prediction covariance* $\mathbf{\Sigma}_{t|t-1}^{\mathbf{yy}}$ is computed as

$$\begin{aligned}
\mathbf{\Sigma}_{t|t-1}^{\mathbf{yy}} &= E[(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})^T|\mathbf{y}_{1:t-1}] \ , \\
&= E[\tilde{\mathbf{y}}_{t|t-1}\tilde{\mathbf{y}}_{t|t-1}^T|\mathbf{y}_{1:t-1}] \ , \\
&= E[(\mathbf{H}\tilde{\mathbf{x}}_{t|t-1} + \mathbf{W}\mathbf{w}_t)(\mathbf{H}\tilde{\mathbf{x}}_{t|t-1} + \mathbf{W}\mathbf{w}_t)^T|\mathbf{y}_{1:t-1}] \ , \\
&= \mathbf{H}E[\tilde{\mathbf{x}}_{t|t-1}\tilde{\mathbf{x}}_{t|t-1}^T|\mathbf{y}_{1:t-1}]\mathbf{H}^T + \mathbf{W}E[\mathbf{w}_t\mathbf{w}_t^T]\mathbf{W}^T + \ , \\
&\quad + \mathbf{H}E[\tilde{\mathbf{x}}_{t|t-1}\mathbf{w}_t^T]\mathbf{W}^T + \mathbf{W}E[\mathbf{w}_t\tilde{\mathbf{x}}_{t|t-1}^T]\mathbf{H}^T \ .
\end{aligned}$$

Using the fact that $\mathbf{w}_t$ and $\tilde{\mathbf{x}}_{t|t-1}$ are uncorrelated (i.e. $E[\tilde{\mathbf{x}}_{t|t-1}\mathbf{w}_t^T] = E[\mathbf{w}_t\tilde{\mathbf{x}}_{t|t-1}^T] = 0$), it results that

$$\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1} \quad = \quad \mathbf{H}\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T + \mathbf{R} \ .$$

Thus, summarising, the measurement prediction distribution $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ corresponds to a Gaussian distribution with parameters

$$\mathcal{N}(\hat{\mathbf{y}}_{t|t-1}, \mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1}) \quad = \quad \mathcal{N}(\mathbf{H}\hat{\mathbf{x}}_{t|t-1},\, \mathbf{H}\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T + \mathbf{R}) \ .$$

B.5 Characterising $\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}$

The cross-covariance between $\mathbf{x}_t$ and $\mathbf{y}_t$ conditioned on $\mathbf{y}_{1:t-1}$ is defined as

$$\begin{aligned}\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1} &= E[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})^T|\mathbf{y}_{1:t-1}] \ , \\ &= E[\tilde{\mathbf{x}}_{t|t-1}\tilde{\mathbf{y}}_{t|t-1}^T|\mathbf{y}_{1:t-1}] \ .\end{aligned}$$

From the result derived in (B.12), and applying the properties of expectation computation, this term is developed as

$$\begin{aligned}E[\tilde{\mathbf{x}}_{t|t-1}\tilde{\mathbf{y}}_{t|t-1}^T|\mathbf{y}_{1:t-1}] &= E[\tilde{\mathbf{x}}_{t|t-1}(\mathbf{H}\tilde{\mathbf{x}}_{t|t-1} + \mathbf{W}\mathbf{w}_t)^T|\mathbf{y}_{1:t-1}] \ , \\ &= E[\tilde{\mathbf{x}}_{t|t-1}\tilde{\mathbf{x}}_{t|t-1}^T\mathbf{H}^T|\mathbf{y}_{1:t-1}] + E[\tilde{\mathbf{x}}_{t|t-1}\mathbf{w}_t^T\mathbf{W}^T|\mathbf{y}_{1:t-1}] \ , \\ &= E[\tilde{\mathbf{x}}_{t|t-1}\tilde{\mathbf{x}}_{t|t-1}^T|\mathbf{y}_{1:t-1}]\mathbf{H}^T + E[\tilde{\mathbf{x}}_{t|t-1}\mathbf{w}_t^T|\mathbf{y}_{1:t-1}]\mathbf{W}^T \ ,\end{aligned}$$

which due to the fact that $\mathbf{w}_t$ is an independent and uncorrelated noise term, it reduces to

$$E[\tilde{\mathbf{x}}_{t|t-1}\tilde{\mathbf{y}}_{t|t-1}^T|\mathbf{y}_{1:t-1}] \quad = \quad \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T \ .$$

B.6 Joining Results: Characterising $p(\mathbf{x}_t|\mathbf{y}_{1:t})$

Using the results detailed in section B.1, the moments of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ are finally computed by

$$\begin{aligned}E[\mathbf{x}_t|\mathbf{y}_{1:t}] &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}(\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})^{-1}(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}) \ , \\ Cov[\mathbf{x}_t|\mathbf{y}_{1:t}] &= \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} - \mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}(\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})^{-1}(\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1})^T \ .\end{aligned}$$

As all the terms involved have been characterised, it just lacks making the proper substitutions and grouping terms to arrive to the final expressions

$$\begin{aligned}\hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T(\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})^{-1}(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}) \ , \\ \mathbf{\Sigma}^{\mathbf{xx}}_{t|t} &= \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} - \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T(\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})^{-1}\mathbf{H}\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} \ , \\ &= \left(I - \mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T(\mathbf{\Sigma}^{\mathbf{yy}}_{t|t-1})^{-1}\mathbf{H}\right)\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} \ .\end{aligned}$$

Given an estimation of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, its statistics are recursively reestimated by iterating the two phases described in algorithm 13. **K** is denoted the *Kalman gain*, and modulates the influence of the predictions and measurements in the final estimation. For didactic examples describing the performance of the Kalman algorithm, the reader may refer to [83].

Algorithm 13 Kalman Filter Iteration

```
$\left[\left(\hat{\mathbf{x}}_{t|t}, \mathbf{\Sigma}^{\mathbf{xx}}_{t|t}\right)\right] = \mathrm{KF}\left[\left(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\right), \mathbf{y}_{1:t}\right]$
  {Prediction step ( $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ estimation)}

  $\hat{\mathbf{x}}_{t|t-1} = \mathbf{A}\hat{\mathbf{x}}_{t-1|t-1}$
  $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1} = \mathbf{A}\mathbf{\Sigma}^{\mathbf{xx}}_{t-1|t-1}\mathbf{A}^T + \mathbf{Q}$

  {Updating Step (fusion of $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$,$p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ and $\mathbf{\Sigma}^{\mathbf{xy}}_{t|t-1}$)}

  $\mathbf{K} = \frac{\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T}{\mathbf{H}\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}\mathbf{H}^T + \mathbf{R}}$
  $\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}(\mathbf{y}_t - \mathbf{H}\hat{\mathbf{x}}_{t|t-1})$
  $\mathbf{\Sigma}^{\mathbf{xx}}_{t|t} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{\Sigma}^{\mathbf{xx}}_{t|t-1}$
```

Table B.1: Expectation properties

Expectation definition $E[x] = \int xp(x)dx$
Expectation of a function $f(x)$ $E[f(x)] = \int f(x)p(x)dx$
Given a constant c: $E[x+c] = E[x] + c$ $E[cx] = cE[x]$ $E[cx\|z] = cE[x\|z]$
Given two random variables x,y $E[x+y] = E[x] + E[y]$ $E[x+y\|z] = E[x\|z] + E[y\|z]$
Given two random independent variables x,y $E[xy] = E[x]E[y]$

Variance definition $Var[x] = E[(x - E[x])(x - E[x])^T] = E[x^2] - E[x]^2$
Given a constant c: $Var[x+c] = Var[x]$ $Var[cx] = cVar[x]c^T$
Given two random independent variables x,y $Var[x+y] = Var[x] + Var[y]$

Given a random variable x with Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$
$E[x] = \mu$ $Var[x] = \sigma^2$

Appendix C

Expectation Computation Using Monte Carlo

The estimation of expectation by Monte Carlo Methods has its origins in the Central Limit Theorem. This theorem states that the distribution of an average value tends to be Normal, even when the distribution of the variables from which the average is computed is decidedly non-Normal. The expectation operation computes

$$E[x] = \int xp(x)dx \quad .$$

Lets assume a random variable x, with unknown statistics $E[x] = \mu$ and $V[x] = \sigma^2$. The Central Limit Theorem states that the sum of $N \gg 1$ independent samples $\{x^{(i)}\}_{i=1}^{N}$ of $p(x)$ follows a Normal distribution with parameters $\mathcal{N}(N\mu, N\sigma^2)$. This results from considering that samples $\{x^{(i)}\}_{i=1}^{N}$ are instances of identical and independent random variables, from what follows that

$$\begin{aligned} E[x^{(1)} + \ldots + x^{(N)}] &= E[x^{(1)}] + \ldots + E[x^{(N)}] = N\mu \quad , \\ V[x^{(1)} + \ldots + x^{(N)}] &= V[x^{(1)}] + \ldots + V[x^{(N)}] = N\sigma^2 \quad . \end{aligned}$$

From the rule of 3σ the sum of samples $\{x^{(i)}\}_{i=1}^{N}$ is bounded in the following range

$$P\{N\mu - 3\sigma\sqrt{N} < x^{(1)} + \ldots + x^{(N)} < N\mu + 3\sigma\sqrt{N}\} = 0.997 \quad .$$

Dividing the inequality by the number of samples yields

$$P\left\{\mu - \frac{3\sigma}{\sqrt{N}} < \frac{x^{(1)} + \ldots + x^{(N)}}{N} < \mu + \frac{3\sigma}{\sqrt{N}}\right\} = 0.997 \quad .$$

This expression is the key of Monte Carlo methods. What it states is that the mean of a distribution $p(x)$ can be estimated by averaging N independent samples from $p(x)$, with an error inferior to $\frac{3\sigma}{\sqrt{N}}$ the 99.7% of the times. Synthesising that in an expression:

$$P\left\{\left|\left(\frac{1}{N}\sum_{i=1}^{N} x^{(i)}\right) - E[x]\right| < \frac{3\sigma}{\sqrt{N}}\right\} = 0.997 \quad .$$

This result can be directly extended to pose the estimation of the expectation of functions

$$\begin{aligned} E[f(x)] &= \int f(x)p(x)dx \ , \qquad\qquad (C.1) \\ &\simeq \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})) \quad \text{with} \quad x^{(i)} \sim p(x) \ . \end{aligned}$$

A very important characteristic of this result is that the precision in the estimation of expectation depends only in the number of samples N, while deterministic numerical methods to solve integrations depend on N, and the dimension of x. Thus, in high-dimensional spaces the Monte Carlo solution achieve accurate results in the estimation of $E[x]$ requiring an inferior number of samples than deterministic numerical methods .

C.1 Importance Sampling

Importance Sampling is a method to estimate $E[f(x)]$ for cases where it is not possible to sample $p(x)$. The method require that following assumptions hold:

1. although $p(x)$ can not be sampled, it is possible to evaluate a function $p'(x) = Cp(x)$, where C is an unknown constant;
2. there exists an *importance* distribution $q(x)$ that can be sampled, but only evaluated through the function $q'(x) = Dq(x)$, where D is an unknown constant.The importance distribution is selected such that $p(x) > 0$ implies $q(x) > 0$.

With that $E[f(x)]$ (Equation (C.1)) can be reexpressed as:

$$E[f(x)] = \int f(x)\frac{p(x)}{q(x)}q(x)dx \ , \qquad (C.2)$$

This expectation can be computed with the classical Monte Carlo method previously described. Thus, generating N samples from $q(x)$ it can be computed

$$E[f(x)] \simeq \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})\frac{p(x^{(i)})}{q(x^{(i)})} \ .$$

As $p(x)$ and $q(x)$ can only be evaluated indirectly through $p'(x)$ and $q'(x)$, what can be really computed is

$$\begin{aligned} E[f(x)] &\simeq \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})\frac{p'(x^{(i)})/C}{q'(x^{(i)})/D} \\ &\simeq \frac{1}{N}\frac{D}{C}\sum_{i=1}^{N} f(x^{(i)})\frac{p'(x^{(i)})}{q'(x^{(i)})} \qquad (C.3) \end{aligned}$$

So, it is necessary to ascertain D/C. As $p(x)$ is a PDF, it follows that

$$\int p'(x)dx = \int Cp(x)dx = C \ . \tag{C.4}$$

Thus, C can be determined by computing integral (C.4) also by importance sampling. That is

$$\begin{aligned} C &= \int \frac{p'(x)}{q(x)} q(x)dx \\ &= \int \frac{p'(x)}{q'(x)/D} q(x)dx \\ &\simeq \frac{D}{N} \sum_{i=1}^{N} \frac{p'(x^{(i)})}{q'(x^{(i)})} \ . \end{aligned} \tag{C.5}$$

Substituting (C.5) into (C.3), it is obtained the expression that synthesises the computation of $E[f(x)]$ by means of the importance sampling technique:

$$E[f(x)] \simeq \frac{\sum_{i=1}^{N} f(x^{(i)}) \frac{p'(x^{(i)})}{q'(x^{(i)})}}{\sum_{j=1}^{N} \frac{p'(x^{(j)})}{q'(x^{(j)})}} \ . \tag{C.6}$$

This result obtained implicitly provides an approximation of the distribution of $p(x)$ by means of a population of weighted samples. For each $x^{(i)}$ in (C.6) there is a *normalised importance weight* $\tilde{w}^{(i)}$ defined as

$$\tilde{w}^{(i)} = \frac{w(x^{(i)})}{\sum_{i=1}^{N} w(x^{(i)})} \ .$$

where $w(x^{(i)})$ computes the *importance weight*

$$w(x^{(i)}) = \frac{p'(x^{(i)})}{q'(x^{(i)})} \tag{C.7}$$

Thus, (C.6) can be seen as the solution to $E[f(x)]$ using the approximation of $p(x)$ given by

$$\widetilde{p_N}(x) \simeq \sum_{i=1}^{N} \tilde{w}^{(i)} \delta_{(x^{(i)})}(dx) \ , \tag{C.8}$$

where $\delta(x)$ denotes the delta-Dirac function. $\widetilde{p_N}(x)$ is commonly referred as the *point mass* approximation of $p(x)$. Weights $w^{(i)}$ evaluate how good samples from $q(x)$ resemble an hypothetical sampling of $p(x)$. They give an idea of the frequency that each sample should ocurr, in order to generate a sample set distributed according to $p(x)$. Obviously, the behaviour of importance sampling will better as long as $q(x)$ is more similar to $p(x)$. A very important point to remark is that, unlike the case when is possible to sampled directly from $p(x)$, the importance sampling technique does not scale well with dimensionality (see [79]). So in high dimensional spaces the number of samples required to approximate $p(x)$ accurately it is commonly huge.

Appendix D

Performance Quantification

The effectiveness of the different techniques analysed in this thesis has been examined through exhaustive experimental work. Tracking algorithms have been evaluated on testing sequences, which have been distorted with synthetic noise artifacts, in order to check the algorithms performance under different noise conditions.

Several methodologies have been proposed in the literature to synthetically distort images, with the aim to then check the performance of tracking algorithms under different challenging scenarios. In [113] is proposed to add Gaussian noise to video frames. However, in most of video camera systems the effect of this kind of perturbation is too weak to disturb the performance of tracking algorithm. Hence, we find unappropiate to measure tracking performance in situation where the Gaussian noise is big enough to become a challenge, since this would imply testing algorithms in unrealistic situations. The proposal in [15] is to add very realistic disturbances onto sequences (basically occluding objects), but requires a big database of ground truth sequences from which to generate the artifacts. In this thesis the technique in [11] is used, which is a distortion model that tries to reproduce the artifacts found in contour tracking applications where targets of interest are first presegmented from frames. Target presegmentation is a common approach in applications where a robust appearance model of the target or their environment is available, and foreground/background segmentation techniques can be applied. Case–study applications studied in this thesis adjust to this kind of applications. The distortion artifacts added to frames correspond to random circles with the foreground of background colour (see Figure D.1). These artifacts challenge the performance of tracking algorithms by provoking missdetections and wrong localisations of the tracked contour (see Figure D.2).

Algorithms are evaluated in sequences with different levels of distortion. For each noise condition considered, a given sequence is distorted randomly, generating one hundred noise sequences where the performance of algorithms is quantified. Tracking performance is measured using two different methods: an image–based and a contour–based method.

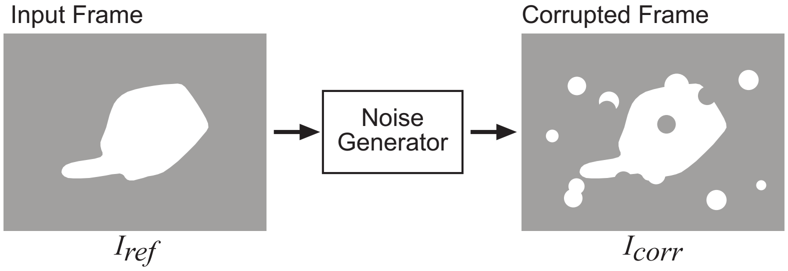


Figure D.1: Frame distortion process.

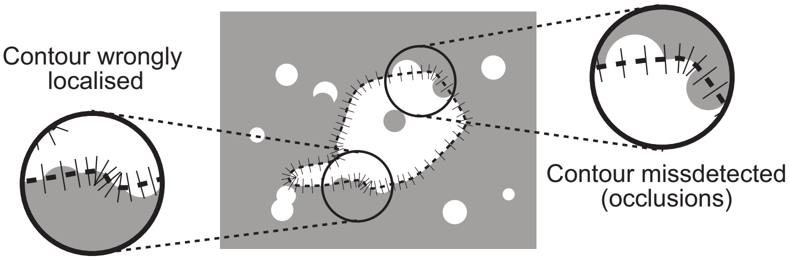


Figure D.2: Contour measurement problems provoked by the noise artifacts added.

D.1 Image–based Performance Quantification

An image–based criterion is used in this thesis to quantify the noise distorting the sequences processed by tracking algorithms, as well as to quantify the performance of tracking algorithms. It is based in measuring the pixel–to–pixel disparity between an ideal ground truth sequence, and another sequence of interest. This ground truth sequence is interpreted as the *signal* that tracking algorithms observe distorted by some noise, and which has to be recovered by them as accurately as possible. The distortion of the input signal, as well as the quality of its retrieval by tracking algorithms, is quantified in terms of the signal–to–noise ratio

$$SNR = 10 \log \left[\frac{signal}{noise} \right] \quad . \tag{D.1}$$

Concerning the distortion of the input sequence, the signal and noise terms of this expression are defined as

$$signal \quad = \quad \sum_{frames} \sum_{x,y} \left[I_{ref}(x,y) - I_0 \right]^2 \quad , \tag{D.2}$$

$$noise \quad = \quad \sum_{frames} \sum_{x,y} \left[\frac{I_{ref}(x,y) - I_{corr}(x,y)}{2} \right]^2 \quad . \tag{D.3}$$

where $I_{ref}(x,y)$ is the pixel value at (x,y) for the ground truth image and $I_{corr}(x,y)$ is the corresponding pixel in the corrupted image. The constant I_0 is set to halfway between the background and foreground pixel values, so that both have the same signal strength, thus ensuring that the SNR is independent of the relative image and object size. A value of $0dB$ corresponds to a distortion of all pixels of the tracking sequence.

Concerning the measurement of how well the input signal is recovered by the tracking algorithm, the SNR terms are defined as

$$\begin{aligned} signal &= 2 \sum_{frames} \sum_{x,y} I_{ref}(x,y)^2 \ , && \text{(D.4)} \\ noise &= \sum_{frames} \sum_{x,y} \left[I_{ref}(x,y) - I_{track}(x,y) \right]^2 \ . && \text{(D.5)} \end{aligned}$$

where I_{track} is a frame synthesised from the state estimated by the tracking algorithm. In this case, the signal just accounts for foreground pixels, as in binary images a background pixel is zero. The scale factor of two in the signal value is chosen so that an SNR of zero (i.e., signal = noise) would occur if the tracker silhouette consisted of a shape of the same area as the ground truth shape, but inaccurately placed so that there is no overlap between the two.

This image–based method provides an objective measure of the tracking conditions (i.e., the noise in the input sequence), but with respect to tracking performance, it is a quite rough measure. It is based on the number of pixels that the ground truth and estimated shape overlap, so, as long as both shapes have a high degree of overlapping, a high SNR value will be obtained, although their respective orientation could be completely opposed. For this reason, the tracking performance measurement is complemented by a contour–based method.

D.2 Contour–based Performance Quantification

This tracking quantification measure is based on a point–to–point comparison between the ideal contour of the shape being tracked, and the contour finally estimated by the tracking algorithm. Given the B-spline control points of the tracked ground truth contour (i.e., $\mathbf{q}_t$), and the ones corresponding to the state estimated by the tracker (i.e., $\hat{\mathbf{q}}_{t|t}$), their contour mean disparity (i.e., the MCE of the estimated contour) is computed as

$$MCE \ = \ \sqrt{\left(\mathbf{q}_t - \hat{\mathbf{q}}_{t|t}\right)^T \mathcal{M} \left(\mathbf{q}_t - \hat{\mathbf{q}}_{t|t}\right)} \ .$$

Matrix $\mathcal{M}$ is a metric matrix that allows to measure the distance between B-spline curves from only their control points. It implicitly establishes a correspondence between points of both contours, reflecting thus the contour disparity even if both have a high overlap.

D.3 Performance Plots

To obtain an statistical view of the performance of the analysed tracking methods, a hundred noisy sequences are generated for each one of several SNR_{IN} values considered. The performance measured by the image–based method is displayed on figures using a the box–and–whiskers or boxplot representation, while the contour–based one is described in a table just from its median value.

A boxplot[85] (see Figure D.3) is a quick graphic approach for summarising the performance achieved in the hundred tracked sequences. It depicts the lower quartile, median, and upper quartile of the performances achieved, as well as the worst and best one. Performances considered as unusual (outliers) are also distinguished.

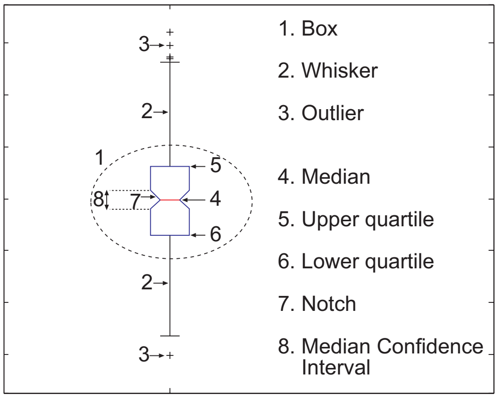


Figure D.3: Graphical elements of the box–and–whiskers representation.

Using boxplots, figures are generated showing the performance of two compared methods, in sequences disturbed by different levels of noise. The results of both method are displayed in pairs, in order to make easier their comparison. For each noisy situation, the method with a better median performance is distinguished by the colour of an small rectangle painted at the bottom of the plot. The tracking performance measured with the contour-based approach is shown in a table attached to the boxplot. The median MCE of each evaluated method is shown, as well as the median of the Mean Paired Difference (MPD) between the two compared methods. When this difference is greater or equal than 0.01 and is statistically significant, the results in the table are written in bold numbers. A percentage is also shown quantifying how one method reduces the MCE of the other one. Figure D.4 shows an example of the kind of plots generated, with added comments to help clarify and understand the information displayed.

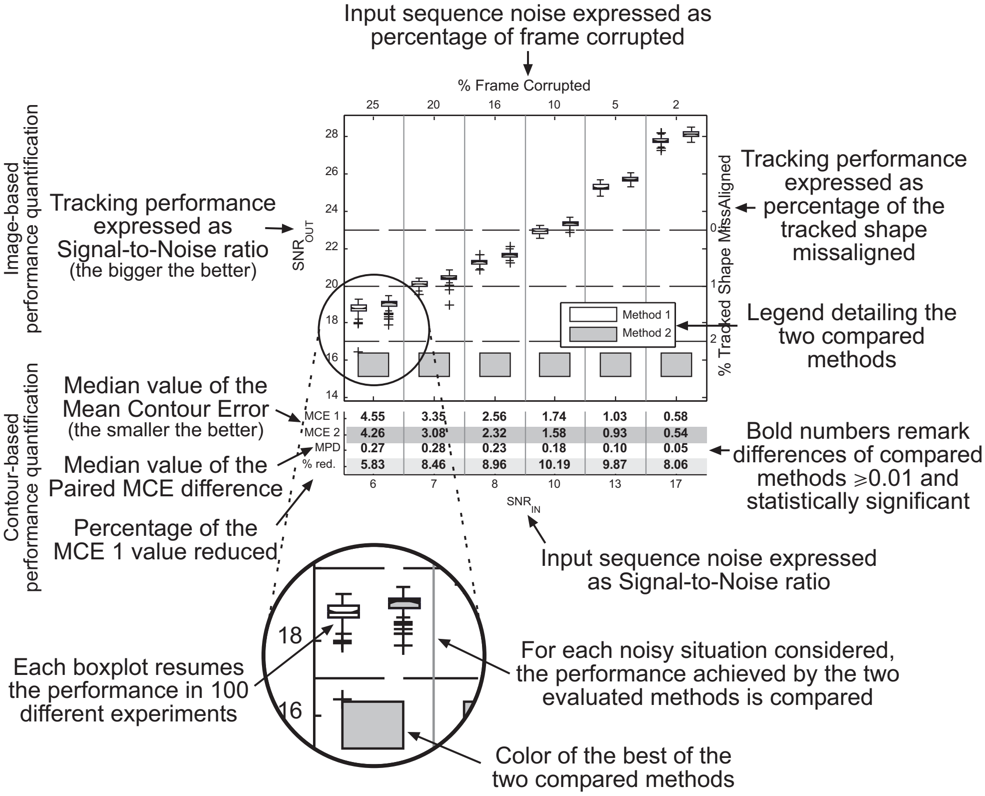


Figure D.4: Plot summarising the results of experiments carried out.

Bibliography

[1] G.A. Ackerson and K.S. Fu. On state estimation in switching environments. *IEEE Transactions on Automatic Control*, 15(1):10–17, January 1970.

[2] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.

[3] Giancarlo Alessandretti, Alberto Broggi, and Pietro Cerri. Vehicle and guard rail detection using radar and vision data fusion. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):95–105, March 2007.

[4] José Manuel Álvarez and Antonio López. Road segmentation for ADAS applications. In *1st CVC Internal Workshop, Computer Vision: Progress of Research and Development*, pages 117–122, Bellaterra, Spain, October 2006.

[5] C. Andrieu, J. de Freitas, and A. Doucet. Sequential Bayesian estimation and model selection applied to neural networks. Technical Report CUED/F-INFENG/TR 341, Cambridge University, 1999. http://svr-www.eng.cam.ac.uk/.

[6] C. Andrieu, N. de Freitas, and A. Doucet. Sequential MCMC for Bayesian model selection. In *IEEE Higher Order Statistics Workshop*, pages 130–134, 1999.

[7] Christophe Andrieu, Manuel Davy, and Arnaud Doucet. Efficient particle filtering for jump Markov systems. application to time-varying autoregressions. *IEEE Transactions on Signal Processing*, 51(7), July 2003.

[8] 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 [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 50(2):174–188, 2002.

[9] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingan Kirubarajan. *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons, Inc., 2001.

[10] Adam Baumberg. Hierarchical shape fitting using an itereated linear filter. In *British Machine Vision Conference*, 1996.

[11] A.M. Baumberg. *Learning Deformable Models for Tracking Human Motion*. PhD thesis, The University of Leeds. School of Computer Studies, 1995.

[12] M. Bertozzi, A. Broggi, M. Carletti, A. Fascioli, T. Graf, P. Grisleri, and M. Meinecke. IR pedestrian detection for advanced driver assistance systems. *Lecture Notes in Computer Science*, 2781:582–590, 2003.

[13] Margrit Betke, Esin Haritaoglu, and Larry S. Davis. Real–time multiple vehicle detection and tracking from a moving vehicle. *Machine Vision and Applications*, (12):69–83, 2000.

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

[15] J. Black, T. Ellis, and P. Rosin. A novel method for video tracking performance evaluation, 2003.

[16] Samuel B. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, January 2004.

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

[18] A. Blake and A. Isard. *Active Contours*. Springer, 1998.

[19] Henk A.P. Blom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33(8):780–783, August 1988.

[20] Yvo Boers and Hans Driessen. A particle filter multi target track before detect application: Some special aspects. In *Fusion 2004*, Stockholm, Sweden, June 2004.

[21] J.Y. Bouguet. Camera calibration toolbox for matlab, October 2004.

[22] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, (Q2):15, 1998.

[23] Christoph Bregler and Stephen M. Omohundro. Surface learning with applications to lipreading. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 43–50. Morgan Kaufmann Publishers, Inc., 1994.

[24] Alberto Broggi, Pietro Cerri, and Pier Claudio Antonello. Multi–resolution vehicle detection using artificial vision. In *IEEE Intelligent Vehicles Symposium*, pages 310–314, Parma, Italy, June 2004.

[25] Robert S. Caprari and Alvin S. Goh. Detect-track-confirm filter with minimal constraints. *IEEE Transactions on Aerospace and Electronic Systems*, 40(1):336–345, January 2004.

[26] J. Carpenter, P. Clifford, and P. Fernhead. An improved particle filter for non-linear problems. In *IEE Proceedings on Radar, Sonar and Navigation*, pages 2–7, 1999.

[27] R. Chen and J. Liu. Mixture Kalman filters. *J. R. Statist. Soc. B*, 62(Part 3):493–508, 2000.

[28] Yunqiang Chen, Thomas Huang, and Yong Rui. Parametric contour tracking using unscented Kalman filter. In *International Conference on Image Processing*, 2002.

[29] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In SC Hilton Head, editor, *IEEE Conf. on Computer Vision and Pattern Recognition*, volume II, pages 142–149, June 2000.

[30] T.F. Cootes and C.J. Taylor. A mixture model for representing shape variation. *Image and Vision Computing*, 17(8):567–573, June 1999.

[31] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. Technical report, Imaging Science and Biomedical Engineering, University of Manchester, 2004.

[32] Corrsys Datron Sensorsysteme Gmbh. *Correvit © Pitch and Roll Sytem*, July 2006. Art. No. 11324.

[33] Crossbow Technology. *VG400. Solid State Vertical Gyro*. Document Part Number: 6020-0020-11 Rev C.

[34] D. Davies, P. Palmer, and M. Mirmehdi. Detection and tracking of very small low constrast objects. In *British Machine Vision Conference*, pages 599–608, 1998.

[35] Fernando de la Torre, Xavier Jove, Elisa Martinez, and Lluis Vicent. Appearance based tracking with switching. In *2nd Congrés Català d'Intel.ligencia Artificial*, pages 274–282, Girona, October 1999.

[36] Frank Dellaert and Chuck Thorpe. Robust car tracking using Kalman filtering and Bayesian templates. In *Proceedings of SPIE: Intelligent Transportation Systems*, volume 3207, October 1997.

[37] N. M. Dempster, A.P. Laird and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39:185–197, 1977.

[38] E.D. Dickmanns. The development of machine vision for road vehicles in the last decade. In *Int. Symp. on Intelligent Vehicles*, Versailles,France, June 2002.

[39] Randal Douc, Olivier Cappé, and Eric Moulines. Comparison of resampling schemes for particle filtering. In *4th Internationalymposium on Image Signal Processing and Analysis*, pages 64–69, 2005.

[40] Arnaud Doucet. On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR. 310, Cambridge University Department of Engineering, 1998.

[41] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[42] Arnaud Doucet, Neil J. Gordon, and Vikram Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624, March 2001.

[43] James M. Ferryman, Stephen J. Maybank, and Anthony D. Worrall. Visual surveillance for moving vehicles. *International Journal of Computer Vision*, 37(2), 2000.

[44] K. Fleischer, H.-H. Nagel, and T. M. Rath. 3D-model-based-vision for innercity driving scenes. In *IEEE Intelligent Vehicles Symposium*, volume 2, pages 477–482, Versailles, France, June 2002.

[45] C. Fraley and A. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *Computer Journal*, 41:578–588, 1998.

[46] Jerome H. Friedman and Lawrence C. Rafsky. Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. *Annals of Statistics*, 7(4):697–717, July 1979.

[47] Arthur Gelb, editor. *Applied Optimal Estimation*. MIT Press, Cambridge, MA, 1974.

[48] Alexander Gepperth, Johann Edelbrunner, and Thomas Bücher. Real-time detection and classification of cars in video sequences. In *IEEE Intelligent Vehicles Symposium*, pages 625–631, Las Vegas, USA, June 2005.

[49] Axel Gern, Uwe Franke, and Paul Levi. Robust vehicle tracking fusing radar and vision.

[50] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140(2):107–113, April 1993.

[51] D. Guo, T. Fraichard, M. Xie, and C. Laugier. Color modeling by spherical influence field in sensing driving environment. In *IEEE Intelligent Vehicles Symposium*, pages 249–254, Dearborn, MI, USA, June 2000.

[52] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and Per-Johan Nordlund. Particle filters for positioning, navigation and tracking. *IEEE Transactions on Signal Processing*, 50, Feb 2002.

[53] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, (3):1157–1182, 2003.

[54] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10), October 1998.

[55] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[56] A. Heap and D. Hogg. Improving specificity in pdms using a hierarchical approach. In Adrian F. Clark, editor, *In Proc. British Machine Vision Conference*, volume 1, pages 80–89, 1997.

[57] T. Heap and D. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. In *Sixth International Conference on Computer Vision*, pages 344–349, 1998.

[58] C. Hilario, J.M. Collado, J.M. Armingol, and A. de la Escalera. Pyramidal image analysis for vehicle detection. In *IEEE Intelligent Vehicles Symposium*, pages 88–93, Las Vegas, USA, June 2005.

[59] B. K. P. Horn. *Robot Vision*. The MIT Press, Cambridge, MA, 1986.

[60] Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[61] Michael Isard and Andrew Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. *Lecture Notes in Computer Science*, 1406:893–908, 1998.

[62] S. J. Julier. The Scaled Unscented Transformation. In *Proceedings of the IEEE American Control Conference*, pages 4555–4559, Anchorage AK, USA, 8–10 May 2002. IEEE.

[63] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[64] S.J. Julier. The spherical simplex unscented transformation. In *IEEE American Control Conference*, volume 3, pages 2430—2434, June 2003.

[65] T. Kalinke, C. Tzomakas, and W. von Seelen. A texture based object detection and an adaptive model-based classification. In *Procs. IEEE Intelligent Vehicles Symposium'98*, pages 341–346, October 1998.

[66] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[67] Zu Kim. Realtime obstacle detection and tracking based on constrained delaunay triangulation. In *IEEE Intelligent Transportation Systems Conference*, pages 548–553, Toronto, Canada, September 2006.

[68] Genshiro Kitagawa. Monte Carlo filter and smoother for non–Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[69] Augustine Kong, Jun S. Liu, and Wing Hung Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

[70] Gildas Lefaix, Éric Marchand, and Patrick Bouthemy. Motion-based obstacle detection and tracking for car driving assistance. In *International Conference on Pattern Recognition*, pages 74–77, Québec, Canada, August 2002.

[71] Peihua Li, Tianwen Zhand, and Bo Ma. Unscented Kalman filter for visual curve tracking. *Image and Vision Computing*, (22):157–164, 2004.

[72] Peihua Li, Tianwen Zhand, and Arthur E.C. Pece. Visual contour tracking based on particle filters. *Image and Vision Computing*, 21(1):111–123, 2003.

[73] Jun S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer-Verlag New York, 2001.

[74] Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[75] Tie Liu, Nanning Zheng, Li Zhao, and Hong Cheng. Learning based symmetric features selection for vehicle detection. In *IEEE Intelligent Vehicles Symposium*, pages 124–129, Las Vegas, USA, June 2005.

[76] John MacCormick. *Probabilistic Modelling and Stochastic Algorithms for Visual Localisation and Tracking.* PhD thesis, Department of Engineering Science. University of Oxford, January 2000.

[77] John MacCormick and Andrew Blake. A probabilistic contour discriminant for object localisation. In *ICCV*, pages 390–395, 1998.

[78] John MacCormick and Michael Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *ECCV (2)*, pages 3–19, 2000.

[79] D.J.C. Mackay. *Learning in Graphical Models*, chapter Introduction to Monte Carlo Methods, pages 175–204. MIT Press, 1999.

[80] Jorge S. Marques and Joao M. Lemos. Optimal and suboptimal shape tracking based on multiple switched dynamic models. *Image and Vision Computing*, 19(8):539–550, July 2001.

[81] J.S. Marques and J.M. Lemos. Tracking of moving objects with multiple models using Gaussian mixtures. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3317–3320, Phoenix, March 1999.

[82] M. Maurer, R. Behringer, S. Fürst, F. Thomanek, and E.E. Dickmanns. A compact vision system for road vehicle guidance. In *Int. Conf. on Pattern Recognition*, volume 3, pages 313–317, August 1996.

[83] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[84] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J.Dayan. Interacting multiple model methods in target tracking: A survey. *IEEE Transactions on Aerospace and Electronic Systems*, 34(1), January 1998.

[85] Robert McGill, John W. Tukey, and Wayne A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[86] K. Murphy. Switching Kalman filters. Technical report, U.C. Berkeley, 1998.

[87] Kevin P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1015–1021, 2000.

[88] Christian Musso, Nadia Oudjane, and Francois Le Gland. *Sequential Monte Carlo Methods in Practice*, chapter Improving Regularised Particle Filters, pages 247–271. Springer-Verlag New York, 2001.

[89] Ben North and Andrew Blake. Learning dynamical models using expectation-maximisation. In *ICCV*, pages 384–389, 1998.

[90] J.J. Oliver, Baxter R.A., and Wallace C.S. Unsupervised Learning using MML. In *Machine Learning: Proceedings of the Thirteenth International Conference (ICML 96)*, pages 364–372. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[91] R.R. Pitre, V.P. Jilkov, and X.R. Li. A comparative study of multiple-model algorithms for maneuvering target tracking. In Ivan Kadar, editor, *Proc. 2005 SPIE Conf. Signal Processing, Sensor Fusion, and Target Recognition XIV*, volume 5809, pages 549–560, Orlando, March 2005.

[92] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1998.

[93] David Reynard, Andrew Wildenberg, Andrew Blake, and John A. Marchant. Learning dynamics of complex motions from image sequences. In *ECCV (1)*, pages 357–368, 1996.

[94] Maria Isabel Ribeiro. Notas dispersas sobre filtragem de Kalman. Technical report, Centro de Análise e Processamento de Sinais, June 1989.

[95] S. Richardson and P.J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society (Series B)*, 59(4):731–758, 1997.

[96] Jens Rittscher, J. Kato, S. Joga, and Andrew Blake. A probabilistic background model for tracking. In *ECCV (2)*, pages 336–350, 2000.

[97] S. Roberts, C. Holmes, and D. Denison. Minimum entropy data partitioning using Reversible Jump Markov Chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):909–915, August 2001.

[98] S.J. Roberts, D. Husmeier, I. Rezek, and W. Penny. Bayesian Approaches To Gaussian Mixture Modelling. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(11):1133–1142, 1998.

[99] Yong Rui and Yunqiang Chen. Better proposal distributions: Object tracking using unscented particle filter. In *Conference in Computer Vision and Pattern Recognition*, volume 2, pages 786–793, 2001.

[100] Robert E. Schapire and Yoram Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[101] Jens Schick and Ernst D. Dickmanns. Simultaneous estimation of 3D shape and motion of objects by computer vision. In *IEEE Workshop on Visual Motion*, pages 256–261, Princeton, NJ, October 1991.

[102] Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, March 1978.

[103] Kevin Smith and Daniel Gatica-Perez. Order matters: A distributed sampling method for multi-object tracking. In *British Machine Vision Conference (BMVC)*, number 11, pages 25–32, 2004.

[104] Stephen P. Smith and Anil K. Jain. A test to determine the multivariate normality of a data set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):757–761, September 1988.

[105] I.M. Sobol. *Método de Montecarlo*. Mir, 1976.

[106] M.A. Sotelo, J. Nuevo, L.M. Bergasa, M. Ocana, I. Parra, and D. Fernández. Road vehicle recognition in monocular images. In *IEEE International Symposium on Industrial Electronics*, volume 4, pages 1471–1476, Dubrovnik, Croatia, June 2005.

[107] G. Stein, O. Mano, and A. Shashua. A robust method for computing vehicle ego-motion. In *IEEE Intelligent Vehicles Symposium (IV2000)*, pages 362–368, Dearborn,MI, October 2000.

[108] Zehang Sun, G. Bebis, and R. Miller. On-road vehicle detection using Gabor filters and support vector machines. In *International Conference on Digital Signal Processing*, volume 2, pages 1019–1022, 2002.

[109] Zehang Sun, George Bebis, and Ronald Miller. On–road vehicle detection using evolutionary Gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):125–137, June 2005.

[110] Zehang Sun, George Bebis, and Ronald Miller. On–road vehicle detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):694–711, May 2006.

[111] Zehang Sun, Ronald Miller, George Bebis, and David DiMeo. A real-time pre-crash vehicle detection system. In *Sixth IEEE Workshop on Applications of Computer Vision*, pages 171–176, Orlando, FL, December 2002.

[112] T.K. ten Kate, M.B. van Leewen, S.E. Moro-Ellenberger, B.J.F. Driessen, A.H.G. Versluis, and F.C.A. Groen. Mid–range and distant vehicle detection with a mobile camera. In *IEEE Intelligent Vehicles Symposium*, pages 72–76, Parma, Italy, June 2004.

[113] Prithiraj Tissainayagam and David Suter. Performance measures for assessing contour trackers. *International Journal of Image and Graphics*, 2(2):343–359, 2002.

[114] S. Julier und J. K. Uhlmann. A new extension of the Kalman filter to non-linear systems. In *The Proceedings of AeroSense: The 11th Int. Symp. on Aerospace/Defense Sensing, Simulation and Controls*, Orlando, Florida, 1997.

[115] Rudolph van der Merwe, Nando de Freitas, Arnaud Doucet, and Eric Wan. The unscented particle filter. Technical Report CUED/F-INFENG/TR380, Cambridge University Engineering Department, August 2000.

[116] Rudolph van der Merwe, Nando de Freitas, Arnaud Doucet, and Eric Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems 13*, Nov 2001.

[117] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.

[118] B. Walsh. Markov chain Monte Carlo and Gibbs sampling, April 2004.

[119] E.A. Wan and R. van der Merwe. The unscented Kalman filter for nonlinear estimation. In *IEEE Symposium on Adaptive Systems for Signal Processing, Communication and Control (AS-SPCC '00)*, pages 153–158, Lake Louise, Alberta, Canada, 2000.

[120] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, March 1963.

[121] Robert C. Warren. A Bayesian track-before-detect algorithm for IR point target detection. hybrid information systems, first international workshop on hybrid intelligent systems, adelaide, australia, december 11-12, 2001, proceedings. In *HIS*, Advances in Soft Computing, pages 539–553, Adelaide, Australia, December 11–12 2002. Physica-Verlag.

[122] Greg Welch and Gary Bishop. An introduction to the Kalman filter. Course 8 in Siggraph, 2001.

[123] A.P. Wildenberg. *Learning and Initialisation for Visual Tracking.* PhD thesis, University of Oxford, Robotics Research Group. Departament of Engineering Science., 1998.

[124] T. Yamamoto and R. Chellappa. Shape and motion driven particle filtering for human body tracking. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 3 (ICME '03)*, pages 61–64, Washington, DC, USA, 2003. IEEE Computer Society.

[125] Chih-Hsien Yeh and Yung-Hsin Chen. Development of vision–based lane and vehicle detecting systems via the implementation with a dual–core DSP. In *IEEE Intelligent Transportation Systems Conference*, pages 1179–1184, Toronto, Canada, September 2006.

[126] Yan Zhang, Stephen J. Kiselewich, and William A. Bauson. Legendre and Gabor moments for vehicle recognition in forward collision warning. In *IEEE Intelligent Transportation Systems Conference*, pages 1185–1190, Toronto, Canada, September 2006.

[127] Qiang Zhu, Shai Avidan, Mei-Chen Yeh, and Kwang-Ting Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE Computer Society Conference on Computer vision and Pattern Recognition*, volume 2, pages 1491–1498, June 2006.

[128] Ying Zhu, Dorin Comaniciu, Visvanathan Ramesh, Martin Pellkofer, and Thorsten Koehler. An integrated framework of vision-based vehicle detection with knowledge fusion. In *IEEE Intelligent Vehicles Symposium*, pages 199–204, Las Vegas, USA, June 2005.

Publications

- D. Ponsa. Anàlisi del Flux Òptic Utilitzant Mètodes Probabilistics. *Master Thesis, Universitat Autònoma de Barcelona*, September 1998.
- A. Pujol, A. Solé, D. Ponsa, J. Varona, and J.J. Villanueva. Robust Rotational Invariant Coin Shape Identification. In *Proceedings of Workshop on European Scientific and Industrial Collaboration on Promoting Advanced Technologies in Manufacturing (WESIC)*, 1998.
- D. Ponsa and J. Vitrià. Mobile Robots Monitoring System Using an Agent-Oriented Approach. In *Proceedings of the 1er Congrès Català d'Intelligència Artificial*, Tarragona (Spain), 1998.
- D. Ponsa and J. Vitrià. Mobile Monitoring System Using an Agent-Oriented Approach. In *Proceedings of the VIII Symposium Nacional de Reconocimiento de Formas y Análisis de Imágenes*, Bilbao (Spain), January 1999.
- A. Solé, A. López, C. Cañero, J. Vitrià, and P. Radeva. Regularized EM. In *Proceedings of the VIII Symposium Nacional de Reconocimiento de Formas y Análisis de Imágenes*, Bilbao (Spain), January 1999.
- D. Ponsa and J. Vitrià. Mobile Monitoring System Using an Agent-Oriented Approach. In *VII IEEE International Conference on Emerging Technologies and Factory Automation Proceedings ETFA'99*, volume 2, pp. 1511-1515, Barcelona (Spain), 1999.
- A. Pujol, A. Solé, D. Ponsa, J. Varona, and J.J. Villanueva. Satellite Image Setmentation Through Rotational Invariant Feature Eigenvector Projection. *Machine Vision & Advanced Image Processing in Remote Sensing*, pp. 317-327, January 1999.
- D. Ponsa, A. Solé, A. López, C. Cañero, J. Vitrià, and P. Radeva. Regularized EM. *Pattern Recognition Applications*, IOS Press, pp. 69-77, January 2000.
- D. Ponsa. A Model Based Pedestrian Tracking Review. *CVC Technical Report #69*, CVC(UAB), 2001.
- F. Lumbreras, X. Roca, D. Ponsa, J. Martínez, S. Sánchez, C. Antens, and J.J. Villanueva. Visual Inspection of Safety Belts. In *Proceedings of International Conference on Quality Control by Artificial Vision (QCAV)*, volume 2, pp. 526-531. *Best poster award*, France 2001.

- D. Ponsa and X. Roca. Unsupervised Parameterisation of Gaussian Mixture Models. In *Topics in Artificial Intelligence, LNAI 2504*, pp. 388-398, Castelló (Spain), October 2002.
- D. Ponsa and X. Roca. A Novel Approach to Generate Multiple Shape Models. In *Proceedings of Workshop on Articulated Motion and Deformable Models, LNCS 2492*, pp. 80-91, Palma de Mallorca (Spain), November 2002.
- D. Ponsa, R. Benavente, F. Lumbreras, J. Martínez, and X. Roca. Quality control of safety belts by machine vision inspection for real-time production. *Optical Engineering, 42:1114-1120 (IF: 0.877)*, January 2003.
- D. Ponsa and X. Roca. Multiple Model Approach to Deformable Shape Tracking. In *Proceedings of of the 1st Iberian Conf. on Pattern Recognition and Image Analysis, LNCS 2652*, pp. 782-792, Port d'Andratx (Spain), June 2003.
- D. Ponsa, A. López, J. Serrat, F. Lumbreras, and T. Graf. Multiple Vehicle 3D Tracking Using an Unscented Kalman Filter. In *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems*, pp. 1108–1113, Vienna (Austria), September 2005.
- D. Ponsa, A. López, F. Lumbreras, J. Serrat, and T. Graf. 3D Vehicle Sensor Based on Monocular Vision. In *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems*, pp. 1096–1101, Vienna (Austria), September 2005.
- C. Julià, J. Serrat, A. López, F. Lumbreras, and D. Ponsa. Motion Segmentation through Factorization. Application to Night Driving Assistance. In *Proceedings of International Conference on Computer Vision Theory and Applications*, volume 2, Setúbal (Portugal), February 2006.
- D. Gerónimo, A. Sappa, A. López, and D. Ponsa. Pedestrian Detection Using AdaBoost Learning of Features and Vehicle Pitch Estimation. In *Proceedings of the 6th IASTED Internation Conference on Visualization, Imaging and Image Processing*, pp. 400-405, Palma de Mallorca (Spain), August 2006.
- D. Gerónimo, A. López, A. Sappa, and D. Ponsa. Model Features and Horizon Line Estimation for Pedestrian Detection in Advanced Driver Assistance Systems. In *1st CVC Internal Workshop, Computer Vision: Progress of Research and Development*, pp. 99-104, Bellaterra (Spain), October 2006.
- D. Ponsa, A. López, F. Lumbreras, and J. Serrat. A Proposal of a Monocular Multiple Vehicle 3D Tracking System. In *1st CVC Internal Workshop, Computer Vision: Progress of Research and Development*, pp. 111-116, Bellaterra (Spain), October 2006.
- D. Gerónimo, A. Sappa, A. López, and D. Ponsa. Adaptive Image Sampling and Windows Classification for On-board Pedestrian Detection. In *Vision Systems in the Real World: Adaption, Learning, Evaluation, LNCS*, Bielefeld (Germany), March 2007.

- D. Ponsa and A. López. Feature Selection Based on a New Formulation of the Minimal-Redundancy-Maximal-Relevance. In *Proceedings of the 3rd Iberian Conf. on Pattern Recognition and Image Analysis, LNCS 4477*, volume 1, pp. 47–54, Girona (Spain), June 2007.

- D. Ponsa and A. López. Vehicle Trajectory Estimation based on Monocular Vision. In *Proceedings of the 3rd Iberian Conf. on Pattern Recognition and Image Analysis, LNCS 4477*, volume 1, pp. 587–594, Girona (Spain), June 2007.

- D. Gerónimo, A. Sappa, A. López, and D. Ponsa. Haar Wavelets and Edge Orientation Histograms for On–Board Pedestrian Detection. In *Proceedings of the 3rd Iberian Conf. on Pattern Recognition and Image Analysis, LNCS 4477*, volume 1, pp. 418–425, Girona (Spain), June 2007.

- D. Ponsa and A. López. Cascade Of Classifiers for Vehicle Detection. In *Proceedings on Advanced Concepts for Intelligent Vision Systems, LNCS*, Delft (Netherlands), August 2007.

- A. Sappa, F. Dornaika, D. Ponsa, D. Gerónimo, and A. López. An Efficient Approach to On-Board Stereo Vision System Pose Estimation. *IEEE Transactions on Intelligent Transportation Systems* (accepted under review).

Glossary of Notation

$\mathbf{0}_N$	vector of zeros of length N, 10
$\mathbf{1}_N$	vector of ones of length N, 14
$\mathbf{A}$	system matrix, 19
$B_{n,o}(s)$	B-spline basis function, 11
$\mathbf{H}$	observation matrix, 19
$\mathbf{I}$	identity matrix, 14
$\mathcal{L}$	linear shape model, 13
$\mathcal{M}$	B-spline curve metric matrix, 13
$\mathcal{N}$	Gaussian distribution, 16
$p(\mathbf{x}_t \vert \mathbf{y}_{1:t})$	probability density function of $\mathbf{x}_t$ given $\mathbf{y}_{1:t}$, 18
$p_{\mathcal{N}}(\mathbf{x})$	Normal distribution density, 20
$\widetilde{p_N}(\mathbf{x})$	point mass distribution, 26
$\widehat{p_N}(\mathbf{x})$	point distribution density, 26
$\mathbf{Q}$	system noise covariance matrix, 19
$\mathbf{q}^x$	x coordinates of the control points in $\mathbf{q}$, 14
$\mathbf{R}$	observation noise covariance matrix, 19
^{T}	superscript denoting transpose, 10
$\tilde{w}_t$	normalised importance weight, 26
w_t	unnormalised importance weight, 26
X	constant, 10
$\mathbf{X}$	matrix, 13

$\mathbf{X}(s)$	function matrix, 10
x	variable, 10
$x(s)$	function, 10
$\mathbf{x}$	vector (column). State, 2
$\mathbf{x}(s)$	function vector, 10
$\bar{\mathbf{x}}$	mean vector, 13
$\mathbf{x}_t$	vector (state) at instant t, 2
$\hat{\mathbf{x}}_{t_1\|t_2}$	estimation of $\mathbf{x}$ at t_1 using observations up to t_2, 19
$\mathbf{x}_{1:N}$	set of N vectors, 90
$\{\mathbf{x}_i\}_{i=1}^N$	set of N vectors, 13
$\mathbf{x}^{(i)}$	i-th sample, 23
$\{\mathbf{x}^{(i)}\}_{i=1}^N$	particle set, 26
$\mathbf{y}$	observation vector, 2

List of Acronyms

ADAS	Advanced Driver Assistance Systems, 1
AGMMFA	Adaptive Gaussian Mixture Model Fitting Algorithm, 91
AR(n)	Auto–Regressive process of order n, 16
BIC	Bayesian Information Criterion, 94
BM	Brownian motion, 167
CBM	Constrained Brownian Motion, 167
CDO	Critically Damped Oscillation, 169
COC	Cascade of Classifiers, 118
DBT	Detect Before Track, 135
EKF	Extended Kalman Filter, 20
EM	Expectation Maximisation, 86
FP	False Positives, 115
GMM	Gaussian Mixture Model, 85
GNN	Global Nearest Neighbour, 135
GPBF	Gaussian Pseudo Bayesian Filter, 102
GT	Ground Truth, 94
HCA	Hierarchical Clustering Algorithm, 89
HMM	Hidden Markov Model, 105
IMM	Interacting Multiple Model filter, 102
JMLS	Jump Markov Linear System, 101
JMS	Jump Markov System, 86
KF	Kalman Filter, 18
KF1	Kalman Filter using first order dynamics, 137

KF2	Kalman Filter using second order dynamics, 137
KS	Kalman Smoother, 137
MCE	Mean Contour Error, 40
MHT	Multiple Hypothesis Tracking, 135
MPD	Mean Paired Difference, 190
MSE	Mean Square Error, 140
MST	Minimal Spanning Tree, 91
NN	Neural Network, 116
PCA	Principal Component Analysis, 13
PDF	Probability Density Function, 23
PF	Particle Filter, 23
PS	Partitioned Sampling, 73
RBPF	Rao–Blackwell Particle Filter, 65
SIS	Sequential Importance Sampling, 26
SISR	Sequential Importance Sampling with Resampling, 27
SNR	Signal–to–Noise Ratio, 40
STM	State Transition Matrix, 103
SVM	Support Vector Machine, 116
SVS	Subspace of Valid Shapes, 85
TBD	Track Before Detect, 134
UKF	Unscented Kalman Filter, 22
UKPF	Unscented Kalman Particle Filter, 62