BAYES FILTERS WITH IMPROVED MEASUREMENTS FOR VISUAL OBJECT TRACKING

Dissertation
ZUR ERLANGUNG DES MATHEMATISCH-NATURWISSENSCHAFTLICHEN DOKTORGRADES
"DOKTOR RERUM NATURALIUM" DER GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

vorgelegt von
Guoliang Liu
aus
Shandong

Göttingen, 2012

Referentin/Referent: Prof. Florentin Wörgötter
Koreferentin/Koreferent: Prof. Dieter Hogrefe
Tag der mündlichen Prüfung: 20. March 2012

Abstract

Visual object tracking uses cameras to track target objects in the environment, which has many applications nowadays, such as intelligent surveillance, medical care, intelligent transportation and human-machine interaction. However, it is still a challenging task because of background noises, occlusions, illumination changes and fast motion. The goal of this dissertation is to improve measurements in Bayesian filtering frameworks for visual object tracking as follows:

First, we combine multiple visual cues to improve the measurement for lane tracking. The lane is modeled by a linear-parabolic shape, which is a trade-off between accuracy of the fit and robustness with respect to image artifacts. In contrast to previous methods for linear-parabolic lane tracking, we use not only the color and edge information, but also the gradient orientation as visual cues. The lane tracking becomes a statistical reference problem when these local visual cues are available. The probabilistic distribution of lane parameters are estimated from the visual cues by multiple kernel density estimation, which is proved to be very robust to the image noise. Furthermore we use this probabilistic distribution function as the measurement model of the partitioned particle filter to update lane parameters. The experiments show that our novel lane tracking framework has its strength in a new combination and improvement of various advanced methods.

Second, we use color invariant histograms to improve the measurement for rigid object tracking. Color histograms have become popular and important descriptors for object tracking, due to their simplicity, effectiveness and efficiency. However, they suffer from illumination changes, e.g., the RGB color histogram is the most prevalently used histogram, but it has no invariance properties to illumination changes. This paper addresses this problem by: a) studying the invariance properties and the distinctiveness of color histograms; b) evaluating the color histograms on large benchmark datasets; c) studying the effects of the kernel mask which adds the spatial information to the color histogram; d) investigating three state-of-the-art object tracking algorithms for evaluations: the integral histogram based exhaustive search, the kernel based mean shift and the particle filter. The results reveal that color histograms which have invariance properties can improve the performance of object tracking. If no prior knowledge about the environment of the dataset is available, the HSV, Spherical and nRGB histograms are recommended.

Third, we employ multiple sensors to improve the measurement for moving object tracking. Tracking systems can be more accurate, complete and robust by using fused information from multiple sensors. Therefore, we develop a new filter called central difference information filter (CDIF) for nonlinear estimation and sensor fusion, which has fewer predefined parameters as compared to the unscented information filter (UIF) which was introduced in the literature recently. In addition, we introduce the square-root extensions of the CDIF and UIF to improve the numerical stability, e.g., improved numerical accuracy, double order precision and preservation of symmetry.

In summary, we have proposed three methods to improve the measurement for robust object tracking, i.e., multiple visual cues combination, color invariant histograms and multiple sensor fusion. We believe that the new ideas and theoretical insights presented in this thesis, will open new ways of research for future algorithms and applications.

Acknowledgements

First and foremost I want to express my sincere gratitude and thanks to my supervisor Prof. Florentin Wörgötter for supporting me over all these years, both financially and mentally, along the way. He created opportunities and gave me the freedom to explore unknown scientific fields. I also had lots of fun with him, e.g., skiing trip, sauna, home made barbecue and so on. Many of these are the first time experiences for me. Special thanks goes to Dr. Irene Markelić for encouraging support and advice in the research. This thesis will not be possible without their help and assistance. I also thank all the members of the committee and the reviewers of this dissertation.

Furthermore, I would like to thank all my colleagues in the BCCN group, Dr. Tomas Kulvičius, Jan-Matthias Braun, Dr. Sinan Kalkan, Dr. Babette Dellen, Dr. Poramate Manoonpong, Dr. Minija Tamosiunaite, Dr. Kejun Ning, Xiaofeng Xiong, Guanjiao Ren, Eren Aksoy, Alexey Abramov, Dr. Christoph Kolodziejski, Faramaz Faghihi, Mohamad Javad Aein, Markus Schoeler, Sakyasingha Dasgupta, Martin Biehl, Dr. Frank Hesse, Waldemar Kornewald, Michael Fauth, Dr. Alejandro Gabriel Agostini, Thomas Wanschik, Jeremie Papon, Johannes Dörr, Vishal Patel, Harm-Friedrich Steinmetz, Simon Reich, Timo Nachstedt, Steffen Zenker, Dennis Goldschmidt, Birk Urmersbach and our secretary Ursula Hahn-Wörgötter.

Last but not least, I wish to thank my family who have always supported me since the beginning of my life. I would also like to thank all my friends for helping me maintain a life outside work.

Contents

Abstract 3

Acknowledgements 5

1 Introduction 11
- 1.1 Motivation . . . 11
- 1.2 Multiple Visual Cues . . . 12
- 1.3 Color Invariant Histograms . . . 14
- 1.4 Multiple Sensor Fusion . . . 15
- 1.5 Contributions . . . 16
- 1.6 Thesis Outline . . . 17

2 Bayes Filters 19
- 2.1 General Form of Bayes Filters . . . 20
- 2.2 Kalman Filters . . . 21
  - 2.2.1 Linear Kalman Filter . . . 21
  - 2.2.2 Extended Kalman Filter . . . 23
  - 2.2.3 Unscented Kalman Filter . . . 24
  - 2.2.4 Central Difference Kalman Filter . . . 26
- 2.3 Information Filters . . . 31
  - 2.3.1 Linear Information Filter . . . 31
  - 2.3.2 Extended Information Filter . . . 32
  - 2.3.3 Unscented Information Filter . . . 32
- 2.4 Particle Filters . . . 32
  - 2.4.1 Standard Particle Filter . . . 33
  - 2.4.2 Partitioned Particle Filter . . . 38
- 2.5 Conclusion . . . 39

3 Multiple Visual Cues for Lane Tracking 41
3.1 Introduction 41
3.2 Related Work 42
  3.2.1 Lane Model 42
  3.2.2 Estimation Method 43
  3.2.3 Measurement Model 44
3.3 Local Image Descriptors 44
3.4 Lane Parameter Estimation by Kernel Density 45
  3.4.1 Line Model 45
  3.4.2 Parabolic Model 46
  3.4.3 Linear-Parabolic Model 47
3.5 Partitioned Particle Filter for Lane Detection and Tracking 49
  3.5.1 Multiple Lane Model 49
  3.5.2 State Definition 49
  3.5.3 Image Preprocessing 50
  3.5.4 Initialization Samples 50
  3.5.5 Linear Part Estimation 51
  3.5.6 Parabolic Part Estimation 52
3.6 Results 53
  3.6.1 PPF-Kernel *vs.* PPF-NoKernel 54
  3.6.2 PPF-Kernel *vs.* PF-Kernel 55
  3.6.3 The Computational Cost 56
3.7 Conclusion 57

4 Color Invariant Histograms 59
4.1 Introduction 59
4.2 Illumination Changes 60
  4.2.1 Diagonal-Offset Model 61
  4.2.2 Definition of Illumination Changes 61
4.3 Color Histograms 62
4.4 Distance Metrics 64
4.5 Experimental Setup 65
  4.5.1 Exhaustive Search 65
  4.5.2 Mean Shift 66
  4.5.3 Particle Filter 67
  4.5.4 Evaluation Criteria 67
  4.5.5 Datasets 68
4.6 Results 69

4.6.1 Exhaustive Search . . . 69
4.6.2 Mean Shift . . . 70
4.6.3 Particle Filter . . . 71
4.7 Conclusion . . . 71

5 Multiple Sensor Fusion 73
5.1 Introduction . . . 73
5.2 Central Difference Information Filter . . . 74
5.2.1 Stirling's Interpolation . . . 75
5.2.2 Prediction . . . 76
5.2.3 Measurement Update . . . 77
5.2.4 Global Information Fusion . . . 78
5.3 Square-Root Central Difference Information Filter . . . 79
5.3.1 SRCDIF for State Estimation . . . 79
5.3.2 SRCDIF for Multiple Sensor Fusion . . . 82
5.4 Square-Root Unscented Information Filter . . . 82
5.5 Experiment on Sensor Fusion . . . 84
5.5.1 Space-Vehicle Tracking . . . 84
5.5.2 Bearing-Only Tracking . . . 86
5.6 Experiment on Numerical Stability . . . 88
5.7 Conclusion . . . 90

6 General Discussion 91
6.1 Summary . . . 91
6.2 Discussion . . . 93
6.3 Future Works . . . 94

A Cholesky Factorization for Drawing Error Ellipses 97

B Inverse Perspective Mapping 99

Bibliography 101

Curriculum Vitae 111

Chapter 1

Introduction

1.1 Motivation

Visual object tracking is a process of continuously estimating the state of an object in an image given prior states from previous frames [Zimmermann, 2008]. The state of an object can be position, velocity, shape, size and so on. This process is a very important step for video analysis applications, such as motion recognition, automated surveillance, video indexing, traffic monitoring and vehicle navigation [Yilmaz et al., 2006]. Object tracking can be difficult due to noise, partial or full occlusions, complex object motion, and shape and illumination changes. In order to deal with these difficulties, object tracking algorithms require robust measurements which can describe the appearance of objects correctly when the environmental conditions change. Therefore, our main goal in this dissertation is to push forward advances in visual object tracking by focusing on improving the measurement process. More precisely we investigate the use of multiple visual cues, color invariant features and multiple sensors.

Multiple visual cues can give a more complete description of objects for tracking. For instance, humans can visually distinguish between objects using features, such as color, texture, shape and size. It can be difficult or even impossible to recognize an object using only one visual cue. For example, two colored balls can be of the same size and shape but differ in their colors. If we only know the size or shape of these balls, we can not distinguish them. Consequently, to track an object in a complex environment, it is reasonable to use *multiple visual cues* to give a comprehensive description of the target object.

Color invariant features can improve the performance of object tracking under various lighting conditions, such conditions can be different light sources, shadows and highlights. Cameras are passive sensors, sensing light from the environment. The same object under different lighting conditions can have dissimilar appearances in images recorded by the same camera. In other words, the appearance of the target object can change distinctly under these different lighting conditions during tracking. Therefore, color invariant features which are robust to lighting condition changes are desired for object tracking.

Multiple sensors can provide complementary and redundant information that can contribute to the tracking performance. Such sensors can consist of cameras only, forming a smart camera network, or cameras combined with other sensors, e.g., acoustic, infrared and thermal sensors, forming a multi-model sensor network [Aghajan and Cavallaro, 2009]. Such a sensor network can offer multiple views, multiple properties of target objects and long distance tracking, whereas a single camera gives a single visual view information only. As a result, object tracking can be improved by fusing information from multiple sensors.

The measurements improved by multiple visual cues, color invariant features and multiple sensors, can be used in a *Bayesian filtering* framework for object tracking. As mentioned, the goal of object tracking is to estimate the state of objects, given some prior knowledge. Hence, object tracking can be described as a statistical inference process. One of the most popular methods in the literature accomplishing such a process is *Bayesian filtering*. Bayesian filtering for object tracking comprises two steps: state prediction and measurement update. The state prediction uses a *motion model* of the target object to predict possible states. In practice this motion model is often unknown, but can be approximated by setting it to a random-walk [Isard and Blake, 1998], constant velocity, constant acceleration [Yilmaz et al., 2006], etc. The measurement update employs the *measurement model* to update the predicted state of the object. This *measurement model* is a function that describes the mathematical relation between the state and the measurements. In this way, Bayes filters encode both motion and measurement information of the object.

As we mentioned in the beginning of this section, this dissertation focuses on improvements of the measurements in Bayesian filtering frameworks for object tracking. We demonstrate these improvements in three scenarios. First, we focus on combining multiple visual cues to improve the performance of a lane tracker which we choose as application. Second, we present an evaluation framework for color invariant histograms. Third, we introduce a new Bayesian filtering framework for sensor fusion which has better numerical characteristics than previous work. We will introduce each framework in the following sections.

1.2 Multiple Visual Cues

In this section we introduce the combination of multiple visual cues for object tracking in detail. Visual cues are representations of appearance features of objects in images. These visual cues can be color, shape, texture, etc. Most modern cameras employ the RGB (red, green blue) color space to represent colors in a digital image. The basic element in an image is the pixel, which is assigned a digitalized RGB color vector. The color of a pixel, and the color difference between this pixel and its surrounding pixels are basic visual cue elements, which are sometimes called *local image descriptors* [Dahyot, 2009]. If such an image is interpreted as a 2D function, we can compute the gradient for each pixel, which is characterized by its direction and magnitude. Based on these basic local image descriptors, we can compute higher level visual cues for objects. e.g., shape and texture. For instance, Dahyot [2009] has proposed a method to estimate straight lines from local image descriptors.

As we mentioned in Section 1.1, a single visual cue is usually not sufficient to distinguish the target from the environment. That is why multiple visual cues are required to have sufficient measurements of objects. This leads to the following questions:

- *which visual cues should be used?* First of all, visual cues should be representative, which means that they should represent the main features of objects. Second, they should be distinguishable, which means they should represent features specific to the target object. Third, visual cues should be computable, which means they can be estimated from basic visual cue elements. Last but not the least, they should be robust against geometric distortions and lighting changes, which means they should be constant or change slowly when geometric distortions or lighting changes occur.

- *How to estimate these visual cues from basic visual cue elements?* One of the criteria to choose visual cues is that it should be possible to compute visual cues from the basic visual cue elements. Not all visual cues are computable using current techniques, due to the complexity of algorithms, limited computational power, etc. For example, if the computation of one visual cue takes a long time, it can not be used for applications that require real-time implementations.

- *How to combine these visual cues as measurements in a tracker?* After selecting of a set of appropriate cues, we must decide how to use them as measurements in a tracker. A tracker is an algorithm that uses prior knowledge about an object, e.g., measurements of an object from an initial frame, to find possible candidates of the object in the next frames. These measurements are usually combinations of various visual cues. The tracker can compare prior known measurements to current measurements. In this way, the target can be found in the current frame by looking for the best match.

For testing and evaluating our new ideas concerning multiple visual cues, we require a representative application. We chose the task of lane tracking. Our decision for doing so is based on the following reasons: first, lane tracking is susceptible to geometric distortions, e.g., parallel lanes in real world usually do not appear parallel in images. As a result, the visual cues in original images require additional techniques to describe the parallel feature of lane markers. For instance, we can warp original images to top-view images using inverse perspective mapping (IPM) as discussed in Appendix B. In this way, the geometric distortion is removed and the lane markers look parallel to each other in the resulting IPM images. Second, multiple visual cues are required for lane tracking. Since the main features of lane markers are color and shape, we have to consider how to combine them in the algorithm. Third, lane tracking is a good example to show how to estimate visual cues from basic visual cue elements. The lane shape is one of the main visual cues of lane markers. Therefore, how to estimate the lane shape from basic visual cue elements becomes an important issue.

To conclude this part, we have explained why using multiple visual cues as measurements can improve object tracking. Three essential questions related to this have been introduced. Based on these three questions, we motivated our decision to use the task of lane tracking in Chapter 3 as a representative application.

1.3 Color Invariant Histograms

Color is an important cue in the distinction and recognition of objects. For example, we humans, require color to segment the surface of objects, and then acquire higher level features, e.g., texture and structure [Swain and Ballard, 1991]. However, the color of an object depends not only on the object itself, e.g., its material, but also on the surrounding lighting conditions, e.g., the type of light source and shadows. For instance, a person under a lamp usually appears different than under sun light. Different lighting conditions can change the color of objects, and further influence higher level color features. If we use these color features that are susceptible to color changes for object tracking, the tracker might fail to track the target object when lighting condition changes. Therefore, color features that are invariant to lighting condition changes are preferable for robust object tracking.

One of the most commonly used color features is the color histogram. Given a discrete color space, a color histogram counts how often each color occurs in the image [Swain and Ballard, 1991]. A discrete color space is similar to the Euclidean space, but each point in the color space represents a color, whereas each point in the Euclidean space represents a point in the world. The most popular color space is the RGB color space, which includes three channels, i.e., red, green and blue. A color in the RGB color space is defined as a mix of three components. There are also some other color spaces, e.g., HSV (hue, saturation, and value), nRGB (normalized RGB), Opponent and Spherical color space [Gevers and Smeulders, 1999; van de Weijer et al., 2006]. Since color histograms are invariant to translation and rotation about the viewing axis [Swain and Ballard, 1991], they have been widely used in the field of object tracking [Birchfield, 1998; Comaniciu et al., 2000; Pérez et al., 2002; Khan et al., 2009].

To improve the performance of color histograms concerning illumination changes, the color histograms that are color invariant are preferred [Funt and Finlayson, 1995]. The color invariant property of color histogram depends on its color space. For instance, RGB color space is not color invariant, so the RGB color histogram has no color invariant properties. On the other hand, the nRGB color space is invariant to light intensity changes [van de Sande et al., 2010], such that the nRGB color histogram is more preferable than the RGB color histogram in the case of light intensity change.

Our work is to evaluate the performance of color histograms against illumination changes for object tracking. Most of previous works on color invariance are in the field of image classification [van de Sande et al., 2010; Burghouts and Geusebroek, 2009]. In contrast, we focus on object tracking. The goal is to analyze the color invariant properties of color histograms for the performance of object tracking, and to compare these color histograms on benchmark datasets with various lighting conditions.

1.4 Multiple Sensor Fusion

For some applications of visual object tracking, e.g., video surveillance and monitoring, the measurements of objects from a single camera or cameras are not sufficient. First of all, it is not possible for a single camera to see all areas at once [Collins et al., 2001]. For example, objects sometimes can be occluded by buildings and trees. A single camera has difficulty to solve such a problem because of its limited field of view. In order to expand the field of view, multiple cameras which are installed in different places can be used. Furthermore, cameras have difficulties to handle bad lighting conditions, e.g., bad weather or dark environment. This problem can be solved by using other sensors that are not influenced by lighting conditions, e.g., laser or thermal infrared sensors. Consequently, visual object tracking can benefit from measurements from multiple sensors, which provide complementary and redundant information.

To fuse informations from multiple sensors, one of the most commonly used methods is Bayesian filtering [Aghajan and Cavallaro, 2009]. As we mentioned in Chapter 1, Bayes filters have two steps: prediction and update. The state is first predicted using the motion model of the target object, then the information from multiple sensors are fused in the update step. Therefore, the update step is very important, since it can affect the computational cost and the structure of the sensor network. Durrant-Whyte [2001] recommended the *information filter*, which is one of the Bayesian filters, because the update step in the information filter for multiple sensor fusion is simply achieved by a linear summation of information contribution from each sensor node.

In practical terms, standard information filters are known to be susceptible to numerical errors due to limited word-length arithmetics. As will be discussed in Chapter 2, information filters use the information vector and information matrix to represent the Gaussian distribution of state variables. In addition, the information matrix is the inverse of the covariance of state variables. In order to derive this information matrix, the covariance of state variables has to be symmetric and positive definite. However, the numerical errors introduced by limited word-length arithmetics can destroy the symmetric and positive definite properties of the covariance, which can cause the filter to diverge. To improve the numerical stability of information filters, a square-root of the covariance can be used in information filters instead of full covariance. The square-root forms can improve numerical accuracy, and yield twice the effective precision of the conventional approach

[Kaminski et al., 1971; Arasaratnam and Haykin, 2008].

Our work improves the numerical stability of nonlinear information filters using square-root forms. In the scenario of object tracking using multiple sensors, the motion model and measurement model can be nonlinear. In this case, a nonlinear information filter is required, e.g., the unscented information filter (UIF) [Lee, 2008a]. In this dissertation, we first introduce an alternative, nonlinear information filter, which is called central difference information filter (CDIF). We show that the CDIF has fewer predefined parameters than the UIF. Furthermore, we introduce our square-root extensions of UIF and CDIF, which have better numerical stability than the original ones.

1.5 Contributions

The contributions of this thesis can be summarized as follows:

- **Multiple kernel density for parabolic and linear-parabolic shape estimation**

  We propose a new nonparametric method for parabolic and linear-parabolic shape estimation using multiple kernel density. Dahyot [2009] has utilized multiple kernel density to estimate straight lines from local image descriptors, i.e., gradient magnitude, gradient direction and alignment. Here we extend her work to the parabolic and linear-parabolic cases. In contrast to straight lines, the gradient angle of parabolic shape is not constant, so additional numerical methods are required to derive the probabilistic distribution of parabolic and linear-parabolic lane parameters. This work has been published in [Liu et al., 2011a].

- **Partitioned particle filter for linear-parabolic lane tracking**

  For lane tracking, we utilize the partitioned particle filter with a novel measurement function derived from multiple kernel density estimation. Previous work for linear-parabolic lane tracking usually employs least squares estimation [Jung and Kelber, 2004, 2005] or Kalman filter [Lim et al., 2009]. The least squares method is sensitive to edge noise, and Kalman filter maintains only one hypothesis which leads to poor performance in a cluttered environment [Liu et al., 2010, 2011a]. In contrast, the partitioned particle filter maintains multiple hypotheses, and needs less particles for high dimensional states than the standard particle filter [MacCormick and Blake, 2000]. Furthermore, the measurement model of the partitioned particle filter can be defined as any function that describes how well particles fit visual cues, e.g., the probabilistic distribution of lane parameters modeled by the multiple kernel density estimation. Therefore, here we present a new framework for linear-parabolic lane tracking using a partitioned particle filter. This work has been published in [Liu et al., 2010, 2011a].

- **Evaluating color invariant histograms for object tracking**

Color histograms have been widely used for visual object tracking, due to their simplicity and efficiency. However, color histograms are susceptible to illumination changes. As far as we know, none of the previous work considers color invariant properties of color histograms in the scenario of object tracking. Recently, van de Sande et al. [2010] have analyzed the color invariant properties of color descriptors for object and scene recognition. In this dissertation, we extend their work to the field of object tracking. The robustness and performance of color histograms for object tracking are evaluated on several open-access data sets.

- **Central difference information filter**

  We propose an alternative to the unscented information filter (UIF) for sensor fusion, which we call *central difference information filter* (CDIF). Recently, Lee [2008a] proposed the UIF for nonlinear estimation and sensor fusion, which employs a number of deterministic sigma-points to calculate the mean and covariance of a random variable which undergoes a nonlinear transformation. These sigma points are generated by the unscented transform in the UIF. In contrast, our CDIF employs Stirling's interpolation instead of the unscented transform to generate sigma points which has fewer predefined parameters. This work has been published in [Liu et al., 2011b].

- **Square-root sigma point information filters**

  The UIF and CDIF require the Cholesky factor of the covariance to calculate the sigma points, so the covariance must be positive definite. Therefore the numerical stability becomes an important issue in the UIF and CDIF. In order to improve the numerical stability, we introduce the square-root extensions of the UIF and CDIF. These square-root versions have better numerical properties than the original ones, e.g., improved numerical accuracy, double order precision and preservation of symmetry. We also show that the square-root unscented information filter (SRUIF) might lose the positive-definiteness due to the negative Cholesky update, whereas the square-root central difference information filter (SRCDIF) has only positive Cholesky updates. As a result, the SRCDIF is preferable to the SRUIF concerning numerical stability. This work has been published in [Liu et al., 2012].

1.6 Thesis Outline

We introduce the following Bayes filters in Chapter 2 as far as relevant for this thesis: Kalman filters, information filters and particle filters.

In chapter 3, we introduce our method for combining multiple visual cues in the application of lane tracking. We present how to use multiple kernel density to estimate the probabilistic distribution of lane parameters, i.e., straight line, parabolic and linear-parabolic shapes. Furthermore, we introduce a new lane tracking framework using a partitioned particle filter. The measurement model of this partitioned particle filter is derived from the probabilistic distribution function of lane parameters.

Chapter 4 presents our evaluation framework for the color invariant histogram based object tracking. We first analyze color invariant properties of various color spaces and corresponding color histograms, then compare the performance of the color histograms for object tracking. Three tracking algorithms are utilized for the evaluation: exhaustive search, mean shift and particle filter.

Chapter 5 describes the sigma-point information filters and their square-root extensions for nonlinear estimation and sensor fusion in the scenario of object tracking. We show that the CDIF uses less parameters than the UIF, but achieves a similar performance. Furthermore, the square-root extensions of the CDIF and UIF are derived which have better numerical characteristics. Finally, two classical simulation experiments are used to evaluate the proposed filters concerning sensor fusion and numerical stability.

Chapter 6 concludes our approaches and discusses our key results. The advantages of the proposed algorithms along with their limitations are discussed. We also identify some directions for future work.

Chapter 2

Bayes Filters

Since the main hypothesis of this dissertation is to improve the measurements in the Bayesian filtering framework for visual object tracking, we would like to introduce the theoretical background of Bayes filters in advance. Also, in this chapter, the advantages and disadvantages of three commonly used Bayes filters are analyzed and compared in detail.

In the literature, Bayes filters have been widely used for object tracking, due to their robustness and efficiency in dealing with the uncertainty in the tracking system. This uncertainty can be modeled by a probability function, e.g., Gaussian distribution. For instance, the position of an object at a discrete time is uncertain, then the probability distribution of the position can be modeled by a Gaussian distribution. One of the most famous Bayes filters is the Kalman filter, which was introduced by Kalman [1960]. The Kalman filter has one main assumption concerning the uncertainty in the system, i.e., the uncertainty of variables in the tracking system has a Gaussian distribution. In the Kalman filter, the moments (mean and covariance) are used to represent this Gaussian distribution. In contrast, the information filter, which is also called inverse covariance filter [Maybeck, 1979], employs information vector and information matrix (inverse of the covariance) to represent the Gaussian distribution. The differences in representing of Gaussian distributions, will lead to different algorithmic structures, which will be discussed in following sections. However, the distribution of the uncertainty in the tracking system sometimes is not the Gaussian distribution. In this case, a particle filter can be utilized, since the particle filter uses a number of particles to approximate the real probability distribution, which can have any profile [Arulampalam et al., 2002]. These three Bayes filters are popularly used for object tracking tasks, and they have various extensions to deal with the nonlinear, ill conditions, numerical stabilities, etc.

In Section 2.1, we start with an introduction of the basic idea of Bayesian filtering. Next we show the standard Kalman filter and its extensions in Section 2.2. Then the information filters are discussed in Section 2.3. Finally, we present the particle filters in Section 2.4.

2.1 General Form of Bayes Filters

The goal of Bayes filters is to estimate the current system state given prior knowledge, i.e., prior control inputs and measurements. For instance, the system state of a simple robot system can be the position and velocity of the robot. The control inputs can be the velocity, e.g., setting the velocity to be $5cm$ per second in the actuator. The measurements are from the sensors, e.g., cameras and lasers, which describe a momentary state of the environment [Thrun et al., 2005].

For a certain estimation problem, we first need to define the system state vector x that includes one or more variables of the system, e.g., position coordinates, velocities and acceleration. Then we can estimate the probability distribution of the current state x_t at time t using prior control inputs $u_{1:t}$ and measurements $z_{1:t}$:

$$Goal: \quad prob(x_t) = p(x_t|u_{1:t}, z_{1:t}), \tag{2.1}$$

where $prob$ is the abbreviation for the posterior distribution, and $1 : t$ means the discrete time from beginning until now. Here, the posterior $prob(x_t)$ incorporates the current measurement z_t. However, it has been proved to be useful to calculate the posterior before incorporating z_t:

$$Prediction: \quad \hat{prob}(x_t) = p(x_t|u_{1:t}, z_{1:t-1}). \tag{2.2}$$

This process is called *prediction*, which implies that we first derive a predicted state by considering the current control input u_t. The next step is to calculate the $prob(x_t)$ from $\hat{prob}(x_t)$ by incorporating the current measurement z_t using the Bayes rule, which is referred to as the *measurement update*:

$$Measurement\ update: \quad prob(x_t) = \eta\ p(z_t|x_t, z_{1:t-1}, u_{1:t})\ \hat{prob}(x_t), \tag{2.3}$$

where η is a constant normalizer, which ensures $0 \leq prob(x_t) \leq 1$. In this way, the whole process of the Bayesian filtering is separated into two individual steps: prediction and measurement update.

In practice, it is often impossible and not necessary to keep all the prior information to estimate the current state. To reduce the complexity of the estimation, we can assume that the current state is a complete summary of the part, which is known as the *Markov assumption* [Thrun et al., 2005]. Following this assumption, the prediction step in Eq. (2.2) and measurement update step in Eq. (2.3) can be rewritten as:

$$\begin{aligned} Prediction: \quad \hat{prob}(x_t) &= p(x_t|z_{1:t-1}, u_{1:t}) \\ &= \int p(x_t, x_{t-1}|z_{1:t-1}, u_{1:t})\ dx_{t-1} \\ &= \int p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t})\ p(x_{t-1}|z_{1:t-1}, u_{1:t})\ dx_{t-1} \\ &= \int p(x_t|x_{t-1}, u_t)\ prob(x_{t-1})\ dx_{t-1}, \end{aligned} \tag{2.4}$$

$$
\begin{aligned}
Measurment\ update: \quad prob(x_t) &= \eta\ p(z_t|x_t, z_{1:t-1}, u_{1:t})\ \hat{prob}(x_t) \\
&= \eta\ p(z_t|x_t)\ \hat{prob}(x_t).
\end{aligned} \tag{2.5}
$$

The Eq. (2.4) and Eq. (2.5) shape up the fundamental of Bayes filters. A remaining question is how to represent the probabilities in these two equations. As we mentioned at the beginning of this chapter, a commonly used distribution is the Gaussian distribution. Both the Kalman filter and information filter use the Gaussian distribution. In contrast, for other cases where the Gaussian distribution can not be employed, a particle filter can be used.

2.2 Kalman Filters

The Kalman filter introduced by Kalman [1960], is one of the most used Bayes filters for tracking. A number of variants of the Kalman filter has been developed to handle different scenarios. The Kalman filters represent the probability distribution of the state by the mean μ_t and the covariance P_t of the Gaussian distribution. The recursive estimation of the system state becomes the propagation of the moments.

2.2.1 Linear Kalman Filter

As we mentioned in the last section, the Bayes filtering has two steps: prediction and measurement update as shown in Eq. (2.4) and Eq. (2.5). For prediction, we require a state transition model (also called motion model) which describes the state transition from the last state x_{t-1} to the current state x_t. For measurement update, we also require a measurement model which describes the mathematical relation between the current state x_t and current measurement z_t. For the *linear Kalman filter* (LKF), both the state transition model and the measurement model are linear functions, which can be presented as

$$
Transition\ model: \quad x_t = Ax_{t-1} + Bu_t + \epsilon_t, \tag{2.6}
$$

$$
Measurement\ model: \quad z_t = Cx_t + \delta_t, \tag{2.7}
$$

where ϵ_t and δ_t are Gaussian noises: $\epsilon_t \sim \mathcal{N}(0, R_t)$ and $\delta_t \sim \mathcal{N}(0, Q_t)$, and A, B and C are matrices.

The state transition probability $p(x_t|x_{t-1}, u_t)$ in Eq. (2.4) and the measurement probability $p(z_t|x_t)$ in Eq. (2.5) can be calculated from Eq. (2.6) and Eq. (2.7) respectively:

$$
p(x_t|x_{t-1}, u_t) = \mathcal{N}(Ax_{t-1} + Bu_t, R_t), \tag{2.8}
$$

$$
p(z_t|x_t) = \mathcal{N}(Cx_t, Q_t). \tag{2.9}
$$

Then the predicted posterior $\hat{prob}(x_t)$ can be computed by substituting Eq. (2.8) into Eq. (2.4) and then integrating on x_{t-1}. Furthermore, the final estimated distribution $prob(x_t)$ is derived by substituting Eq. (2.9) into Eq. (2.5) and then timing the predicted posterior $\hat{prob}(x_t)$.

The whole algorithm for the LKF is shown in Algorithm 1. Because the LKF is a well studied algorithm, we do not present the mathematical derivations of the algorithm. A good derivation of the LKF can be found in [Thrun et al., 2005].

Algorithm 1 Linear Kalman filter

- Initialization: $\mu_0 = E(x_0)$, $P_0 = E\left((x_0 - \mu_0)(x_0 - \mu_0)^T\right)$.
- For $t = 1, \cdots, \infty$:
  1. Prediction:
  $$\hat{\mu}_t = A\mu_{t-1} + Bu_t \tag{2.10}$$
  $$\hat{P}_t = AP_{t-1}A^T + R_t \tag{2.11}$$
  2. Measurement update:
  $$K_t = \hat{P}_t C^T (C\hat{P}_t C^T + Q_t)^{-1} \tag{2.12}$$
  $$\mu_t = \hat{\mu}_t + K_t(z_t - C\hat{\mu}_t) \tag{2.13}$$
  $$P_t = (I - K_t C)\hat{P}_t \tag{2.14}$$

To gain an intuition for the mechanism of the Kalman filter, we give an example which tackles the so-called robot localization problem. A robot at time t has the state $X_t = (x_t, v_t, \alpha_t)^T$, where x_t, v_t and α_t are the robot's position, speed and acceleration respectively. The measurement Z_t measures the position. The state transition model and measurement model are:

$$Transition\ model: \quad X_t = AX_{t-1} + \epsilon_t \tag{2.15}$$

$$Measurement\ model: \quad Z_t = CX_t + \delta_t \tag{2.16}$$

where $A = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$, $C = [1, 0, 0]^T$, and $\epsilon_t \sim \mathcal{N}(0, R)$ and $\delta_t \sim \mathcal{N}(0, Q)$ are Gaussian noise. The corresponding covariances are: $R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix}$ and $Q = \sigma_m^2$.

In the simulation, we set $\Delta_t = 1$, $\sigma_x^2 = \sigma_v^2 = \sigma_\alpha^2 = 1$ and $\sigma_m^2 = 10$. The robot has only an observation $Z_t = 5$ at $t = 5$, otherwise there is no observable data. The estimated robot's position x and speed v, and their error ellipses are shown in Fig. 2.1. The error ellipses are drawn by using the Cholesky factorization, which is introduced in Appendix A. Since there are no measurement data available at the first four time steps, we can see that the covariance of (x, v) becomes larger from $t = 1$ to $t = 4$, and the mean does not change very much. However, when the robot receives the measurement at time $t = 5$, the covariance becomes smaller (blue ellipse), and the mean is close to the true value. Afterwards, the robot observes nothing at time $t = 6$ from the environment, so the covariance becomes larger again, which means the uncertain of the robot's position and speed increases. This experiment illustrates that the prediction step in the Kalman filter always increases the uncertain of the system state. On the other hand, this uncertain will be decreased when the measurement is available.

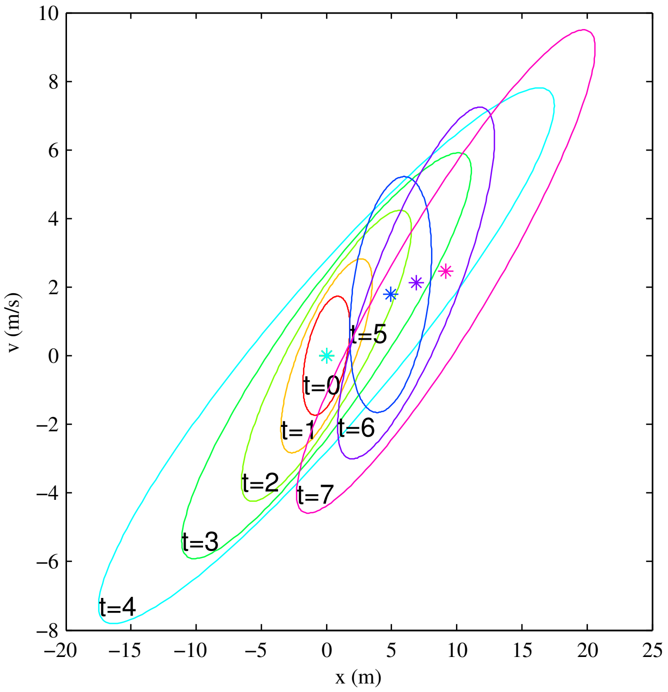


Figure 2.1: The error ellipses of the position x and speed v of the robot from $t = 1$ to $t = 7$. The measurement data is only available at $t = 5$.

2.2.2 Extended Kalman Filter

The LKF assumes that the models of state transition and measurement are linear. However, this is not true in most practical cases. In this section, we introduce a nonlinear version of the Kalman filter, which is called *extended Kalman filter* (EKF). The nonlinear state transition model and measurement model are defined as:

$$Transition\ model: \quad x_t = g(x_{t-1}, u_t) + \epsilon_t, \tag{2.17}$$

$$Measurement\ model: \quad z_t = h(x_t) + \delta_t, \tag{2.18}$$

where g and h are nonlinear functions, and ϵ_t and δ_t are defined in Eq. (2.6) and Eq. (2.7).

The main idea behind the extended Kalman filter is the linearization of nonlinear functions using *first order Taylor expansion*:

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + G_t(x_{t-1} - \mu_{t-1}) \tag{2.19}$$

$$h(x_{t-1}) \approx h(\hat{\mu}_t) + H_t(x_t - \hat{\mu}_t) \tag{2.20}$$

where $G_t = \frac{\partial g(x_{t-1}, u_t)}{\partial x_{t-1}}|_{(x_{t-1}=\mu_{t-1})}$ and $H_t = \frac{\partial h(x_t)}{\partial x_t}|_{(x_t=\hat{\mu}_t)}$ are the first order partial derivatives. The state transition probability $p(x_t|u_t, x_{t-1})$ and measurement probability $p(z_t|x_t)$ can be derived using this approximation:

$$\begin{aligned} p(x_t|u_t, x_{t-1}) &\approx det(2\pi R_t)^{-\frac{1}{2}} exp\{-\frac{1}{2}[x_t - g(\mu_{t-1}, u_t) - G_t(x_{t-1} - \mu_{t-1})]^T \\ &\quad R_t^{-1}[x_t - g(\mu_{t-1}, u_t) - G_t(x_{t-1} - \mu_{t-1})]\} \qquad (2.21) \\ p(z_t|x_t) &\approx det(2\pi Q_t)^{-\frac{1}{2}} exp\{-\frac{1}{2}[z_t - h(\hat{\mu}_t) - H_t(x_t - \hat{\mu}_t)]^T \\ &\quad Q_t^{-1}[z_t - h(\hat{\mu}_t) - H_t(x_t - \hat{\mu}_t)]\}, \qquad (2.22) \end{aligned}$$

where *det* is determinant. The algorithm of the EKF is derived by substituting $p(x_t|u_t, x_{t-1})$ and $p(z_t|x_t)$ into Eq. (2.4) and Eq. (2.5) respectively, as shown in Algorithm 2.

Algorithm 2 Extended Kalman filter

- Initialization: $\mu_0 = E(x_0)$, $P_0 = E\left((x_0 - \mu_0)(x_0 - \mu_0)^T\right)$.
- For $t = 1, \cdots, \infty$:
  1. Prediction:
  $$\hat{\mu}_t = g(\mu_{t-1}, u_t) \tag{2.23}$$
  $$\hat{P}_t = G_t P_{t-1} G_t^T + R_t \tag{2.24}$$
  2. Measurement update:
  $$K_t = \hat{P}_t H_t^T (H_t \hat{P}_t H_t^T + Q_t)^{-1} \tag{2.25}$$
  $$\mu_t = \hat{\mu}_t + K_t(z_t - h(\hat{\mu}_t)) \tag{2.26}$$
  $$P_t = (I - K_t H_t)\hat{P}_t \tag{2.27}$$

2.2.3 Unscented Kalman Filter

The EKF linearizes the nonlinear functions by the first order Taylor expansion, which can cause large estimation errors when the system is highly nonlinear. To overcome this limitation, [Julier et al., 1995] introduced the *unscented transform* to the framework of the Kalman filter, which is called the *unscented Kalman filter* (UKF). In the UKF, the unscented transform is used instead of Taylor series to linearize the nonlinear functions, which can accurate up the second order. Therefore, the

UKF can achieve more accurate result than the first order EKF. In this section, we first present the idea of the unscented transform, then the algorithm of UKF is derived.

The unscented transform propagates a set of *sigma points* through the nonlinear transition function g and measurement function h, then the moments (mean and covariance) of the Gaussian distribution $p(x_t|x_{t-1}, u_t)$ and $p(z_t|x_t)$ can be derived from these sigma points. It is built on the principle that it is easier to approximate a probability distribution than an arbitrary nonlinear function [Van der Merwe, 2004].

For a nonlinear model $y = g(x)$ where x is a L dimensional variable and meets $x \sim \mathcal{N}(\mu, P_x)$, a set of $2L + 1$ weighted *sigma points* are deterministically chosen according to the following rule:

$$\mathcal{X}_i = \begin{cases} \mu, & i = 0 \\ \mu + (\sqrt{(\lambda + L)P_x})_i, & i = 1, \cdots, L \\ \mu - (\sqrt{(\lambda + L)P_x})_i, & i = L+1, \cdots, 2L \end{cases} \tag{2.28}$$

where $\lambda = \alpha^2(L + \kappa) - L$, α and κ are scaling parameters that determine how far the *sigma points* spread from the mean μ [Van der Merwe, 2004; Thrun et al., 2005], and i denotes the i_{th} column of the square root of the prior covariance matrix P_x. An efficient way to calculate the square root of a positive-define matrix is the *Cholesky factorization*.

These sigma points $\mathcal{X}_i$ are then propagated through the nonlinear system function g:

$$\mathcal{Y}_i = g(\mathcal{X}_i). \tag{2.29}$$

The mean μ_y, covariance P_y and cross-covariance P_{xy} of the variable y are extracted from the output sigma points $\mathcal{Y}_i$ as follows:

$$\mu_y = \sum_{i=0}^{2L} w_i^m \mathcal{Y}_i, \tag{2.30}$$

$$P_y = \sum_{i=0}^{2L} w_i^c (\mathcal{Y}_i - \mu_y)(\mathcal{Y}_i - \mu_y)^T, \tag{2.31}$$

$$P_{xy} = \sum_{i=0}^{2L} w_i^c (\mathcal{X}_i - \mu)(\mathcal{Y}_i - \mu_y)^T, \tag{2.32}$$

where the weights w_i^m and w_i^c are defined as follows:

$$\begin{aligned} w_0^m &= \frac{\lambda}{L + \lambda}, \\ w_0^c &= \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta), \\ w_i^m &= w_i^c = \frac{1}{2(L + \lambda)} \quad i = 1, \cdots, 2L, \end{aligned} \tag{2.33}$$

where β, κ and α are constant parameters. As discussed by Van der Merwe [2004], the optimal value β for a Gaussian distribution is 2. In addition, the positive semi-definiteness of the covariance matrix requires $\kappa \geq 0$ and $0 \leq \alpha \geq 1$.

Finally, the UKF is derived using the unscented transform on the state transition function g and measurement function h, as shown in Algorithm 3 where $\gamma = \sqrt{\lambda + L}$ in Eq. 2.34 and Eq. 2.38 for simplifying the algorithm.

In Algorithm 3, we use the unscented transform twice to generate sigma points, since the state transition and measurement update have different additional Gaussian noise. If we use an augmented state vector to include the additional noise information, two operations can be simplified to one. The augmented state vector is defined as

$$x_t^a = \begin{bmatrix} x_t \\ \epsilon_t \\ \delta_t \end{bmatrix}, \tag{2.45}$$

where $x_t \sim \mathcal{N}(\mu_t, P_t)$, $\epsilon_t \sim \mathcal{N}(\mu_t^r, R_t)$ and $\delta_t \sim \mathcal{N}(\mu_t^q, Q_t)$. The augmented mean and covariance of x_t^a are

$$\mu_t^a = \begin{bmatrix} \mu_t \\ \mu_t^r \\ \mu_t^q \end{bmatrix}, \quad P_t^a = \begin{bmatrix} P_t & 0 & 0 \\ 0 & R_t & 0 \\ 0 & 0 & Q_t \end{bmatrix}. \tag{2.46}$$

We substitute μ_t^a and P_t^a into the UKF algorithm to derive the augmented version, as shown in Algorithm 4 where the augmented sigma point $\mathcal{X}_t^a = [\mathcal{X}_t^x, \mathcal{X}_t^r, \mathcal{X}_t^q]^T$.

The augmented UKF and the non-augmented UKF are identical only if $L + \kappa = constant$ [Wu et al., 2005]. The augmented version can capture the odd-order moment information, such that it can give a better estimation result than the non-augmented version for nonlinear problems. However, the augmented version has a larger state dimension than the non-augmented one, so it requires more computation to propagate the sigma points.

2.2.4 Central Difference Kalman Filter

The UKF uses the unscented transform to handle nonlinear functions. Another option to linearize the nonlinear functions is to use the *Stirling's interpolation*. If we use the Stirling's interpolation instead of the unscented transform in the Algorithm 3, a new filter can be derived, which is called the *central difference Kalman filter* (CDKF) [Ito and Xiong, 2000].

Similar to the *unscented transform*, the $2L + 1$ prior sigma points used in the Stirling's interpolation are given by the prior mean μ plus or minus the columns of the scaled square root of the

---

Algorithm 3 Unscented Kalman filter

- Initialization: $\mu_0 = E(x_0)$, $P_0 = E\Big((x_0 - \mu_0)(x_0 - \mu_0)^T\Big)$.
- For $t = 1, \cdots, \infty$:
  1. Generate sigma points for prediction:
  $$\mathcal{X}_{t-1} = [\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{P_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{P_{t-1}}] \tag{2.34}$$
  2. Prediction:
  $$\mathcal{X}_{t|t-1} = g(u_t, \mathcal{X}_{t-1}) \tag{2.35}$$
  $$\hat{\mu}_t = \sum_{i=0}^{2L} w_i^m \mathcal{X}_{i,t|t-1} \tag{2.36}$$
  $$\hat{P}_t = \sum_{i=0}^{2L} w_i^c (\mathcal{X}_{i,t|t-1} - \hat{\mu}_t)(\mathcal{X}_{i,t|t-1} - \hat{\mu}_t)^T + R_t \tag{2.37}$$
  3. Generate sigma points for measurement update:
  $$\hat{\mathcal{X}}_t = \left[\hat{\mu}_t \quad \hat{\mu}_t + \gamma\sqrt{\hat{P}_t} \quad \hat{\mu}_t - \gamma\sqrt{\hat{P}_t}\right] \tag{2.38}$$
  4. Measurement update equations:
  $$\hat{\mathcal{Z}}_t = h(\hat{\mathcal{X}}_t) \tag{2.39}$$
  $$\hat{P}_{z_t} = \sum_{i=0}^{2L} w_i^c (\hat{\mathcal{Z}}_{i,t} - \hat{z}_t)(\hat{\mathcal{Z}}_{i,t} - \hat{z}_t)^T + Q_t \tag{2.40}$$
  $$\hat{P}_{x_t z_t} = \sum_{i=0}^{2L} w_i^c (\hat{\mathcal{X}}_{i,t} - \hat{\mu}_t)(\hat{\mathcal{Z}}_{i,t} - \hat{z}_t)^T \tag{2.41}$$
  $$K_t = \hat{P}_{x_t z_t} \hat{P}_{z_t}^{-1} \tag{2.42}$$
  $$\mu_t = \hat{\mu}_t + K_t (z_t - \hat{z}_t) \tag{2.43}$$
  $$P_t = \hat{P}_t - K_t \hat{P}_{z_t} K_t^T \tag{2.44}$$

Algorithm 4 Augmented unscented Kalman filter

- Initialization: $\mu_0 = E(x_0)$, $P_0 = E\Big((x_0-\mu_0)(x_0-\mu_0)^T\Big)$.
- For $t = 1, \cdots, \infty$:
  1. Generate sigma points for prediction and measurement update:

$$\mu^a_{t-1} = \begin{bmatrix} \mu_{t-1} \\ \mu^r_{t-1} \\ \mu^q_{t-1} \end{bmatrix}, \quad P^a_{t-1} = \begin{bmatrix} P_{t-1} & 0 & 0 \\ 0 & R_{t-1} & 0 \\ 0 & 0 & Q_{t-1} \end{bmatrix} \tag{2.47}$$

$$\mathcal{X}^a_{t-1} = \Big[\mu^a_{t-1} \quad \mu^a_{t-1} + \gamma\sqrt{P^a_{t-1}} \quad \mu^a_{t-1} - \gamma\sqrt{P^a_{t-1}}\Big] \tag{2.48}$$

  2. Prediction:

$$\mathcal{X}^x_{t|t-1} = g(u_t, \mathcal{X}^x_{t-1}, \mathcal{X}^r_{t-1}) \tag{2.49}$$

$$\hat{\mu}_t = \sum_{i=0}^{2L} w^m_i \mathcal{X}^x_{i,t|t-1} \tag{2.50}$$

$$\hat{P}_t = \sum_{i=0}^{2L} w^c_i (\mathcal{X}^x_{i,t|t-1} - \hat{\mu}_t)(\mathcal{X}^x_{i,t|t-1} - \hat{\mu}_t)^T \tag{2.51}$$

  3. Measurement update equations:

$$\hat{\mathcal{Z}}_t = h(\mathcal{X}^x_{t|t-1}, \mathcal{X}^q_{t-1}) \tag{2.52}$$

$$\hat{z}_t = \sum_{i=0}^{2L} w^m_i \hat{\mathcal{Z}}_{i,t} \tag{2.53}$$

$$\hat{P}_{z_t} = \sum_{i=0}^{2L} w^c_i (\hat{\mathcal{Z}}_{i,t} - \hat{z}_t)(\hat{\mathcal{Z}}_{i,t} - \hat{z}_t)^T \tag{2.54}$$

$$\hat{P}_{x_t z_t} = \sum_{i=0}^{2L} w^c_i (\mathcal{X}^x_{i,t|t-1} - \hat{\mu}_t)(\hat{\mathcal{Z}}_{i,t} - \hat{z}_t)^T \tag{2.55}$$

$$K_t = \hat{P}_{x_t z_t} \hat{P}^{-1}_{z_t} \tag{2.56}$$

$$\mu_t = \hat{\mu}_t + K_t(z_t - \hat{z}_t) \tag{2.57}$$

$$P_t = \hat{P}_t - K_t \hat{P}_{z_t} K^T_t \tag{2.58}$$

---

prior covariance matrix P_x:

$$\mathcal{X}_i = \begin{cases} \mu, & i = 0 \\ \mu + (h\sqrt{P_x})_i, & i = 1, \cdots, L \\ \mu - (h\sqrt{P_x})_i, & i = L+1, \cdots, 2L, \end{cases} \tag{2.59}$$

where h is a scaling parameter and L is the dimension of the state vector. The subscript i indicates the i_{th} column of the matrix. A set of the posterior sigma points can be derived by propagating these prior sigma points through the nonlinear system function g: $\mathcal{Y}_i = g(\mathcal{X}_i)$. Furthermore, the estimations of mean $\bar{y}$, covariance P_y and cross-covariance P_{xy} are obtained as follows:

$$\bar{y} = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{Y}_i, \tag{2.60}$$

$$P_{xy} = \sqrt{w_1^{(c1)} P_x} (\mathcal{Y}_{1:L} - \mathcal{Y}_{L+1:2L})^T, \tag{2.61}$$

$$\begin{aligned} P_y = & \sum_{i=1}^{L} w_i^{(c_1)} (\mathcal{Y}_i - \mathcal{Y}_{i+L})(\mathcal{Y}_i - \mathcal{Y}_{i+L})^T \\ & + \sum_{i=1}^{L} w_i^{(c_2)} (\mathcal{Y}_i + \mathcal{Y}_{i+L} - 2\mathcal{Y}_0)(\mathcal{Y}_i + \mathcal{Y}_{i+L} - 2\mathcal{Y}_0)^T. \end{aligned} \tag{2.62}$$

The corresponding weights for the mean and covariance are defined as:

$$\begin{aligned} w_0^{(m)} &= \tfrac{h^2 - L}{h^2} \\ w_i^{(m)} &= \tfrac{1}{2h^2}, \\ w_i^{(c1)} &= \tfrac{1}{4h^2}, \\ w_i^{(c2)} &= \tfrac{h^2 - 1}{4h^4}, \quad i = 1, \cdots, 2L. \end{aligned} \tag{2.63}$$

As proved in [Van der Merwe, 2004], if the random variables obey a Gaussian distribution, the optimal value of h is $\sqrt{3}$. In addition, the Stirling's interpolation only depends on the interval size h, in contrast to three parameters (α, β, κ) required in the unscented transform. This makes the Stirling's method simpler and easier to adjust.

By employing the Stirling interpolation instead of the unscented transform in Algorithm 3, we can further derive the central difference Kalman filter as shown in Algorithm 5. The augmented version also can be derived following the ideas of Algorithm 4, as shown in Van der Merwe [2004].

Algorithm 5 Central difference Kalman filter

- Initialization: $\mu_0 = E(x_0)$, $P_0 = E\Big((x_0 - \mu_0)(x_0 - \mu_0)^T\Big)$.
- For $t = 1, \cdots, \infty$:
    1. Generate sigma points for prediction:
    $$\mathcal{X}_{t-1} = \Big[\mu_{t-1} \quad \mu_{t-1} + h\sqrt{P_{t-1}} \quad \mu_{t-1} - h\sqrt{P_{t-1}}\Big] \tag{2.64}$$
    2. Prediction:
    $$\mathcal{X}_{t|t-1} = g(u_t, \mathcal{X}_{t-1}) \tag{2.65}$$
    $$\hat{\mu}_t = \sum_{i=0}^{2l} w_i^m \mathcal{X}_{i,t|t-1} \tag{2.66}$$
    $$\hat{P}_t = \sum_{i=1}^{l} w_i^{(c_1)}(\mathcal{X}_i - \mathcal{X}_{i+l})(\mathcal{X}_i - \mathcal{X}_{i+l})^t + \sum_{i=1}^{l} w_i^{(c_2)}(\mathcal{X}_i + \mathcal{X}_{i+l} - 2\mathcal{X}_0)(\mathcal{X}_i + \mathcal{X}_{i+l} - 2\mathcal{X}_0)^t + r_t \tag{2.67}$$
    3. Generate sigma points for measurement update:
    $$\hat{\mathcal{X}}_t = \Big[\hat{\mu}_t \quad \hat{\mu}_t + h\sqrt{\hat{P}_t} \quad \hat{\mu}_t - h\sqrt{\hat{P}_t}\Big] \tag{2.68}$$
    4. Measurement update equations:
    $$\hat{\mathcal{Z}}_t = h(\hat{\mathcal{X}}_t) \tag{2.69}$$
    $$\hat{z}_t = \sum_{i=0}^{2L} w_i^m \hat{\mathcal{Z}}_{i,t} \tag{2.70}$$
    $$\hat{P}_{z_t} = \sum_{i=1}^{L} w_i^{(c_1)}(\hat{\mathcal{Z}}_i - \hat{\mathcal{Z}}_{i+L})(\hat{\mathcal{Z}}_i - \hat{\mathcal{Z}}_{i+L})^T + \sum_{i=1}^{L} w_i^{(c_2)}(\hat{\mathcal{Z}}_i + \hat{\mathcal{Z}}_{i+L} - 2\hat{\mathcal{Z}}_0)(\hat{\mathcal{Z}}_i + \hat{\mathcal{Z}}_{i+L} - 2\hat{\mathcal{Z}}_0)^T + Q_t \tag{2.71}$$
    $$\hat{P}_{x_t z_t} = \sqrt{w_1^{(c1)} P_x}(\hat{\mathcal{Z}}_{1:L} - \hat{\mathcal{Z}}_{L+1:2L})^T \tag{2.72}$$
    $$K_t = \hat{P}_{x_t z_t} \hat{P}_{z_t}^{-1} \tag{2.73}$$
    $$\mu_t = \hat{\mu}_t + K_t(z_t - \hat{z}_t) \tag{2.74}$$
    $$P_t = \hat{P}_t - K_t \hat{P}_{z_t} K_t^T \tag{2.75}$$

2.3 Information Filters

In the framework of Kalman filters, the Gaussian distribution is represented by the moments (mean and covariance). Another way to represent the Gaussian distribution is to use the *information vector* and *information matrix*. Here the information matrix is an inverse of the covariance, and the information vector is equal to the product of the information matrix and the mean. If we use information matrix and information vector to replace the covariance and mean in the algorithm of Kalman filter, a new filter can be derived, which is referred to *information filter* or *inverse covariance filter* [Fraser, 1967].

The information vector and information matrix are used in the information filter to represent a Gaussian distribution. This representation is also named *canonical parameterization* [Thrun et al., 2005]. We can derive the canonical parameters from the moments:

$$Information\ matrix: \quad Y = P^{-1}, \tag{2.76}$$

$$Information\ vector: \quad y = Y\mu, \tag{2.77}$$

where Y is the information matrix and y is the information vector. We can see that the *information matrix* and the *information vector* are a complete parameterization of a Gaussian distribution. More over, the moments can also be derived from canonical parameters:

$$P = Y^{-1}, \tag{2.78}$$

$$\mu = Py. \tag{2.79}$$

The difference in the representation of multivariate Gaussian distributions leads to a different update procedure. In the following sections, we will review the state of the art of the information filters, and analyze the difference between the information filter and the Kalman filter.

2.3.1 Linear Information Filter

The *linear information filter* (LIF) assumes that the state transition model and measurement model are linear as defined in Eq. (2.6) and Eq. (2.7). The algorithm of LIF is obtained by simply replacing the moments by the canonical forms in the linear Kalman filter (LKF) using Eq. (2.78) and Eq. (2.79) as shown in Algorithm 6.

By comparing the LIF algorithm in Algorithm 6 and the LKF algorithm in Algorithm 1, we see that the LIF needs more computational cost than the LKF in the predict part, since it requires an inverse of a $L \times L$ matrix, where L is the dimension of the state vector. By contrast, the LKF needs more computational cost than the LIF in the update part, since the calculation of the Kalman gain K requires the matrix inverse.

The measurement update part of the LIF in Algorithm 6 is additive, which is useful for the multiple sensor fusion, since the predicted information vector and information matrix can be updated by simply summing the information contributions ($C^T Q_t^{-1} C$ and $C^T Q_t^{-1} z_t$) from each of the individual sensors [Durrant-Whyte, 2001].

Algorithm 6 Linear information filter

- Initialization: $\mu_0 = E(x_0)$, $P_0 = E\left((x_0 - \mu_0)(x_0 - \mu_0)^T\right)$, $Y_0 = (P_0)^{-1}$, $y_0 = Y_0 \mu_0$.
- For $t = 1, \cdots, \infty$:
    1. Prediction:

$$\hat{Y}_t = (A Y_{t-1}^{-1} A^T + R_t)^{-1} \tag{2.80}$$

$$\hat{y}_t = \hat{Y}_t (A Y_{t-1}^{-1} y_{t-1} + B u_t) \tag{2.81}$$

    2. Measurement update:

$$Y_t = C^T Q_t^{-1} C + \hat{Y}_t \tag{2.82}$$

$$y_t = C^T Q_t^{-1} z_t + \hat{y}_t \tag{2.83}$$

2.3.2 Extended Information Filter

To deal with the nonlinear state transition model and measurement model, we need to linearize the nonlinear functions in the information filter. If we use the Taylor series for linearization, a new filter can be derived, which is called *extended information filter* (EIF) as shown in Algorithm 7. It replaces the moments by the canonical forms in the EKF algorithm [Maybeck, 1979]. The nonlinear models are the same as Eq. (2.17) and Eq. (2.18).

2.3.3 Unscented Information Filter

We have shown that the unscented transform can also be used for linearization instead of Taylor series in Section 2.2.3. Therefore, a new nonlinear information filter can be derived by using unscented transform for linearization, which is the *unscented information filter* (UIF) developed by [Lee, 2008a; Kim et al., 2008] as shown in Algorithm 8.

2.4 Particle Filters

Since both the Kalman filters and information filters have the same assumption on the probability distribution of the state variables, i.e., Gaussian distribution, they can not work well when the system has non-Gaussian noises. To solve this challenging problem, a number of samples can be used to approximate this probability distribution. Then these samples are propagated through the state transition model and measurement model which can be linear or nonlinear. In this way,

Algorithm 7 Extended information filter

```
• Initialization: $\mu_0 = E(x_0)$, $P_0 = E\left((x_0 - \mu_0)(x_0 - \mu_0)^T\right)$, $Y_0 = (P_0)^{-1}$, $y_0 = Y_0\mu_0$.
• For $t = 1, \cdots, \infty$:
    1. Prediction:
        $\mu_{t-1} = Y_{t-1}^{-1} y_{t-1}$   (2.84)
        $\hat{\mu}_t = g(\mu_{t-1}, u_t)$   (2.85)

        $\hat{Y}_t = (G_t Y_{t-1}^{-1} G_t^T + R_t)^{-1}$   (2.86)
        $\hat{y}_t = \hat{Y}_t \hat{\mu}_t$   (2.87)
    2. Measurement update:
        $Y_t = \hat{Y}_t + H_t^T Q_t^{-1} H_t$   (2.88)
        $y_t = \hat{y}_t + H_t^T Q_t^{-1}(z_t - h(\hat{\mu}_t) + H_t\hat{\mu}_t)$   (2.89)
```

it can handle any probability distributions, e.g., Gaussian, Log-normal, uniform and chi-square distributions. A commonly used Bayes filter that utilizes this mechanism is the *particle filter*. In the particle filter, these samples are called particles. Each single particle represents one hypothesis of the state variables, in contrast to the Gaussian filters which have only one state hypothesis. After the particles are propagated through the state transition model, the measurement model can be used to check how well one particle (hypothesis) fits the measurements. The degree of the fitness can be represented by a weight. In other words, if the particle coincides with the measurements well, it will be given a high weight. The probability value of one particle is proportional to its weight value. As a result, the particles together with their weights present the profile of the probability distribution of the state variables. In this section, we will introduce the standard particle filter and one of its extensions, i.e., the partitioned particle filter.

2.4.1 Standard Particle Filter

The particle filter is one of the Bayes filters, so it also composes two processing steps: state prediction and measurement update. However, in the particle filter, they usually have different names: *sampling* (prediction) and *resampling* (measurement update). In the step of sampling, the particle filter draws a number of samples from a known distribution, e.g., Gaussian distribution. Given m particles $X_{t-1} = \{x_{t-1}^1 \cdots x_{t-1}^m\}$ and their weights $W_{t-1} = \{w_{t-1}^1, \cdots, w_{t-1}^m\}$ at time step $t-1$, the new predicted particles can be drawn from the state transition probability:

$$\hat{x}_t^i \sim p(x_t | x_{t-1}^i, u_t), \tag{2.105}$$

Algorithm 8 Unscented information filter

- Initialization: $\mu_0 = E(x_0)$, $P_0 = E\Big((x_0 - \mu_0)(x_0 - \mu_0)^T\Big)$.
- For $t = 1, \cdots, \infty$:
  1. Generate sigma points for prediction:

$$\mu_{t-1}^a = \begin{bmatrix} \mu_{t-1} \\ \mu_{t-1}^r \end{bmatrix}, \quad P_{t-1}^a = \begin{bmatrix} P_{t-1} & 0 \\ 0 & R_{t-1} \end{bmatrix} \tag{2.90}$$

$$\mathcal{X}_{t-1}^a = \Big[\mu_{t-1}^a \quad \mu_{t-1}^a + \gamma\sqrt{P_{t-1}^a} \quad \mu_{t-1}^a - \gamma\sqrt{P_{t-1}^a}\Big] \tag{2.91}$$

  2. Prediction:

$$\mathcal{X}_{t|t-1}^x = g(u_t, \mathcal{X}_{t-1}^x, \mathcal{X}_{t-1}^r) \tag{2.92}$$

$$\hat{\mu}_t = \sum_{i=0}^{2L} w_i^m \mathcal{X}_{i,t|t-1}^x \tag{2.93}$$

$$\hat{P}_t = \sum_{i=0}^{2L} w_i^c (\mathcal{X}_{i,t|t-1}^x - \hat{\mu}_t)(\mathcal{X}_{i,t|t-1}^x - \hat{\mu}_t)^T \tag{2.94}$$

$$\hat{Y}_t = (\hat{P}_t)^{-1} \tag{2.95}$$

$$\hat{y}_t = \hat{Y}_t \hat{\mu}_t \tag{2.96}$$

  3. Generate sigma points for measurement update:

$$\mathcal{X}_t = \Big[\hat{\mu}_t \quad \hat{\mu}_t + \gamma\sqrt{\hat{P}_t} \quad \hat{\mu}_t - \gamma\sqrt{\hat{P}_t}\Big] \tag{2.97}$$

  4. Measurement update equations:

$$\hat{\mathcal{Z}}_t = h(\mathcal{X}_t) \tag{2.98}$$

$$\hat{z}_t = \sum_{i=0}^{2L} w_i^m \hat{\mathcal{Z}}_{i,t} \tag{2.99}$$

$$\hat{P}_{x_t z_t} = \sum_{i=0}^{2L} w_i^c (\mathcal{X}_{i,t} - \hat{\mu}_t)(\hat{\mathcal{Z}}_{i,t} - \hat{z}_t)^T \tag{2.100}$$

$$\phi_t = \hat{Y}_t \hat{P}_{x_k z_k} Q_t^{-1} \Big[z_k - \hat{z}_t + (\hat{P}_{x_k z_k})^T \hat{y}_t\Big] \tag{2.101}$$

$$\Phi_t = \hat{Y}_t \hat{P}_{x_k z_k} Q_t^{-1} (P_{x_k z_k})^T (\hat{Y}_t)^T \tag{2.102}$$

$$y_t = \hat{y}_t + \phi_t \tag{2.103}$$

$$Y_t = \hat{Y}_t + \Phi_t \tag{2.104}$$

---

where $i = 1, \cdots, m$ and the state transition can be a linear or nonlinear process.

The second step is to resample the particle set $\hat{X}_t = \{\hat{x}_t^1, \cdots, \hat{x}_t^m\}$ based on the new weights defined by:

$$\hat{w}_t^i = w_{t-1}^i p(z_t|\hat{x}_t^i), \tag{2.106}$$

where measurement probability $p(z_t|\hat{x}_t^i)$ is determined by the measurement model. Afterwards, the weights have to be normalized to ensure that they sum up to one:

$$w_t^i = \frac{\hat{w}_t^i}{\sum_{i=1}^m \hat{w}_t^i}. \tag{2.107}$$

From the Eq. (2.106), the weight is proportional to the measurement likelihood, which means a high weight indicates that the hypothesis given by this particle is likely to be true (close to the true state). The mechanism of the resampling step is to keep these particles which have high weights, and in the mean time, remove these particles which have lower weights. This is known as *survival of fittest*. On the other hand, Thrun et al. [2005] noted that the sampling variance is amplified through repetitive resampling. One solution is resampling the particle set only when the effective number of particles N_{eff} is less than a threshold N_{thr}:

$$N_{eff} = \frac{1}{\sum_{i=1}^m (w_t^i)^2}. \tag{2.108}$$

Another solution is to use the low variance sampler, e.g., the Stratified resampling introduced by Kitagawa [1996] as shown in Algorithm 9. The function $rand(1, m)$ samples m random numbers from a uniform distribution on the interval $[0, 1]$. After the resampling, the final estimated result can be derived by a weighted sum of all particles. Finally, we can summarize the algorithm of particle filter in Algorithm 10.

To better understand the idea of the particle filter, we again use the example of the robot localization problem, which was introduced in Section 2.2.1. The demonstration of the particle filter for solving this problem is as follows:

- Sampling from the state transition probability. The predicted particle set $\hat{X}_t = (\hat{x}_t^1, \cdots, \hat{x}_t^m)$ is drawn from the state transition probability $p(x_t|x_{t-1}) = \mathcal{N}(Ax_{t-1}, R)$.
- Calculate the weights. $w_t^i = w_{t-1}^i p(z_t^i|x_t^i) = \mathcal{N}(z_t^i, Q) = \frac{1}{\sqrt{2\pi\sigma_m^2}} exp(-\frac{(z_t^i - \hat{z}_t^i)^2}{2\sigma_m^2})$, where $\hat{z}_t^i = C\hat{x}_t^i$ is the predicted measurement and z_t^i is the measurement from the sensor. Finally the normalized weight set is $W_t = (w_t^1, \cdots, w_t^m)$.
- Importance resampling. A new particle set X_t is derived using the Stratified resampling method, then the weights are reset to be equal again: $W_t = (1/m, \cdots, 1/m)$.

In order to show the effect of the resampling, we plot the distribution of the particles before

Algorithm 9 Stratified resampling

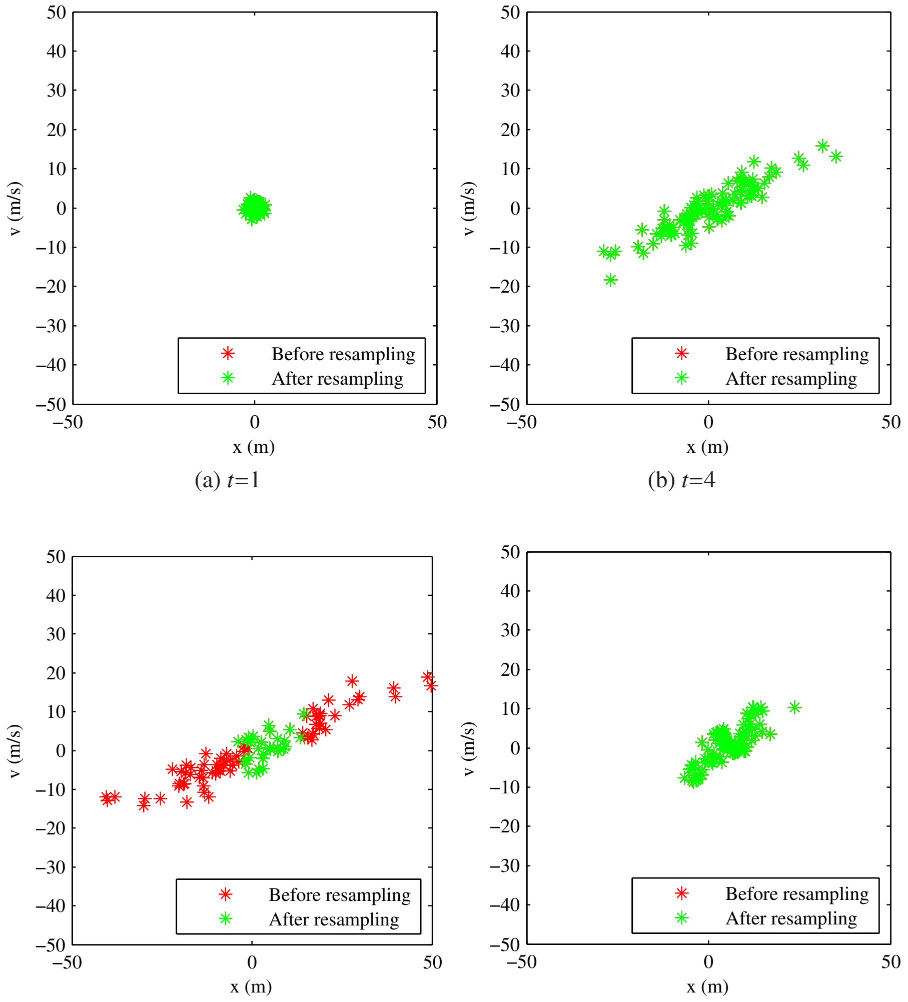
```
Given a particle set $X_t = (x_t^1, \cdots, x_t^m)$ and its weight $W_t = (w_t^1, \cdots, w_t^m)$ at time $t$
$\overline{X_t} = \phi$
$r = rand(1, m)/m$
$c = w_t^1$
$k = 1$
for $i = 1$ to $m$ do
    $U = r(i) + (i - 1)/m$
    while $U > c$ do
        $k = k + 1$
        $c = c + w_t^k$
    end while
    add $x_t^i$ to $\overline{X_t}$
end for
$X_t = \overline{X_t}$
```

Algorithm 10 Particle filter

```
For particle set $X_{t-1} = (x_{t-1}^1, \cdots, x_{t-1}^m)$ and corresponding weights $W_{t-1} = \{w_{t-1}^1, \cdots, w_{t-1}^m\}$ at time $t - 1$:
for $i = 1$ to $m$ do
    sample $\hat{x}_t^i \sim p(x_t|u_t, x_{t-1}^i)$
    $\hat{w}_t^i = w_{t-1}^i p(z_t|x_t^i)$
end for
normalize the weights: $w_t^i = \frac{\hat{w}_t^i}{\sum_{i=1}^m \hat{w}_t^i}$
effective number: $N_{eff} = \frac{1}{\sum_{i=1}^m (w_t^i)^2}$
if $N_{eff} < N_{thr}$ then
    resample particles set $X_t$ with probability $\propto W_t = (w_t^1, \cdots, w_t^m)$
    reset the weights to be equal: $w_t^i = \frac{1}{m}$
end if
return $X_t$ and $W_t$
```

(red stars) and after (green stars) resampling in the $2D$ coordinates of the position x and speed v during first six time steps in Fig. 2.2. The corresponding weights are shown in Fig. 2.3. In the first four time steps, i.e., $t < 5$, the robot receives no measurement data, so the weights of particles are all equal and the resampling does nothing. Therefore, the red stars and green stars are overlapped in Fig. 2.2a and Fig. 2.2b. However, as we mentioned in Section 2.2.1, the state transition step will increase the covariance, so the particles spread to a larger area as time goes on. Until $t = 5$, the robot senses its position is $z = 5$ from the sensor, so the particles close to $x = 5$ have higher weights than others as shown in Fig. 2.3. As a result, after resampling only these particles close to $x = 5$ are kept (green stars in Fig. 2.2c), whereas others are removed from the set (red stars in Fig. 2.2c). We can also see that the distribution of particles is shrunk at $t = 5$, which means the covariance of these particles is smaller than before. However, at time step $t = 6$, the robot senses nothing again, so the weights are equal again and the distribution of particles is expanded to a larger area again.

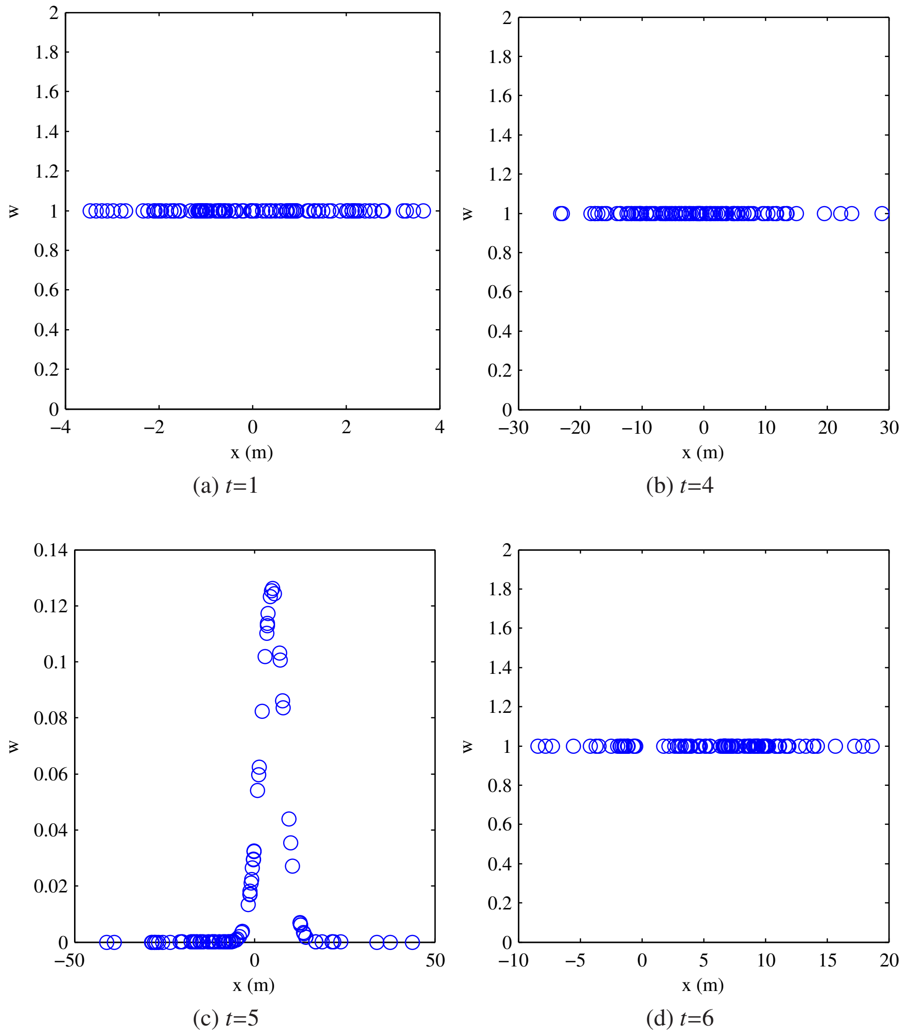
From this simple experiment, we can see how particle filter works, and how the particles together with their weights approximate the probability distribution. In addition, to compare with Kalman filters and information filters, the particle filter can deal with both linear and nonlinear models, and does not require the calculation of Jacbian matrix. More importantly, the particle filter can handle different probability distribution. Therefore, the particle filter is a very easy and powerful tool for solving state estimation problems.



(a) *t*=1 (b) *t*=4

(c) *t*=5 (d) *t*=6

Figure 2.2: The distribution of particles before (red) and after (green) resampling at time=1, 4, 5 and 6.



(a) t=1 (b) t=4 (c) t=5 (d) t=6

Figure 2.3: The weights for particle set at time=1, 4, 5 and 6

2.4.2 Partitioned Particle Filter

As we mentioned in Section 2.4, the particle filter uses a number of particles (samples) to approximate the true probability distribution. The accuracy of this approximation depends on the number of the particles, i.e., more particles usually give more accurate results, but more particles also means that more computational cost are required. In general, a high dimensional probability distribution needs more particles to approximate than the low dimensional one. To reduce the prerequisite number of particles for a high dimensional distribution, MacCormick and Blake [1999] presented an extension of the standard particle filter, which is called *partitioned particle filter* (PPF). The PPF first divides the whole state space into several subspaces, then for each subspace except the final one, an individual sampling and *weighted resampling* process is employed. The main difference between weighted resampling and normal resampling is that the former does not reset the weights to be equal after resampling. Instead, the weights are set to be the inverses of previous normalized weights. In this way, the weighted resampling improves the concentration of samples at specific subspace within the distribution of the whole state space, without introducing a bias to these regions[Louw, 2004]. On the other hand, the standard resampling, e.g., Stratified resampling, is only used for the last subspace. The hierarchical processing of PPF can reduce the required number of particles, and has been proved very robust for the multi-objects tracking and multi-cues tracking [MacCormick and Blake, 1999; Louw, 2004; Duffner et al., 2009].

To better illustrate the process of PPF, we here give a simple example. Concerning the problem of tracking three individual objects, the whole state space x includes three subspaces $x = \{x^1, x^2, x^3\}$ which correspond to three objects. First, given the prior distribution at time $t-1$ defined by the m particles $x = \{x^i_{t-1}, w^i_{t-1}\}, i = 1, \cdots, m$, we use the state transition model of the first object to predict its new state x^1, which is denoted as $*p(x^1|x)$. Second, we apply the weighted resampling on subspace x^1 using the likelihood function g_1 of the first object, which is denoted as $\otimes g_1$. Because we only resample on the subspace x^1, this will introduce a bias to the whole distribution if the standard resampling mechanism is used. Instead, we employ the weighted resampling to set $w'^i_t = w^i_t/\rho_i$ for the predicted particle x^i_t at time t, where $\rho_i = g_1(x^i_t)/\sum_{i=1}^{m} g_1(x^i_t)$. Third, we uses the same procedure as we did for the first object to estimate the second object x^2. Fourth, we uses the state transition model of the third object to predict its new state, which is denoted as $*p(x^3|x^2)$. Then the predicted particles are resampled using the standard resampling method, which is denoted as $\odot g_3$. Finally, the posterior distribution is derived after the whole hierarchical processing. These steps are summarized as follows:

$$\begin{aligned} &\boxed{prior} \rightarrow \boxed{*p(x^1|x)} \rightarrow \boxed{\otimes g_1} \\ &\rightarrow \boxed{*p(x^2|x^1)} \rightarrow \boxed{\otimes g_2} \\ &\rightarrow \boxed{*p(x^3|x^2)} \rightarrow \boxed{\odot g_3} \rightarrow \boxed{posterior} \end{aligned}$$

In this section, we have introduced the basic idea of the PPF. More details are referred to [MacCormick and Blake, 1999]. From our discussions, we can see that the PPF is suitable to handle the high dimensional state. In Chapter 3, we will demonstrate the PPF on a real application for lane tracking.

2.5 Conclusion

In this chapter, we have shown the general form of Bayes filters, which is a theoretical basis of Bayesian filtering. In addition, we have discussed three commonly used Bayes filters, i.e., Kalman filter, information filter and particle filter. More over, the extensions of these three filters have also been introduced to handle difficult cases, e.g., nonlinear models and high dimensional states. Although this chapter is mostly about the state of the art works in the field of Bayesian filtering, we want to illustrate the pros and cons of each filter compared to others. In this way, we can choose the suitable Bayes filter for a specific task.

Chapter 3

Multiple Visual Cues for Lane Tracking

This chapter presents a novel lane tracking framework using improved measurements by combining multiple visual cues. The related works in this field are reviewed in Section 3.2. We introduce the local image descriptors as visual cues in Section 3.3. In Section 3.4, the multi-kernel density estimation for line, parabolic and linear-parabolic shapes are presented. Section 3.5 describes the partitioned particle filter (PPF) for linear-parabolic parameters estimation. Results and analyses are given in Section 3.6, which introduces experimental results on different road situations.

3.1 Introduction

Autonomous navigation on various roads requires the knowledge of lane information, which is also an open problem for Driver Assistance Systems. To extract lane boundary information, vision is a natural and powerful tool. However, high curvature, occlusions, varying illumination and unmarked or partly marked lanes in the image are still challenging situations for this task [McCall and Trivedi, 2006; Kim, 2008].

This chapter presents a new lane tracking framework. A linear-parabolic model is chosen to depict the lane shapes in top-view images. The visible lanes are separated into two parts: the near part and the far part, which depend on the distance to the vehicle. There are few reasons to design the shape of the lane markers in this way: first, the lane markers in real situations have curved shape, but the near parts in most cases are straight. Second, because of the projection geometry of the camera, the lane markers in the near part have higher resolution than in the far part, so the estimation of the near part in general is more accurate than that of the far part. Third, for the drivers, the near part of lane markers is very important information for immediate behavioral response.

We introduce a new lane shape estimation mechanism for the linear-parabolic model. The previously used method for the linear-parabolic lane shape estimation is the least-squares or Kalman filter [Jung and Kelber, 2004; Lim et al., 2009]. However, the least-squares method is sensitive to image noise, and the Kalman filter only maintains one hypothesis of the lane parameter. In contrast, the Particle filter (PF) has multiple hypotheses of lane parameters, and has the ability to recover the true state from the tracking failure. Our chapter applies the novel Partitioned Particle filter (PPF) to the linear-parabolic model estimation. The PPF employs the partitioned sampling to estimate the lane parameters in a hierarchical processing scheme. In our case, the PPF first samples on the parameters of the linear part and then on the parameters of the parabolic part. The PPF has superiority over the standard PF for the linear-parabolic shape estimation: a) the PPF requires a smaller number of particles than the PF to model a distribution of the high dimensional system state, b) the importance of the linear part estimation is emphasized by the PPF.

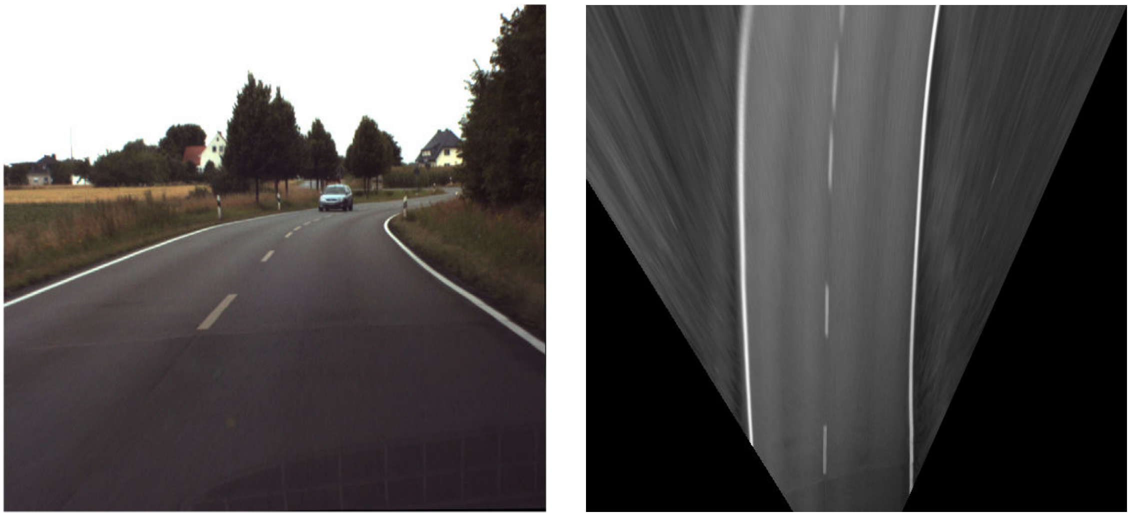
The other main contribution of this chapter is the multiple kernel density based measurement function for the PPF. For each particle in the PPF, we want to use a measurement function to describe how much this particle (one hypothesis of the lane parameter) fits the image. In contrast to other works which only employ the color and edge information (thresholded gradient magnitude) as the measurements, we consider the local gradient orientation as an important information for estimation. The statistical relation between the global lane model and the local image descriptors (color, edge and gradient orientation) is modeled using the multiple kernel density. Furthermore, we show how to derive the measurement function in the PPF from this statistical relation.

3.2 Related Work

3.2.1 Lane Model

Numerous approaches for vision based lane detection and following have been developed in the past [McCall and Trivedi, 2006]. Different lane models have been studied, e.g., straight line, parabolic, circular and spline based models. Normally, simple models are more robust against image artifacts, but can not give an exact fit. Complex models, such as parabolic, circular and spline models, are flexible, but are more sensitive to the image noise [Jung and Kelber, 2004; Danescu and Nedevschi, 2009]. Jung and Kelber [2004] introduced a linear-parabolic model for lane following, which splits the lanes into two parts: the linear part and the parabolic part. This model is a trade-off between accuracy of the fit and robustness with respect to image artifacts [Jung and Kelber, 2004].

In contrast to Jung and Kelber's work which employed the linear-parabolic model in the original image, we use this model on a top-view image which is also known as Inverse Perspective Mapping (IPM) image. IPM is an image transformation that removes the perspective effect, based on the flat ground hypothesis with known extrinsic and intrinsic parameters of the camera [Bertozzi and Broggi, 1998]. It remaps pixels from the original image to a top view image that has a different coordinate system. This remapping procedure can be done by a fast lookup table



(a) (b)

Figure 3.1: (a) An image from the data set, where the lane markers are not parallel. (b) the transformed image after applying IPM shows a top-view of the scene and the lane markers appear nearly parallel.

with a distortion compensation [Bergener and Bruckho, 1999]. As the resolutions for near and far objects are different in the original image, an interpolation process is needed in the IPM algorithm. A resulting example image is shown in Fig. 3.1b, which is a top-view image of Fig. 3.1a with a cubic interpolation. The lanes in the IPM images look parallel and the width between lanes becomes nearly constant.

3.2.2 Estimation Method

After the lane model is defined, the parameters of the model can be estimated from image sequence. Jung and Kelber [2004] employed the Hough transform for initial detection, and updated the parameters of the linear-parabolic model by minimizing a weighted square error. However, a tracking mask with fixed width is needed to find possible lane edges in every iteration, and the least squares method is sensitive to edge noise. Lim et al. [2009] used a Kalman filter to update the linear-parabolic parameters. The Kalman filter continuously estimates the model parameters. The nearby edges of the estimated boundaries are found as measurements for the Kalman filter. This method increases the robustness of the algorithm as compared to the previous least squares method, but the Kalman filter only maintains one hypothesis, so that it is difficult to recover the true state once a tracking failure occurs [Zhou et al., 2006].

This chapter applies the Partitioned Particle filter for estimating the linear-parabolic shape. The Partitioned Particle filter was introduced by MacCormick and Blake [2000], which was designed for multiple object tracking. The PPF inherits the advantages of the PF, such as the multiple hypotheses of the system state and the ability to recover from tracking failure [Danescu and Nedevschi, 2009]. In addition, the PPF also has its own special advantage compared to the standard PF, which is that the PPF uses a smaller number of particles to estimate a distribution of the high dimensional state [Southall and Taylor, 2001]. In addition, the other reason for us to employ PPF instead of the PF is that the parabolic part has more noises than the linear part. Because of the perspective geometry of the camera, the parabolic part has lower resolution in the image than the linear part. This lead to more noises in the parabolic part, which makes its estimation more difficult. The PPF handles this problem by estimating each part in a hierarchical way, first the linear part and second the parabolic part. In this way, the accuracy of the linear part is enhanced, and the noise in the parabolic part does not affect the linear part estimation.

3.2.3 Measurement Model

The estimator of the lane models requires a measurement model to update the lane parameters. In previous works, the color and edge are used as main measurement features for the linear-parabolic shape estimation [Jung and Kelber, 2004; Lim et al., 2009], but the gradient orientation information is omitted. Dahyot [2009] presented the statistical Hough transform (SHT), where the statistical distribution of the lane parameters is modeled by the multiple kernel density estimation. One advantage of the SHT is that the local gradient orientation is considered for estimation.

However, the SHT only handles straight lines. This chapter extends Dahyot's work to the parabolic case, and uses the statistical distribution function of the lane parameters as the measurement function in the PPF.

3.3 Local Image Descriptors

For every pixel in an image, basic descriptors can be defined, e.g., color C, position (x, y) and gradient (I_x, I_y). The advanced descriptors can be derived from those basic descriptors, e.g., the magnitude, alignment, and direction of the gradient: ΔI, ρ and θ. In an urban environment, lanes are not always of the same color, such that the color information C can not as robustly be used as on a highway, whereas position and gradient information are useful for this case.

Our algorithm uses local image descriptors (x, y, θ, C) as observation, and models them by kernel density, e.g., Gaussian kernel, which includes color, position and gradient information [Dahyot, 2009]. The observation space of one image is $Q_{cxy\theta} = \{c_i, x_i, y_i, \theta_i, i = 1, \cdots, n\}$, where n denotes the total number of pixels used for estimating the lane parameters. In addition, the lane parameters are independent of the color information, so we first study the statistical relation between the global model and the local image descriptors on the subspace $Q_{xy\theta} = \{x_i, y_i, \theta_i, i = 1, \cdots, n\}$, then the color kernel will be added in the final measurement model.

3.4 Lane Parameter Estimation by Kernel Density

In this section, we first introduce the line parameters estimation using the multi-kernel density [Dahyot, 2009], then we show the new estimation method for the parabolic estimation, and finally we extend the work to the linear-parabolic parameter estimation.

3.4.1 Line Model

For a straight line model we have

$$\rho = xcos\theta + ysin\theta, \tag{3.1}$$

where x and y are image coordinates, ρ and θ are line parameters, which need to be estimated.

The probability distribution $p(\rho, \theta, x, y|Q_{xy\theta})$ can be written according to the Bayes rule as

$$p(\rho, \theta, x, y|Q_{xy\theta}) = p(\rho|x, y, \theta, Q_{xy\theta}) \cdot p(x, y, \theta|Q_{xy\theta}). \tag{3.2}$$

In Eq. (3.2), the first probability $p(\rho|x, y, \theta, Q_{xy\theta})$ is determined by Eq. (3.1), and the second probability $p(x, y, \theta|Q_{xy\theta})$ can be modeled by the multi-kernel density function. Thus Eq. (3.2) becomes

$$p(\rho, \theta, x, y|Q_{xy\theta}) = \delta(\rho - xcos\theta - ysin\theta)\frac{1}{n}\sum_i K_x K_y K_\theta \tag{3.3}$$

where δ is the Dirac delta function, $K_x = \mathcal{N}(x_i, \sigma_{x_i}^2)$, $K_y = \mathcal{N}(y_i, \sigma_{y_i}^2)$ and $K_\theta = \mathcal{N}(\theta_i, \sigma_{\theta_i}^2)$ are Gaussian kernels. $\sigma_{x_i}^2$, $\sigma_{y_i}^2$ and $\sigma_{\theta_i}^2$ are variances of x_i, y_i, and θ_i, respectively. In the experiment, we set $\sigma_{x_i}^2 = 1$, $\sigma_{y_i}^2 = 1$ and $\sigma_{\theta_i}^2 = \frac{1}{\Delta I}$.

The distribution $p(\rho, \theta|Q_{xy\theta})$ can be obtained by integrating Eq. (3.3) over (x, y):

$$p(\rho, \theta|Q_{xy\theta}) = \frac{1}{n}\sum_i K_\theta \cdot G_{li}(\rho, \theta) \tag{3.4}$$

where

$$\begin{aligned} G_{li}(\rho, \theta) \quad = \quad & \frac{1}{\sqrt{2\pi(\sigma_{x_i}^2 cos^2\theta + \sigma_{y_i}^2 sin^2\theta)}} \cdot \\ & exp\left(\frac{-(\rho - x_i cos\theta - y_i sin\theta)^2}{2(\sigma_{x_i}^2 cos^2\theta + \sigma_{y_i}^2 sin^2\theta))}\right) \end{aligned} \tag{3.5}$$

A detailed description of Eq. (3.5) can be found in [Dahyot, 2009]. Given observation space $Q_{xy\theta}$ of an image, the statistical distribution of the lane parameters can be calculated using Eq. (3.4). One example result is shown in Fig. 3.2.

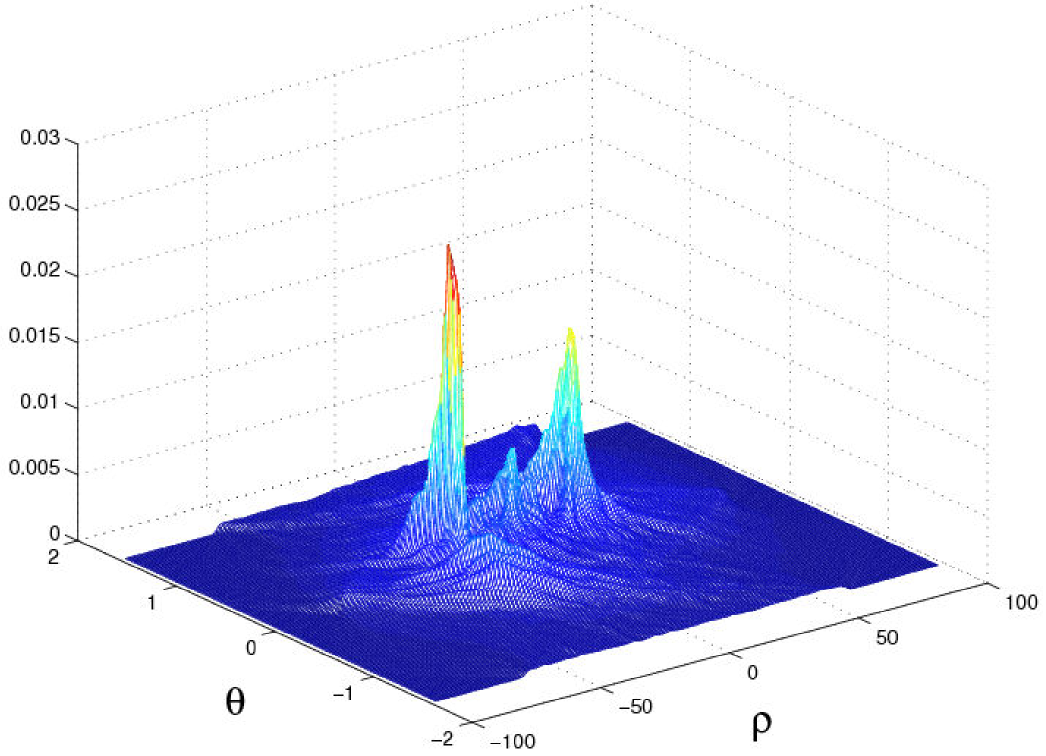


Figure 3.2: Statistical distribution $p(\rho, \theta|Q_{xy\theta})$ given observation space $Q_{xy\theta}$ of Fig. 3.1b

3.4.2 Parabolic Model

Similar to the line model estimation, the parabolic lane model also can be estimated using the kernel densities. However, the parameter θ is not constant in the parabolic case. To model K_θ correctly, the gradient information is introduced to our parameter estimation.

For a parabolic lane model we have:

$$x = c + dy + ey^2 \tag{3.6}$$

where c, d and e are the parabolic lane parameters. The orientation of gradient θ is derived by the first order derivation of Eq. (3.6):

$$\theta = atan(-2ey - d). \tag{3.7}$$

The probability distribution $p(c, d, e, x, y, \theta|Q_{xy\theta})$ can be written as:

$$p(c, d, e, x, y, \theta|Q_{xy\theta}) = p(c, d, e|x, y, \theta, Q_{xy\theta}) \cdot p(x, y, \theta|Q_{xy\theta}) \tag{3.8}$$

where $p(c, d, e|x, y, \theta, Q_{xy\theta})$ is determined by Eq. (3.6) and Eq. (3.7), and the second probability term $p(x, y, \theta|Q_{xy\theta})$ can be modeled by multiple Gaussian kernel density:

$$p(c, d, e, x, y, \theta|Q_{xy\theta}) = \delta_1 \cdot \delta_2 \cdot \frac{1}{n} \sum_i K_x K_y K_\theta \tag{3.9}$$

where $\delta_1 = \delta(c + dy + ey^2 - x)$ and $\delta_2 = \delta(\theta - atan(-2ey - d)$ are Dirac functions, K_x, K_y and K_θ are Gaussian kernels which are same as in Eq. (3.3). The distribution $p(c, d, e|Q_{xy\theta})$ can be obtained by integrating Eq. (3.9) over (x, y, θ):

$$p(c, d, e|Q_{xy\theta}) = \frac{1}{n} \sum_i G_{pi}(c, d, e) \tag{3.10}$$

where

$$G_{pi}(c, d, e) = \iiint_{-\infty}^{\infty} \delta_1 \delta_2 K_x K_y K_\theta \; dxdyd\theta \tag{3.11}$$

Eq. (3.11) also is known as the Radon transform. Because x and θ are represented by y using δ_1 and δ_2 functions, the three-fold integration over (x, y, θ) in Eq. (3.11) is simplified to a single integration over y:

$$G_{pi}(c, d, e) = \int_{-\infty}^{\infty} K'_x K_y K'_\theta \; dy \tag{3.12}$$

where

$$K'_x = \frac{1}{\sqrt{2\pi\sigma^2_{x_i}}} exp\left(-\frac{(c + dy + ey^2 - x_i)^2}{2\sigma^2_{x_i}}\right) \tag{3.13}$$

$$K'_\theta = \frac{1}{\sqrt{2\pi\sigma^2_{\theta_i}}} exp\left(-\frac{(atan(-2ey - d) - \theta_i)^2}{2\sigma^2_{\theta_i}}\right) \tag{3.14}$$

The analytic expression of the Radon transform Eq. (3.12) can not be obtained. To solve it, the Gauss-Hermite numerical method is used to calculate G_{pi} [Liu et al., 2009].

3.4.3 Linear-Parabolic Model

The linear-parabolic model includes two parts: the linear part and the parabolic part. The linear part is used to model the lanes in the near vision field, whereas the parabolic part is used for the lanes in the far vision field [Jung and Kelber, 2004; Lim et al., 2009].

$$x = f(y) = \begin{cases} a + by & if \; y > y_m \\ c + dy + ey^2 & if \; y \le y_m \end{cases} \tag{3.15}$$

where y_m is the border between the near and far vision fields, and the observation space $Q_{xy\theta}$ is divided into two subspaces: Q_{near} and Q_{far}. In next sections, we consider the separate line y_m is

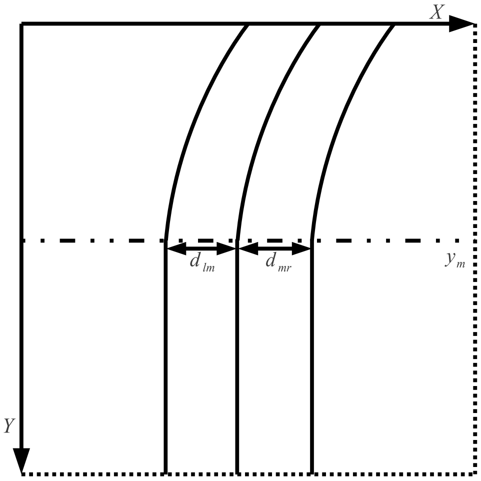


Figure 3.3: The coordinate system of the image and linear-parabolic multiple lane model. In this case the number of lanes are three. y_m is the separate line between the linear part (near vision field) and the parabolic part (far vision field). d_{lm} and d_{mr} are distances between the multiple lanes.

constant and to be the half of the image height as shown in Fig. 3.3.

We impose continuity and differentiability conditions on the function Eq. (3.15) at point y_m, such that $f(y_m^-) = f(y_m^+)$ and $f'(y_m^-) = f'(y_m^+)$. Combining with the Eq. (3.1), we can further obtain:

$$\begin{cases} a = \frac{\rho}{cos(\theta)} \\ b = -tan(\theta) \\ c = \frac{\rho}{cos(\theta)} + \frac{y_m}{2}(-tan(\theta) - d) \\ e = \frac{1}{2y_m}(-tan(\theta) - d) \end{cases} \tag{3.16}$$

According to Eq. (3.16), the linear-parabolic model is totally determined by three parameters: (ρ, θ, d) given y_m. The coordinate system of the linear-parabolic model can be seen in Fig. 3.3.

To estimate the linear-parabolic parameters (ρ, θ, d), we first estimate the linear part (ρ, θ) by Eq. (3.4) using observation Q_{near}, and then we estimate the parabolic part d by Eq. (3.10) using observation Q_{far}. Because c and e are functions of (ρ, θ, d), we find:

$$p(d|Q_{far}) = \frac{1}{n_{far}} \sum_i G_{pi}(c(\rho, \theta, d), d, e(\rho, \theta, d)) \tag{3.17}$$

This hierarchical process can be implemented by the PPF as shown in Section 3.5.

3.5 Partitioned Particle Filter for Lane Detection and Tracking

In [Dahyot, 2009; Liu et al., 2009], the authors estimate line and parabolic parameters by combining the Hough transform with a multi-kernel probability model, which is called SHT. However, the SHT is computational expensive for curved lane detection.

Here we use the PPF instead of the Hough transform to estimate the lane parameters. The multiple kernel density based probability function for the linear-parabolic model is used as the measurement model of the PPF. The PPF was designed for solving high dimensional state problems [MacCormick and Blake, 2000]. The high dimensional state space in the PPF is partitioned into multiple subspaces. Each subspace is estimated hierarchically. This hierarchical process increases efficiency and robustness of the condensation algorithm [Southall and Taylor, 2001].

Our whole algorithm is shown in the Fig. 3.4. In the image preprocessing part, the perspective effect is removed by the IPM algorithm as shown in Appendix B, then the local image descriptors are calculated, e.g., gradient magnitude and orientation. Afterwards, the initialization samples drawn from the default lane model are introduced to the particle array. As we mentioned in the Section 3.2, those initialization samples can be used to recover the state from tracking failure. Finally the linear and parabolic part are estimated hierarchically using the PPF with the observation model based on the multi-kernel density.

3.5.1 Multiple Lane Model

Here we use a simple multiple lane model as shown in Fig. 3.3. The additional conditions between the parallel lanes can be used to check the quality of the particles. For instance, the distances between each pair of nearby lanes are larger than 12 pixels, e.g., $d_{lm} > 12$ and $d_{mr} > 12$, and the difference between d_{lm} and d_{mr} is smaller than 5 pixels.

3.5.2 State Definition

In section 3.4.3, we introduce the probability estimation of linear-parabolic lane parameters from local image descriptors. The linear-parabolic lane model is defined by three parameters (ρ, θ, d). For the situation that the road has multiple lanes, the whole state probability distribution at time t is given by a set of N particles $X_t = \{x_t^j, j = 1, \cdots, N\}$, where $x_t^j = \{\rho_k^j, \theta_k^j, d_k^j, k = 1, \cdots, n_l\}$ is a single particle state, and n_l is the number of lanes.

The dimension of the state x_t^j is $3 \times n_l$ which depends on the number of lanes n_l. In case of three lanes $n_l = 3$, the state x_t^j has nine parameters. That means we need a large number of particles to correctly model this high dimensional state distribution, which leads to the high computational cost.

The solution to decrease the number of particles is to use the PPF, which separates the state x_t^j into two subgroups: the linear part $S_l = \{\rho_k^j, \theta_k^j, k = 1, \cdots, n_l\}$ and the parabolic part $S_p = \{d_k^j, k =$

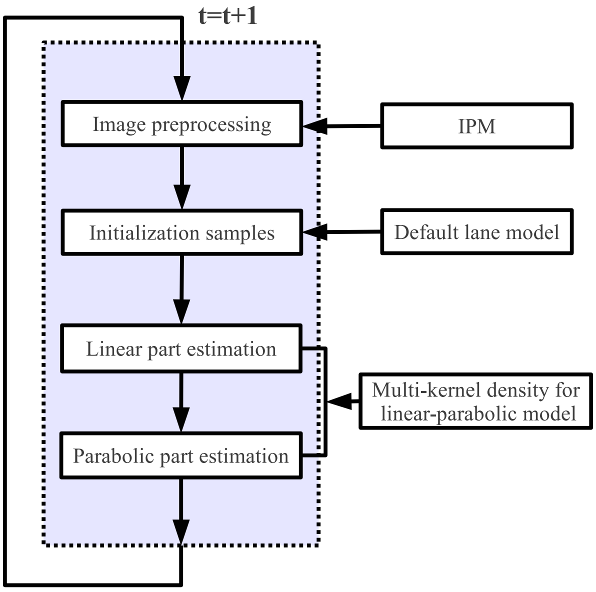


Figure 3.4: Flow chart of our lane detection and tracking algorithm.

$1, \cdots, n_l\}$. The subgroups are estimated by the partitioned sampling in a hierarchical way. Because each subgroup requires less number of particles, the burden of the computational cost is eased off.

3.5.3 Image Preprocessing

To remove the perspective effect of the image, the IPM algorithm is implemented on the consecutive frames, then the local gradient magnitude and orientation are calculated from the top-view image. To save computational cost, we threshold the gradient magnitude image to get an image mask. Only the pixels that have higher gradient values are considered to be measurement.

3.5.4 Initialization Samples

When lanes suddenly disappear, e.g., occluded by a car, a robust algorithm has to maintain the best hypothesis or find the lanes again when possible. The solution is to introduce a constant percentage of initialization samples into the state distribution at every iteration, The initialization samples are drawn from the distribution of the default lane models. In our algorithm, we use $N' = 50$ initialization particles. This mechanism can recover the lane state from failures [Zhou et al., 2006]. Here we use the straight lane model as the default lane model, such that the value of d in random samples are defined as $d = b = -tan(\theta)$.

3.5.5 Linear Part Estimation

The linear part estimation includes two steps: the first step is state prediction using a random-walk probability model. The second step is resampling using a specific observation model, i.e., multi-kernel density model. Finally, only the particles which have high weight values are kept. This resampling process follows the idea of "survival of the fittest" [Thrun et al., 2005].

Linear Part Prediction

Assuming the change of the lane boundary between two consecutive frames is small, a normal distribution can be used to model the state transition of the j_{th} particle as

$$p(\hat{x}_t^{lj} | x_{t-1}^j) = \mathcal{N}(A^l x_{t-1}^j, \Sigma^l), \tag{3.18}$$

where $\mathcal{N}$ is the normal distribution. The matrix A^l is an identity matrix as we assume the lane boundaries have smooth changes, and Σ^l is the covariance which handles the difference of lane boundaries between two consecutive frames. Because we only predict the linear part, Σ^l is defined as

$$\Sigma^l = diag\{V_k^l, k = 1, \cdots, n_l\}, \tag{3.19}$$

where $diag$ is the diagonal matrix function, and $V_k^l = \{\sigma_{\rho_k}^2, \sigma_{\theta_k}^2, 0\}$. $\sigma_{\rho_k}^2$ and $\sigma_{\theta_k}^2$ are the covariances of the k_{th} lane parameters ρ_k and θ_k. The predicted particle set is $\hat{X}_t^l = \{\hat{x}_t^{lj}, j = 1, \cdots, N_l\}$, where $N_l = N' + N$ is the number of particles used in the linear part estimation.

Linear Part Resampling

The observation space Q_k^l of the linear part describes the local features in the image. For the j_{th} predicted particle, the measurement of the k_{th} lane is z_{kt}^{lj} which includes n_k^l pixels in Q_k^l that near the predicted lane. The observation model is derived from Eq. (3.4) with additional color information

$$p(z_{kt}^{lj} | \hat{x}_t^{lj}) = \frac{1}{n_k^l} \sum_i K_{ci} \cdot K_{\theta_{kt}^j} \cdot G_{li}(\rho_{kt}^j, \theta_{kt}^j), \tag{3.20}$$

where $K_{ci} = \mathcal{N}(\mu_c, \sigma_c^2)$ is the Gaussian kernel model of the color information, and μ_c and σ_c^2 are mean and covariance of this model. The weight for the j_{th} particle is

$$w_t^{lj} = \eta \prod_{k=1}^{n_l} n_k^l \, p(z_{kt}^{lj} | \hat{x}_t^{lj}), \tag{3.21}$$

where η is a normalization factor, which ensures that the weights sum to one. The number of candidate pixels n_k^l is considered as an important factor for weighting the particles, because only the pixels which have higher gradient magnitude values are selected as the candidate measurements.

Finally, the new particle set $X_t^l = \{x_t^{lj}, j = 1, \cdots, N_p\}$ is obtained by resampling $\hat{X}_t^l$ based on the weights where N_p is the number of particles used to estimate the parabolic parameters. Those particles in $\hat{X}_t^l$ that have high weights will be kept and the others that have lower weights will be removed from the particle set [Liu et al., 2010].

3.5.6 Parabolic Part Estimation

The algorithm of parabolic part estimation is similar to the linear part, but the number of particles and the observation model used for estimation can be different.

Prediction of Parabolic Part

The normal distribution is also used to model the state transition of curve part:

$$p(\hat{x}_t^{pj} | x_{t-1}^{lj}) = \mathcal{N}(A^p x_{t-1}^{lj}, \Sigma^p), \tag{3.22}$$

where function $\mathcal{N}$ is a normal distribution, the matrix A^p is an identity matrix and Σ^p is the covariance defined as:

$$\Sigma^p = diag\{V_k^p, k = 1, \cdots, n_l\}, \tag{3.23}$$

where $V_k^p = \{0, 0, \sigma_{d_k}^2\}$ and $\sigma_{d_k}^2$ is the covariance of the parabolic parameter d_k for the k_{th} lane. As a result, the predicted particle set $\hat{X}_t^p = \{\hat{x}_t^{pj}, j = 1, \cdots, N_p\}$ is obtained.

Resampling of Parabolic Part

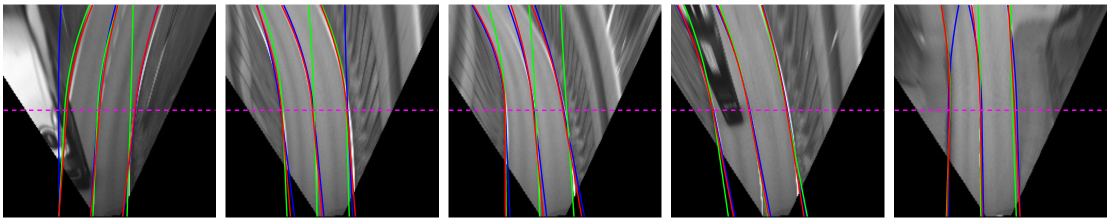
For the j_{th} particle, the measurement of the k_{th} lane in the far vision field observation space Q_k^p is z_{kt}^{pj}, which corresponds to n_k^p pixels in Q_k^p that near the predicted lane. The observation model can be derived from Eq. (3.17):

$$p(z_{kt}^{pj} | \hat{x}_t^{pj}) = \frac{1}{n_k^p} \sum_i K_{ci} \cdot G_{pi}(\rho_{kt}^j, \theta_{kt}^j, d_{kt}^j) \tag{3.24}$$

The weight for the j_{th} particle using above observation model is

$$w_t^{pj} = \eta\, A \prod_{k=1}^{n_l} n_k^p\; p(z_{kt}^{pj} | \hat{x}_t^{pj}) \tag{3.25}$$

where η is the normalization factor. This process is called *weighted resampling* because it has an additional term $A = \frac{1}{w_t^{lj}}$, such that it does not alter the distribution represented by the particle set [MacCormick and Isard, 2000]. However, this weighted resampling can introduce the impoverishment effect on the particle set [Duffner et al., 2009; Smith and Gatica-perez, 2004; Liu et al., 2011a]. To avoid impoverishment effect of original PPF algorithm, and enforce the importance



(a) IPM frame 38 (b) IPM frame 195 (c) IPM frame 470 (d) IPM frame 505 (e) IPM frame 748

(f) original frame 38 (g) original frame 195 (h) original frame 470 (i) original frame 505 (j) original frame 748

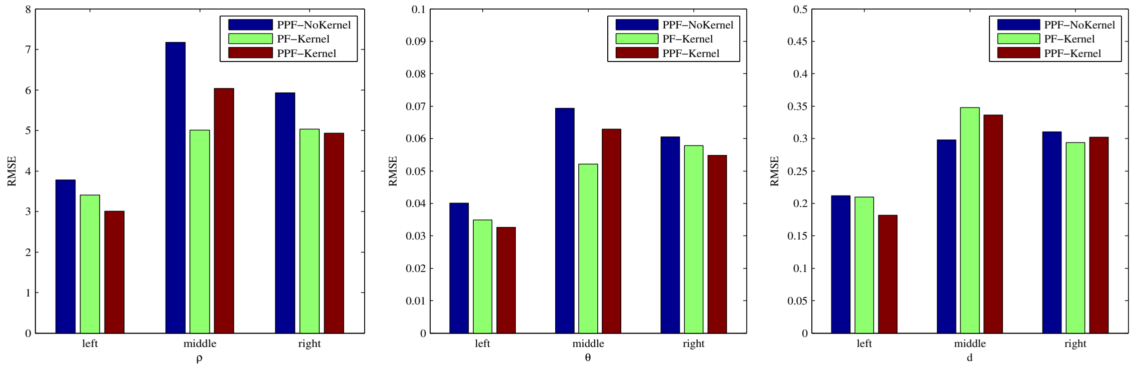
Figure 3.5: The estimation results are drawn on the IPM images of the Suburban Bridge dataset. The top images show the results estimated by PPF-Kernel (Partitioned Particle filter with kernel density based measurement, red), PF-Kernel (Particle filter with kernel density based measurement, green) and PPF-NoKernel (Partitioned Particle filter with color and edge measurement, blue). The bottom images are the original images from the camera.

of the linear part, we set $A = \frac{1}{n_k^l}$. After resampling on $\hat{X}_t^p$ by the weights, the new particle set $X_t = \{x_t^{pj}, j = 1, \cdots, N\}$ for the next iteration is obtained, and the weights are reset to be equal.

3.6 Results

In order to demonstrate the proposed algorithm, we selected two datasets from the DRIVSCO database [Baseski et al., 2009; Markelic et al., 2011]: Suburban Bridge dataset and Trailer dataset. These datasets contain a variety of challenging situations like high-curvature roads, partly marked and occluded lanes, etc.

As we mentioned in Section 3.5, the proposed algorithm has two highlights: one is the Partitioned particle filter (PPF), the other is the multiple kernel density based measurement model. In order to evaluate each highlight, we design the experiment in this way: first, we compare the PPF to the standard Particle filter (PF) using the same kernel density based measurement model. Second, we compare the multiple kernel density based measurement model to the color and edge based measurement model using the same PPF. Therefore, three algorithms are required to be implemented in the experiment, which are: the Partitioned Particle filter with the proposed multiple kernel density based measurement model (PPF-Kernel). the standard Particle filter with the proposed multiple kernel density based measurement model (PF-Kernel), the Partitioned Particle filter with color and edge based measurement model (PPF-NoKernel), In other words, we compare the PPF-Kernel to PF-Kernel to show the performance of the PPF, and we compare the PPF-Kernel to



(a) (b) (c)

Figure 3.6: The RMSE of estimated lane parameters (ρ, θ, d) by PPF-Kernel (Partitioned Particle filter with kernel density measurement), PF-Kernel (Particle filter with kernel density measurement) and PPF-NoKernel (Partitioned Particle filter with color and edge measurement) on Suburban Bridge dataset. $left$, *middle* and *right* mean the left lane, middle lane and right lane respectively.

PPF-NoKernel to show the performance of the kernel density based measurement model,

In order to have quantitative analyses, we manually choose the lane markers in the IPM frames (every 5 frame), then calculate the ground truth of lane parameters using the least-square. The root mean square error (RMSE) of the results from PPF-NoKernel, PF-Kernel and PPF-Kernel are shown in Fig. 3.6 for the Suburban Bridge dataset, and Fig. 3.8 for the Trailer dataset. On the other hand, we project the estimation results on the IPM images to allow visual inspection of the results, as shown in Fig. 3.5 for the Suburban Bridge dataset and Fig. 3.7 for the Trailer dataset. In the following sections, we will give more details and analyses on these estimation results.

3.6.1 PPF-Kernel *vs.* PPF-NoKernel

To show the performance of the proposed kernel density based measurement model, we compare the PPF-Kernel to the PPF-NoKernel. For the PPF-NoKernel, we only employ the color information in the measurement models which are defined as:

$$p(z_{kt}^{lj}|\hat{x}_t^{lj}) = \frac{1}{n_k^l} \sum_i K_{ci}, \tag{3.26}$$

$$p(z_{kt}^{pj}|\hat{x}_t^{pj}) = \frac{1}{n_k^p} \sum_i K_{ci}, \tag{3.27}$$

where $p(z_{kt}^{lj}|\hat{x}_t^{lj})$ and $p(z_{kt}^{pj}|\hat{x}_t^{pj})$ are measurement models for the linear part and the parabolic part respectively, n_k^l is the number of particles for the linear part, n_k^p is the number of particles for the parabolic part. $K_{ci} = \mathcal{N}(\mu_0, \sigma_0^2)$ is the Gaussian kernel model of color information. As in general lane markers have white color, i.e., high intensity value in the gray image, we set $\mu_0 = 1$ and $\sigma_0^2 = 0.5$ for the normalized image in the experiment.

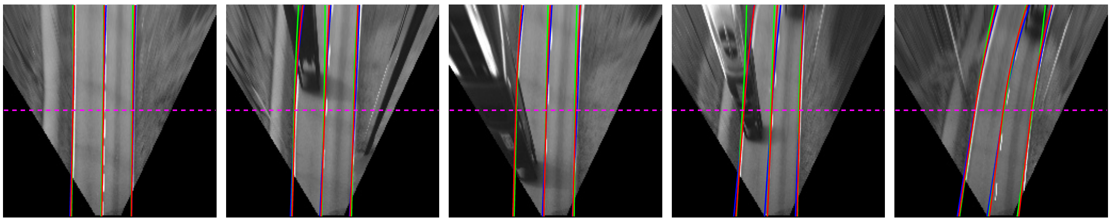
For the Suburban Bridge dataset, the RMSE of the estimated lane parameters is shown in Fig. 3.6. It is clear that the PPF-Kernel has smaller RMSE than the PPF-NoKernel for most of the lane parameters. We could also see the robustness of the PPF-Kernel against to the occlusions and the noise information from the bridge, whereas the PPF-NoKernel has larger error for the estimation of the parabolic part, as shown in Fig. 3.5a, Fig. 3.5b and Fig. 3.5e. On the other hand, the most challenge part is to estimate the left lane because of the occlusions by the car, as shown in Fig. 3.5a and Fig. 3.5d. However, the PPF still can recover from the occlusions. One reason is that we separate the lane into two parts. In spite of one part is occluded, the other part still can support the right particles. The other reason is we introduce the random particles into the algorithm loop, such that the PPF can recover from tracking failure . For the Trailer dataset, the curvature of the lanes is very small, thus the difference of the estimation results between the PPF-Kernel and PPF-NoKernel are not obvious, which can be seen in Fig. 3.7. However, the PPF-Kernel has smaller RMSE than the PPF-NoKernel for most of the lane parameters as shown in Fig. 3.8.

3.6.2 PPF-Kernel *vs.* PF-Kernel

To show the performance of the PPF, we compare the performance of the PPF-Kernel to the PF-Kernel in this section. Both of the PPF and PF employ the kernel density based measurement model. The PPF uses $N_l = 350$ particles to track the linear part and $N_p = 350$ particles to estimate the parabolic part. For every iteration, $N' = 50$ initialization particles drawn from default lane models, are introduced to the particle set. The covariances in the tests are $\sigma_{\rho_k} = 1$, $\sigma_{\theta_k} = 0.02$, and $\sigma_{d_k} = 0.05$. The weighted mean of particle state in the set X_t is used as the estimation result.

The standard PF uses $N = 350$ particles and $N' = 50$ initialization particles. The PF uses the same covariances as the PPF. The results show that the PF can handle small curvature lanes as good as the PPF, see estimation results in Fig. 3.7 for Trailer dataset, but performs worse when the curvature increases, see Fig. 3.5 for Suburban Bridge dataset. The reason is that the standard PF can not give accurate estimation when it only uses $N + N' = 400$ particles for the high dimensional state, i.e., 9 for three lanes in the experiments. However, the PPF fits the high curvature lanes nicely using 350 particles for the linear part and parabolic part respectively.

From the RMSE charts in Fig. 3.6 and Fig. 3.8, we can conclude that the PPF-Kernel has better estimation results than the PF-Kernel for the most of the lane parameters. One exception is in Fig. 3.6, where the (ρ, θ) estimated by the PF-Kernel for the middle lane has smaller RMSE than the PPF-Kernel. That is because the second resampling of the PPF and the linear part of middle lane has large curvature. The first resampling of the PPF keeps those particles that have better estimation for the linear part, but the second resampling of the PPF keeps the particles that have better estimation for the parabolic part. That is why the PPF-Kernel has larger RMSE of (ρ, θ), but



(a) IPM frame 20 (b) IPM frame 112 (c) IPM frame 123 (d) IPM frame 321 (e) IPM frame 446

(f) original frame 20 (g) original frame 112 (h) original frame 123 (i) original frame 321 (j) original frame 446

Figure 3.7: The estimation results are drawn on the IPM images of the Trailer dataset. The top images show the results estimated by PPF-Kernel (Partitioned Particle filter with kernel density based measurement, red), PF-Kernel (Particle filter with kernel density based measurement, green) and PPF-NoKernel (Partitioned Particle filter with color and edge measurement, blue). The bottom images are the original images from the camera.

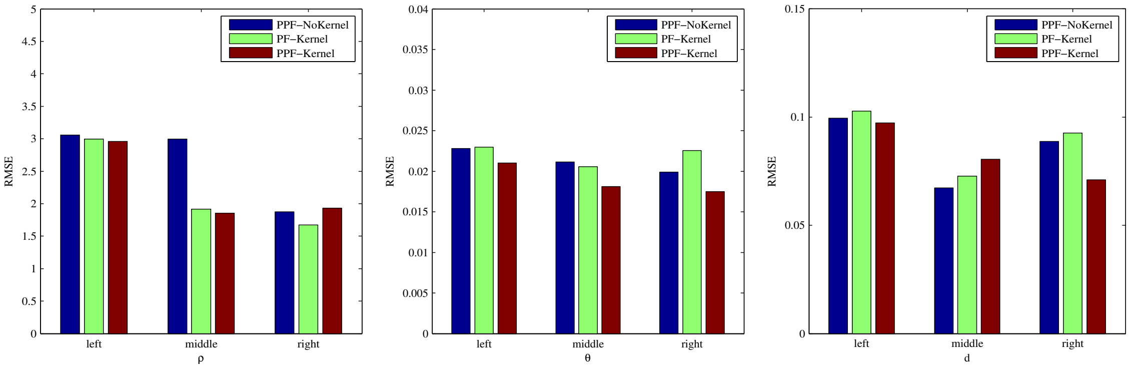
Table 3.1: The comparison of computation time

Dataset	Method	Run time per frame (s)
Suburban Bridge	PPF-NoKernel	$0.11s$
	PF-Kernel	$0.27s$
	PPF-Kernel	$0.62s$
Trailer	PPF-NoKernel	$0.13s$
	PF-Kernel	$0.29s$
	PPF-Kernel	$0.66s$

has smaller RMSE of d as shown in Fig. 3.8c.

3.6.3 The Computational Cost

The current algorithm is programmed on a modern computer with the Intel Core 2 Quad CPU and each CPU is $2.5GHz$ for simulation and analysis purpose. The comparison of computational time is shown in Table 3.1. The PPF with the simple color model is faster than the multi-kernel density model. The reason is that the Gauss-Hermitte numerical integral method in PPF is computation expensive. However, there are several ways to speed up the proposed algorithm. For example, the pixels in the observation space can be processed independently. Therefor it can be easily parallelized, and computed by the GPUs [Abramov et al., 2010].



(a) (b) (c)

Figure 3.8: The RMSE of estimated lane parameters (ρ, θ, d) by PPF-Kernel (Partitioned Particle filter with kernel density measurement), PF-Kernel (Particle filter with kernel density measurement) and PPF-NoKernel (Partitioned Particle filter with color and edge measurement) on Trailer dataset. $left$, *middle* and *right* mean the left lane, middle lane and right lane respectively.

3.7 Conclusion

In this chapter, we presented a framework of the linear-parabolic shape estimation using the PPF on the IPM frames, and a novel statistical measurement model is built using the multiple kernel density. The algorithm was tested on the DRIVSCO datasets and has shown its robustness concerning challenging scenes.

The advantages of our algorithm, as compared to the state of the art in this field, are threefold. First, we use not only the position and color information, but also the gradient information in the statistical observation model, which improves performance. Second, we estimate the linear-parabolic model in a hierarchical architecture using the PPF. This reduces the prerequisite number of particles. Third, the PPF maintains multiple hypotheses of the lane model and can recover from tracking failure, whereas the Kalman filter and least-squares based methods do not support this feature.

Although we have achieved notable progress, our current algorithm has some limits, for example, the simple multiple lane model, flat ground assumption and constant separate line y_m. Future work will require research on a robust multiple lane model [Nieto et al., 2008], online pitch angle estimation using the stereo vision [Danescu and Nedevschi, 2009] and adaptive separate line y_m estimation to improve the proposed algorithm.

Apart from the lane shape estimation, the proposed parabolic shape estimation using the multiple kernel density estimation can be also used for other applications, e.g., eyelid detection [Liu et al., 2009].

Chapter 4

Color Invariant Histograms

This chapter presents an evaluation framework for color invariant histograms for object tracking. Section 4.2 describes illumination changes by photometric analysis. The illumination changes are identified as five types based on the diagonal-offset model. Section 4.3 introduces eight color histograms and their invariance properties to the illumination changes. Section 4.4 discusses the histogram distance metric used in experiments. Three state-of-the-art trackers are used for evaluation in Section 4.5. The final results and analyses are shown in Section 4.6 and 4.7.

4.1 Introduction

The color histogram is one of the most popular region descriptors for object tracking. In general, the appearance of objects in an image can be represented by various colors defined in a specific color space, e.g., RGB (red, green and blue) space and HSV (hue, saturation and intensity value) space. The color distribution of objects can be approximated by a color histogram [Porikli, 2005], which counts the number of pixels that have colors in each of bins. Since color histograms are simple, effective to use and invariant to translation and rotation around the axis perpendicular to the image, they have been widely used for object tracking in the literature [Comaniciu et al., 2003; Cannons, 2008]. However, many color histograms are susceptible to illumination changes [Cannons, 2008]. For instance, the RGB color space is the most popular color space for color histograms, but it is one of the least robust to illumination changes [Cannons, 2008]. In contrast, the normalized RGB (nRGB) color histogram is invariant to intensity changes, e.g., shadow. Therefore, it is natural to choose the nRGB instead of RGB for object tracking to achieve a better performance. In this thesis, we address this problem by analyzing and evaluating a number of commonly used color histograms against illumination changes.

As far as we know, none of the previous work considers color invariant properties of color histograms in the scenario of object tracking. In the literature, most analyses and evaluations on the color invariant features to illumination changes are for object recognition and classification. Gevers and Smeulders [1999] studied the illumination invariance of color spaces for object recognition, and evaluated them under known illumination conditions. van de Sande et al. [2010] studied the invariance properties and distinctiveness of color descriptors (histograms and corners) in the context of image category recognition. Geusebroek et al. [2001] established color invariant descriptors based on differentials in the spectral and spatial domain, and tested the proposed color space on the static images. We have not found any work investigating color spaces related to object tracking scenarios. To achieve robust object tracking using color histograms, it is necessary to investigate and compare the performance of different color spaces.

Besides color invariant properties of color histograms, we are also interested in how to include spatial information in the color histograms. The normal color histograms do not preserve the spatial information of colors, which means the histograms do not know which color comes from which part of the image. To address this problem, Comaniciu et al. [2003] proposed a spatial masking with an isotropic kernel to regularize the RGB histogram representation. The spatial information is introduced into the histogram by the isotropic kernel, but this isotropic kernel is symmetric such that it is not sensitive to rotation. Teuliere et al. [2009] introduced a multiple kernel configuration with the RGB histogram, and gave evidences of its improved sensitivity over single kernel approaches. Khan et al. [2009] employed the anisotropic kernel with the RGB histogram instead of the isotropic kernel to cope with the rotation estimation. The proposed kernels indeed improve the performance of the RGB histogram, but these works have not considered the color invariance properties of the histograms.

In contrast to other works, we focus on color invariance and distinctiveness of color histograms for object tracking. In addition, the effects of the spatial kernels are also investigated. This Chapter addresses this problem by: a) studying the invariance properties and the distinctiveness of color histograms; b) evaluating the color histograms on large benchmark datasets; c) studying the effects of the kernel mask which adds the spatial information to the color histogram; d) investigating three state-of-the-art object tracking algorithms for evaluations: the integral histogram based exhaustive search, the kernel based mean shift and the particle filter.

4.2 Illumination Changes

The illumination changes in an image can be modeled by a linear transformation. von Kries [1970] introduced the diagonal model which describes the transformation between two images which have different illuminations. To include diffuse light effects, Finlayson et al. [2005] appended an additional term to the diagonal model, which is known as diagonal-offset model. Furthermore, van de Sande et al. [2010] introduced five types of illumination changes based on the diagonal-offset model. This section introduces the basic idea of the diagonal-offset model and discusses the definitions of illumination changes.

4.2.1 Diagonal-Offset Model

von Kries [1970] proposed a linear transformation for modeling illumination changes:

$$f^c = D^{u,c} f^u, \tag{4.1}$$

where f^u and f^c are the original image and the transformed image respectively, and $D^{u,c}$ is a diagonal matrix. The image f^u is taken under an unknown light source u, and the image f^c is taken under the reference light source c, which is also called canonical illuminant [van de Sande et al., 2010]. In case of the RGB color space, Eq. (4.1) can be represented by:

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix}, \tag{4.2}$$

where a, b and c act as factors describing intensity changes for three color channels. To include the effects from diffuse lights, the diagonal model was extended by Finlayson et al. [2005] to the diagonal-offset model:

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix} + \begin{pmatrix} o_1 \\ o_2 \\ o_3 \end{pmatrix}, \tag{4.3}$$

where (o_1, o_2, o_3) is the offset vector to three color channels. Obviously, the diagonal-offset model has three more parameters than the diagonal model, so it can deal with a wider range of lighting conditions [van de Sande et al., 2010].

4.2.2 Definition of Illumination Changes

Based on the diagonal-offset model, five basic types of the illumination changes can be defined [van de Sande et al., 2010], which are light intensity change, light intensity shift, light intensity change and shift, light color change, and light color change and shifts. The definitions of these five illumination changes are shown as follows.

- **Light intensity changes**: $a = b = c$ in Eq. (4.2). The shadow and shading in the image belong to this category.
- **Light intensity shifts**: $a = b = c = 1$ and $o_1 = o_2 = o_3$ in Eq. (4.3). As shown in van de Sande et al. [2010], the light intensity shifts normally include scattering of a white light source, object highlights, inter reflections, and infrared sensitivity of the camera sensor.
- **Light intensity changes and shifts**: $a = b = c$ and $o_1 = o_2 = o_3$ in Eq. (4.3).
- **Light color changes**: $a \neq b \neq c$ in Eq. (4.2).

- **Light color change and shifts**: $a \neq b \neq c$ and $o_1 \neq o_2 \neq o_3$ in Eq. (4.3).

4.3 Color Histograms

Given the definitions of the illumination changes, we can analyze the color invariance properties of color histograms. A color histogram depicts the color distribution of the objects in a specific color space, e.g., RGB, HSV and HSI. Therefore, the color constancy of color spaces determines the color invariance properties of color histograms. In this section, we describe eight types of color spaces and their corresponding histograms, and give the proofs of the color invariance properties. The summary of the color invariance properties of color histograms are shown in Fig. 4.1.

- **RGB histogram**. The RGB color space has three channels: red, green, and blue. A 3D histogram can be derived by calculating the number of pixels that have colors in a fixed range which depends on the number of bins. The RGB histogram has no illumination invariance properties [van de Sande et al., 2010].

- **HSV histogram**. The HSV color space has three channels: hue, saturation, and value (lightness). The formula to convert the RGB color space to HSV is [Smith, 1978]:

$$V = M \tag{4.4}$$

$$S = \frac{C}{V} \tag{4.5}$$

$$H = \begin{cases} \frac{G-B}{6\times C} & if M = R \\ \frac{B-R}{6\times C} + \frac{1}{3} & if M = G \\ \frac{R-G}{6\times C} + \frac{2}{3} & if M = B \end{cases} \tag{4.6}$$

where $M = max(R, G, B)$ and $C = M - min(R, G, B)$. In case of $H < 0$, then $H = 1 - H$. The saturation S is invariant to light intensity changes, which can be proved as follows:

$$\begin{aligned} S &= \frac{C}{V} \\ &= \frac{max(R^c, G^c, B^c) - min(R^c, G^c, B^c)}{max(R^c, G^c, B^c)} \\ &= \frac{max(aR^u, aG^u, aB^u) - min(aR^u, aG^u, aB^u)}{max(aR^u, aG^u, aB^u)} \\ &= \frac{max(R^u, G^u, B^u) - min(R^u, G^u, B^u)}{max(R^u, G^u, B^u)}. \end{aligned} \tag{4.7}$$

It is similar to prove that the hue H is invariant to light intensity and shift changes, but the intensity value V has no invariance property. We only use H and S channels in the experiment.

- **nRGB histogram**. The nRGB color space is the normalized RGB color space [Gevers and Smeulders, 1999]:

$$\begin{pmatrix} nR \\ nG \\ nB \end{pmatrix} = \begin{pmatrix} \frac{R}{R+G+B} \\ \frac{G}{R+G+B} \\ \frac{B}{R+G+B} \end{pmatrix} \tag{4.8}$$

  The nRGB histogram is invariant to light intensity change. Since the $nB = 1 - nR - nG$, we only use nR and nG channels in the experiment.

- **Opponent histogram**. The Opponent color space is defined by:

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix} \tag{4.9}$$

  The channels O_1 and O_2 are invariant to light intensity shift, which can be proven by:

$$\begin{aligned} \begin{pmatrix} O_1 \\ O_2 \end{pmatrix} &= \begin{pmatrix} \frac{R^c-G^c}{\sqrt{2}} \\ \frac{R^c+G^c-2B^c}{\sqrt{6}} \end{pmatrix} \\ &= \begin{pmatrix} \frac{R^u+o_1-G^u-o_1}{\sqrt{2}} \\ \frac{R^u+o_1+G^u+o_1-2B^u-2o_1}{\sqrt{6}} \end{pmatrix} \\ &= \begin{pmatrix} \frac{R^u-G^u}{\sqrt{2}} \\ \frac{R^u+G^u-2B^u}{\sqrt{6}} \end{pmatrix} \end{aligned} \tag{4.10}$$

  The third channel O_3 has no invariance properties, so we use O_1 and O_2 channels in the experiment.

- **Transformed histogram**. The invariants to light color change and shift can be achieved by normalizing the pixel value distribution [van de Sande et al., 2010]:

$$\begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} = \begin{pmatrix} \frac{R-\mu_R}{\sigma_R} \\ \frac{G-\mu_G}{\sigma_G} \\ \frac{B-\mu_B}{\sigma_B} \end{pmatrix}, \tag{4.11}$$

  where μ_i and σ_i are the estimated mean and standard derivation of the channel i on the region of interest. After the normalization, the color distribution of each channel is a normal distribution $\mathcal{N}(0, 1)$, so the Transformed histogram is invariant to light color change and shift.

- **Gaussian histogram**. The Gaussian color space was introduced by Geusebroek et al. [2001]:

$$\begin{pmatrix} E \\ E_\lambda \\ E_{\lambda\lambda} \end{pmatrix} = \begin{pmatrix} 0.06 & 0.63 & 0.27 \\ 0.3 & 0.04 & -0.35 \\ 0.34 & -0.6 & 0.17 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \tag{4.12}$$

  where the three components E, E_λ and $E_{\lambda\lambda}$ denote the intensity, blue-yellow and green-red channel. Although the features derived by spatial differential on the Gaussian color space could offer some invariance properties [Burghouts and Geusebroek, 2009], the Gaussian color histogram itself has no invariance properties to illumination changes.

- **Spherical histogram**. The Spherical color space given in [van de Weijer et al., 2006] is:

$$\begin{pmatrix} \theta \\ \phi \\ r \end{pmatrix} = \begin{pmatrix} arctan\left(\frac{G}{R}\right) \\ arcsin\left(\frac{\sqrt{R^2+G^2}}{\sqrt{R^2+G^2+B^2}}\right) \\ \sqrt{R^2+G^2+B^2} \end{pmatrix}, \tag{4.13}$$

  The first two channels θ and ϕ are invariant to the light intensity change, but the third channel r has no invariance properties, so we only use θ and ϕ channels in the experiment.

- **HSI histogram**. The HSI color space is defined by:

$$\begin{pmatrix} h \\ s \\ i \end{pmatrix} = \begin{pmatrix} arctan\left(\frac{O_1}{O_2}\right) \\ \sqrt{O_1^2+O_2^2} \\ O_3 \end{pmatrix}, \tag{4.14}$$

  where O_1, O_2 and O_3 are defined in Eq. (4.9). Because the O_1 and O_2 are invariant to light intensity shift, the components h and i also have this property. In addition, h is invariant to the light intensity changes. In the experiment, we only use h and s channels .

4.4 Distance Metrics

For histogram based object tracking, first we need to learn a histogram of the known target object from one image or a set of images. This learned histogram is called target histogram. Then we can find the object in the other images by matching the target histogram to the candidate histograms using distance metrics. In this chapter, we uses the Bhattacharyya distance, which is nearly optimal due to its link to the Bayes error [Comaniciu et al., 2003].

The accuracy of the matching depends not only on the color space, but also on the distance metrics between the target histogram and the candidate histogram. Here we use five commonly used

Histograms		intensity change	intensity shift	intensity change and shift	color change	color change and shift
RGB	R	-	-	-	-	-
	G	-	-	-	-	-
	B	-	-	-	-	-
HSV	H	+	+	-	-	-
	S	+	-	-	-	-
	V	-	-	-	-	-
nRGB	nR	+	-	-	-	-
	nG	+	-	-	-	-
	nB	+	-	-	-	-
Opponent	O_1	-	+	-	-	-
	O_2	-	+	-	-	-
	O_3	-	-	-	-	-
Transformed	T_1	+	+	+	+	+
	T_2	+	+	+	+	+
	T_3	+	+	+	+	+
Gaussian	E	-	-	-	-	-
	E_λ	-	-	-	-	-
	$E_{\lambda\lambda}$	-	-	-	-	-
Spherical	θ	+	-	-	-	-
	ϕ	+	-	-	-	-
	r	-	-	-	-	-
HSI	h	+	+	-	-	-
	s	-	+	-	-	-
	i	-	-	-	-	-

Figure 4.1: The illumination invariance properties of the color histograms. "+" means invariance and "-" means lack of invariance.

distance metrics. The Bhattacharyya distance between the target histogram H_t and the candidate histogram H_c is defined by:

$$d(H_t, H_c) = \frac{1}{\sqrt{\bar{H}_t \bar{H}_c n^2}} \sum_{i}^{n} \sqrt{H_t(i) H_c(i)}, \tag{4.15}$$

where n is the number of bins in the histogram. Here we assume H_t and H_c have the same number of bins.

4.5 Experimental Setup

To compare the color histograms, we employ three state-of-the-art trackers: exhaustive search, mean shift and the particle filter. For all trackers, the target histogram is calculated from the manually selected object image in the first frame, where the scale of the object is constant. The experimental setup for each tracker is described in the following.

4.5.1 Exhaustive Search

The exhaustive search might be one of the easiest ways to do object tracking. It searches all image parts, and compares candidate histograms from each part to the target histogram. Of course, the computational cost of the exhaustive search is high.

Porikli [2005] introduced a faster way to calculate image histograms using the integral histogram. The integral histogram based searching first calculates an integral histogram image which has the same size as the original image, but each point in this integral histogram image represents a histogram of the rectangle image area from the left-top corner to the current pixel. In this way, the histogram of any rectangular subregion can be easily calculated using simple arithmetics which are independent of the size of the region.

4.5.2 Mean Shift

Comaniciu et al. [2003] introduced a kernel based mean shift tracking method. As we mentioned in Section 4.1, an isotropic kernel is used for masking the target representation. The histogram from the target image is weighted by this kernel. The similarity function used in the kernel based mean shift is the Bhattacharyya distance, then a gradient optimization method is used to localize the position of the object in next frames.

The kernel based mean shift is much faster than the exhaustive search, because it only evaluates the similarity within a limited search region (kernel mask) [Porikli, 2005]. Another advantage of the mean shift is that the spatial information of the object is added to the histogram using kernel mask. In this chapter, we employ the kernel with Epanechnikov profile [Comaniciu and Meer, 2002]:

$$k(x) = \begin{cases} \frac{1}{2} c_d^{-1}(d+2)(1-\|x\|^2) & if\ \|x\| \leq 1 \\ 0 & otherwise \end{cases} \tag{4.16}$$

where c_d is the volume of a d dimensional sphere, e.g., $c_d = 2\pi$ when $d = 2$, and $\|x\|$ is the normalized distance to the coordinate center. Fig. 4.2 shows an example of the Epanechnikov kernel for a 100 × 80 image.

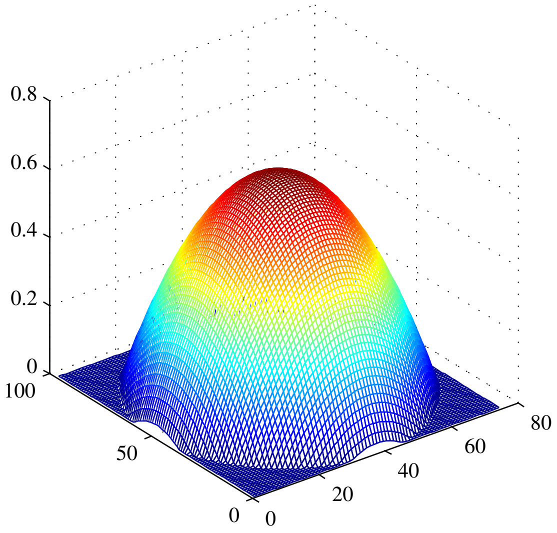


Figure 4.2: The Epanechnikov kernel mask for a 100 × 80 image.

4.5.3 Particle Filter

Nummiaro et al. [2003] proposed an adaptive particle filter using weighted color histograms. The particle filter employs a number of particles to approximate the real probability distribution of the system state, e.g., the coordinates of the object center. Each particle is a hypothesis of the state. The particle filter has two steps: the first step is prediction using the motion model of the object. The second step is the measurement update using the measurement function, e.g., Bhattacharyya distance for weighted color histograms.

The particle filter can deal with rapid movement if the system dynamics represents the movement of the object very well [Nummiaro et al., 2002]. On the other hand, the particle filter tracks multiple hypotheses, so it is more reliable for tracking objects in a cluttered environment.

In our experiment, the state of the particle filter is defined by $X = \{x, y\}$ which is the coordinates of the object center. The dynamics of the system between frames is given by a normal distribution:

$$X_t^i = AX_{t-1}^i + R, \tag{4.17}$$

where X_t^i is the state of i_{th} particle at time t. A is an identity matrix and R is the covariance of the noise in the prediction step. In the experiment we define $R = diag\{15^2, 15^2\}$ where $diag$ is the diagonal function. The measurement function used for weighting particles is defined by:

$$w_t^i = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(d^i-\mu)^2}{2\sigma^2}}, \tag{4.18}$$

where d^i is the distance metric value defined in Section 4.4 for the i_{th} particle. The histograms are weighted by the kernel mask defined in Eq. (4.16). We define $\sigma = 0.2, \mu = 1$ for the Bhattacharyya distance.

After resampling on the particle set $\{X_t^i, i = 1, \cdots, n\}$, a new particle set $\{\hat{X}_t^i, i = 1, \cdots, n\}$ is obtained where n is the number of particles. We set $n = 200$ in the experiment. The final estimated result $\bar{X}_t$ is derived by the weighted mean of all particles:

$$\bar{X}_t = \sum_i^n w_t^i \hat{X}_t^i. \tag{4.19}$$

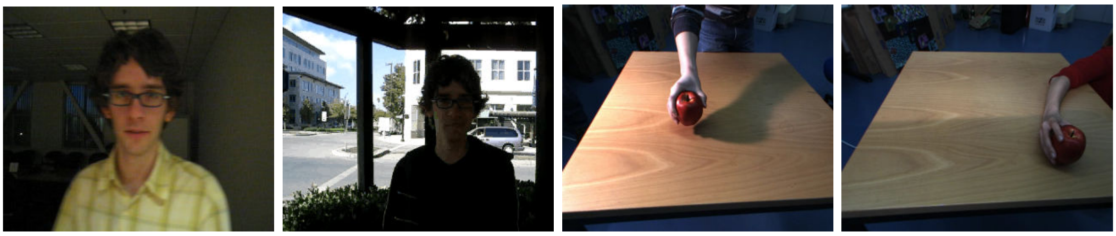
4.5.4 Evaluation Criteria

Given the ground truths, we want to known how good the tracker is, so a criterion to measure the goodness of localization is required. Here we use the same measurement criterion as in [Martinkauppi et al., 2002]:

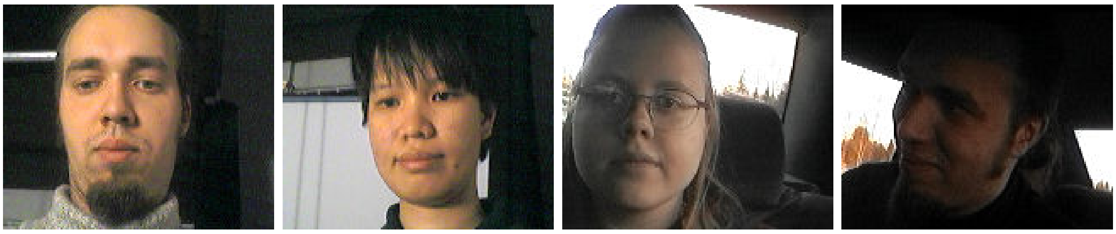
$$E_o = \frac{A_{ov}}{\sqrt{A_{gt}A_{tr}}}, \tag{4.20}$$

where E_o is the similarity between the tracking result and the ground truth, A_{gt} is the area of the ground truth bounding box, A_{tr} is the area of the tracked bounding box, and A_{ov} is the overlap area of the ground truth and tracking bounding box. In our experiment, we assume that the scale of the object does not change, i.e., A_{tr} is constant for a certain dataset.

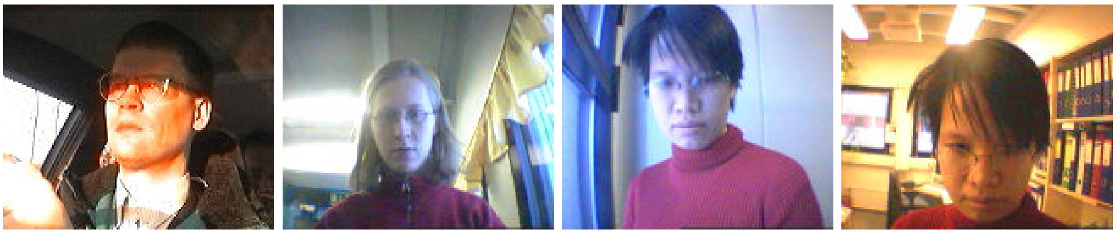
4.5.5 Datasets



(a) David indoor (b) David outdoor (c) Liu apple (d) Irene apple



(e) almov1 (f) almov2 (g) almov9 (h) almov11



(i) almov12 (j) nomov1 (k) nomov2 (l) nomov3

Figure 4.3: The benchmark datasets used in the evaluation experiment. The dataset *nomov4* is not shown here because it is not permitted to publish.

To compare the color histograms, three benchmark datasets are used in our experiment: *Toronto David* [Ross et al., 2008], *BCCN Apple* and *Oulu face* [Martinkauppi et al., 2002] which have many kinds of illumination changes, e.g., light intensity change (shadow and shading), light color change (different color temperature), light illumination direction change and uneven lights. The ground truth data of Oulu face is available online [Martinkauppi et al., 2002]. For the other two datasets, we get the ground truth by manually selecting the object in the images. Some sample images are shown in Fig. 4.3.

The *Toronto David* has two subsets for face tracking: *David indoor* and *David outdoor*. In the subset *David indoor*, the main light source is a fluorescent lamp. The people walk from the shadow to the light with different poses, so the light intensity changes and the illumination direction changes. For the other subset *David outdoor*, the main light source is natural sun light. The person walks underneath a trellis with large illumination change and casts shadows while changing its pose.

The *BCCN Apple* has two subsets for apple tracking: *Liu apple* and *Irene apple*. The light source is the indoor incandescent lamp. The light intensity changes in the frames, and a strong interference occurs in the *Irene apple* sequences where the apple has a similar color as the person's cloth.

The *Oulu face* has nine subsets for face tracking: *almov1*, *almov2*, *almov9*, *almov11*, *almov12*, *nomov1*, *nomov2*, *nomov3*, and *nomov4*. These frames are recorded under different illumination conditions: *almov1* and *almov2* were recorded in the indoor environment with light color changes. *almov9*, *almov11* and *almov12* were recorded in a car but under the natural sun light. *nomov1*, *nomov2*, *nomov3*, and *nomov4* were recorded in the indoor environment with light color change and pose change.

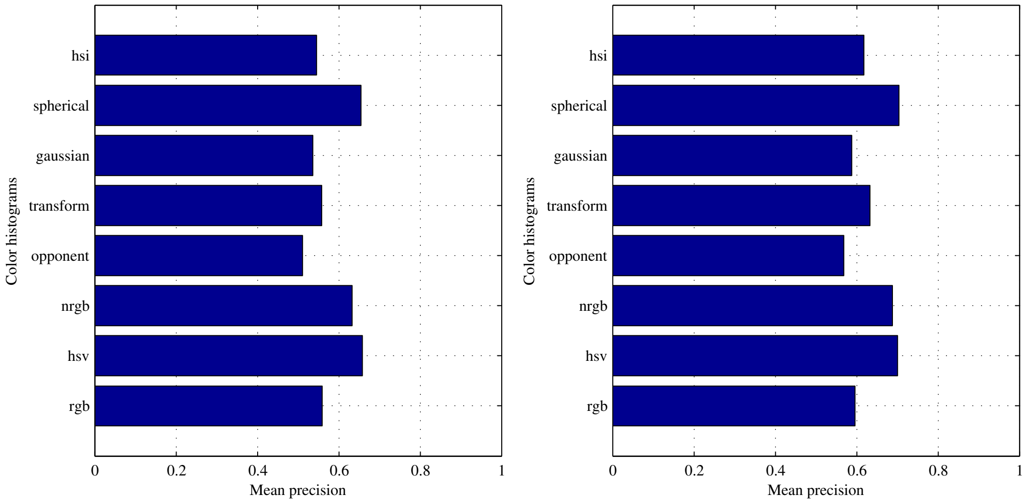
4.6 Results

4.6.1 Exhaustive Search

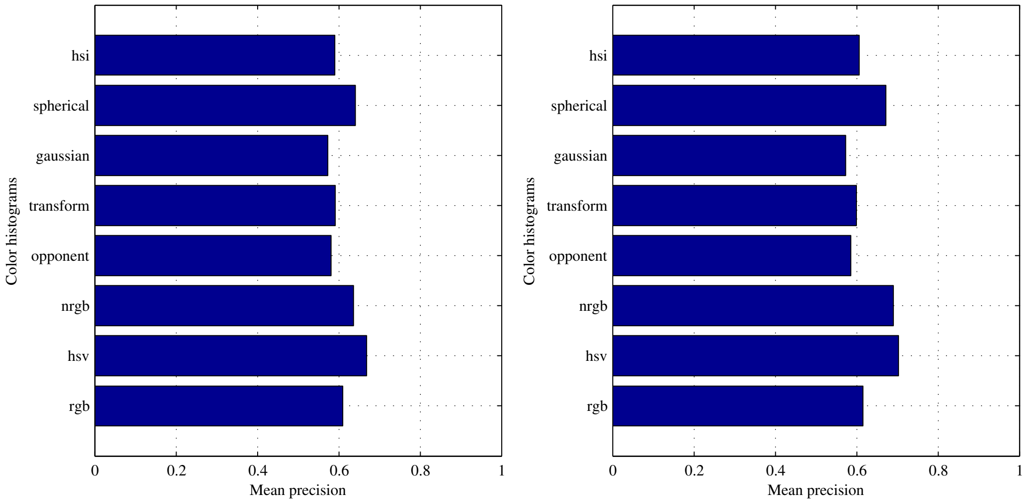
The exhaustive search is a method that will search all sub-regions in the image and get the optimum candidate by comparing the distance metric. The goodness value E_o for each candidate is calculated by Eq. (4.20).

To have a statistical analysis on the performance of each color histogram on a dataset, we compute the mean of E_o on all the frames which is shown in Fig. 4.4a. From the result, we conclude that the histograms that have color invariance properties show better performance than the others, e.g., nRGB is better than RGB and Gaussian. The top three best color histograms are HSV, Spherical and nRGB color histograms, which shows good performance not only on the images that have shadow and shading, but also on those that the background has similar appearance as the object. The Transformed color histogram is only one that is invariant to all illumination changes, see Fig. 4.1, but it does not work as well as expected. The reason is that we use the whole image to estimate the mean and variance value, whereas the illuminations for subregions of the image might come from different light sources, so the estimations might not correct for the subregions.

Since each video has variety illumination conditions, it is hard to give a conclude about which color histogram is the best one for the specific illumination condition. However, we can give a general recommendation for the object tracking under variety illumination conditions. The HSV, Spherical and nRGB color histograms are recommended based on the results which are showed in



(a) Exhaustive search (b) Mean shift



(c) Particle filter without kernel (d) Particle filter with kernel

Figure 4.4: The comparison between different color histograms using the (a) exhaustive search, (b) mean shift, (c) particle filter without kernel and (d) particle filter with kernel.

Fig. 4.4a

4.6.2 Mean Shift

The mean shift uses a kernel mask to include the spatial information for the color histograms, and gives high weight values to the central pixels of the object. The results using different color histograms are shown in Fig. 4.4b. The mean shift indeed gives better results than the exhaustive search, since the spatial information is encoded in the histograms. Another reason is that the mean shift only evaluates the pixels nearby the kernel center, based on the assumption that the object moves slowly between frames, which makes the mean shift more robust against background noise. The results show that the HSV, Spherical and nRGB perform better than the others.

4.6.3 Particle Filter

The particle filter uses a random walk motion model as defined in Eq. (4.17), which equals to randomly select the candidates from the nearby region. The particle filter can handle larger motion than the mean shift, and has less computational cost than the exhaustive search.

In order to study the effects of the kernel mask for the color histogram based object tracking, the particle filter runs twice with and without the kernel mask. The tracking results are shown in Fig. 4.4c and in Fig. 4.4d respectively. Again, the HSV, Spherical and nRGB have better performance than the others. In addition, it is clear that the particle filter with the kernel performs better than the one without the kernel.

4.7 Conclusion

In this chapter, we presented our evaluation framework on the color invariance properties of color histograms for the object tracking. The experiments show that the HSV, Sperical and nRGB color histograms perform better than the others for object tracking in three tracker algorithms. They have invariant properties against the illumination changes, and show good distinctive capability compared to the others. Although the Transformed histogram is invariant to all types illumination changes, it does not work well in our experiment because the true mean and variance of the color distribution are difficult to estimate, and sensitive to the background noise.

The experiment shows that the performance of the color histograms can be improved by using the kernel mask. Because the spatial information of the color is embedded into the histogram. The particle filter with the kernel performs better than without the kernel. On the other hand, the mean shift and particle filter use the motion information of the object for object tracking, so they are more robust than the exhaustive search against the background noise.

Chapter 5

Multiple Sensor Fusion

This chapter introduces new sigma-points information filters to improve measurements using multiple sensors for object tracking. First, we present our central difference information filter (CDIF) algorithm for nonlinear estimation and multiple sensor fusion in Section 5.2, which has fewer predefined parameters than the unscented information filter (UIF) introduced by [Lee, 2008a]. Then the square-root extensions of CDIF and UIF are proposed in Section 5.3 and Section 5.4 respectively. These square-root forms have better numerical stability than the orginal ones. Simulation results of target tracking are presented and discussed in Section 5.5 and 5.6.

5.1 Introduction

The accuracy and robustness of control systems can be improved using fused information from multiple sensors. Therefore, sensor fusion techniques have been widely studied in many research fields, i.e., robot navigation, surveillance, and intelligent vehicles [Lee, 2008b]. Recently, the information filter (IF), which is the dual of the Kalman filter (KF), has attracted much attention for multiple sensor fusion [Wang et al., 2010b]. Both the IF and the KF represent distributions of random state variables with Gaussians. However, in contrast to moment parametrization as done in the KF, the IF uses an information matrix and an information vector to represent the Gaussians. This difference in parameterization makes the IF superior to the KF concerning multiple sensor fusion, as computations are simpler and no prior information of the system state is required [Lee, 2008a].

In the case of nonlinear estimation problems, an extended version of the IF can be obtained using the first order term of the Taylor series expansions of the nonlinear functions, i.e., the dynamic and measurement functions of the system, which is called extended information filter (EIF). This approximation can introduce large errors when the system model is highly nonlinear, and the higher order terms of Taylor series are important [Van der Merwe, 2004]. To address this issue, the unscented information filter (UIF) has been proposed by Kim et al. [2008] and Lee [2008a]. Kim et al. [2008] developed the UIF by using minimum mean square error estimation. In contrast, Lee's UIF algorithm is derived by embedding statistical linear error propagation into the EIF architecture. Although their methods are different, results are essentially identical [Lee, 2008a; Kim et al., 2008; Liu et al., 2011b]. The UIF uses a number of deterministic sigma points to capture the true information matrix and the information vector, which can be accurate up to the second order of any nonlinearity. However, three parameters (α, β, κ) must be defined for the UIF, which depend on the system models. As shown in [Lee, 2008a; Wang et al., 2010a], the UIF is superior to the EIF not only in terms of estimation accuracy but also concerning the convergence speed for nonlinear estimation and multiple sensor fusion. However, the choice of system parameters (α, β, κ) can affect the filter's estimation precision.

In this chapter, we first propose an alternative to the UIF, which we call *central difference information filter* (CDIF). Where the UIF uses the unscented transform to compute the sigma points, the CDIF employs Stirling's interpolation. As proved in [Van der Merwe, 2004], Stirling's interpolation based central difference Kalman filter (CDKF) has the same or superior performance as the unscented transform based Kalman filter (UKF), with one advantage over the UKF: Stirling's interpolation only needs a single parameter, the interval size h, whereas the unscented transform needs three [Zhu et al., 2009].

Second, we propose to use square-root forms for both UIF and CDIF, which have shown improved numerical characteristics compared to their regular forms. Here we call them square-root unscented information filter (SRUIF) and square-root central difference information filter (SR-CDIF) respectively. The square-root filters predict and update the square-root covariance instead of the full covariance. In this way, the square-root filters achieve better numerical characteristics than the regular ones, e.g., improved numerical accuracy, double order precision and preservation of symmetry [Arasaratnam and Haykin, 2009]. The first square-root filter was developed by Potter and Stern. [1963] and was used in the Apollo manned mission [Maybeck, 1979]. Since then, many square-root extensions of conventional filters have been introduced and analyzed. Our work was inspired by Van der Merwe [2004] who proposed square-root forms of sigma-point Kalman filters. Here we introduce the square-root extensions of UIF and CDIF and their numerical advantages for solving nonlinear state estimation.

5.2 Central Difference Information Filter

In this section, we present our CDIF framework, which replaces the unscented transform with Stirling's interpolation to generate the sigma points. The algorithm includes three steps: prediction, measurement update and global information fusion.

5.2.1 Stirling's Interpolation

Stirling's interpolation has been used previously with the Kalman filter in the literature, referred to as the *central difference Kalman filter* (CDKF) [Zhu et al., 2009; Van der Merwe, 2004]. The CDKF uses a symmetric set of $2L+1$ sigma points to approximate nonlinear functions. In the case of Gaussian distributions of the system variables, the mean and covariance can be represented by those sigma points. As we mentioned in Section 5.1, the IF is a dual filter of the KF, such that the information vector and matrix also can be derived by those sigma points. In this section, we first show how the mean and covariance are derived using Stirling's interpolation, then show how the information vector and matrix are obtained from the mean and covariance.

The $2L + 1$ prior sigma points used in Stirling's interpolation step are given by the prior mean $\hat{x}$ plus or minus the columns of the scaled square root of the prior covariance matrix P_x [Van der Merwe, 2004]:

$$\mathcal{X}_i = \begin{cases} \hat{x}, & i = 0 \\ \hat{x} + (h\sqrt{P_x})_i, & i = 1, \cdots, L \\ \hat{x} - (h\sqrt{P_x})_i, & i = L+1, \cdots, 2L \end{cases} \tag{5.1}$$

where h is a scaling parameter and L is the dimension of the state $\hat{x}$. The subscript i indicates the ith column of the matrix. A set of the posterior sigma points can be derived by propagating these prior sigma points through the nonlinear function g: $\mathcal{Z}_i = g(\chi_i)$. Furthermore, the estimations of mean $\hat{z}$, covariance P_z and cross-covariance P_{xz} are obtained as follows:

$$\bar{z} \approx \sum_{i=0}^{2L} w_i^{(m)} \mathcal{Z}_i \tag{5.2}$$

$$\begin{aligned} P_z \approx & \sum_{i=1}^{L} w_i^{(c_1)} (\mathcal{Z}_i - \mathcal{Z}_{i+L})(\mathcal{Z}_i - \mathcal{Z}_{i+L})^T \\ & + \sum_{i=1}^{L} w_i^{(c_2)} (\mathcal{Z}_i + \mathcal{Z}_{i+L} - 2\mathcal{Z}_0)(\mathcal{Z}_i + \mathcal{Z}_{i+L} - 2\mathcal{Z}_0)^T \end{aligned} \tag{5.3}$$

$$P_{xz} \approx \sqrt{w_1^{(c1)} P_x} (\mathcal{Z}_{1:L} - \mathcal{Z}_{L+1:2L})^T. \tag{5.4}$$

The corresponding weights for the mean and covariance are defined as

$$\begin{aligned} w_0^{(m)} &= \tfrac{h^2-L}{h^2} \\ w_i^{(m)} &= \tfrac{1}{2h^2}, \\ w_i^{(c1)} &= \tfrac{1}{4h^2}, \\ w_i^{(c2)} &= \tfrac{h^2-1}{4h^4}, \quad i = 1, \cdots, 2L \end{aligned} \tag{5.5}$$

As proved in [Van der Merwe, 2004], if the random variables obey a Gaussian distribution, the optimal value of h is $\sqrt{3}$. Stirling's interpolation only depends on one parameter, the interval size h, in contrast to three parameters (α, β, κ) which are required in the unscented transform. This makes Stirling's method simpler and easier to tune.

5.2.2 Prediction

Here we consider the discrete-time nonlinear dynamic system

$$x_k = F(x_{k-1}, v), \tag{5.6}$$

where x_k is the state vector of the system at time step k, and $v \sim \mathcal{N}(\bar{v}, R_v)$ is Gaussian noise.

First, the state vector is augmented with the noise variable and the corresponding augmented covariance matrix is derived by:

$$x_{k-1}^{a_v} = \begin{bmatrix} x_{k-1} \\ \bar{v} \end{bmatrix}, \; P_{k-1}^{a_v} = \begin{bmatrix} P_{k-1} & 0 \\ 0 & R_v \end{bmatrix}. \tag{5.7}$$

A symmetric set of $2L + 1$ sigma points is generated:

$$\mathcal{X}_{i,k-1}^{a_v} = \begin{cases} x_{k-1}^{a_v}, & i = 0 \\ x_{k-1}^{a_v} + (h\sqrt{P_{k-1}^{a_v}})_i, & i = 1, \cdots, L \\ x_{k-1}^{a_v} - (h\sqrt{P_{k-1}^{a_v}})_i, & i = L+1, \cdots, 2L \end{cases} \tag{5.8}$$

where h is a scaling parameter and L is the dimension of the state $x_{k-1}^{a_v}$. The subscript i indicates the i_{th} column of the matrix. Each sigma point $\mathcal{X}_{i,k-1}^{a_v}$ contains the state and noise variable components

$$\mathcal{X}_{i,k-1}^{a_v} = \begin{bmatrix} \mathcal{X}_{i,k-1}^{x} \\ \mathcal{X}_{i,k-1}^{v} \end{bmatrix}. \tag{5.9}$$

These sigma points are further passed through the nonlinear function Eq. (5.6), such that the predicted sigma points for the discrete time k are derived as:

$$\mathcal{X}_{i,k|k-1}^{x} = F(\mathcal{X}_{i,k-1}^{x}, \mathcal{X}_{i,k-1}^{v}). \tag{5.10}$$

Finally, the first two moments of the predicted state vector are obtained by weighted sum of the transformed sigma points:

$$\hat{x}_k = \sum_{i=0}^{2L} w_i^m \mathcal{X}_{i,k|k-1}^x \tag{5.11}$$

$$\hat{P}_k = \sum_{i=1}^{L} w_i^{(c_1)} \alpha_i \alpha_i^T + \sum_{i=1}^{L} w_i^{(c_2)} \beta_i \beta_i^T, \tag{5.12}$$

where $\alpha_i = \mathcal{X}_{i,k|k-1}^x - \mathcal{X}_{i+L,k|k-1}^x$ and $\beta_i = \mathcal{X}_{i,k|k-1}^x + \mathcal{X}_{i+L,k|k-1}^x - 2\mathcal{X}_{0,k|k-1}^x$. The corresponding weights for the mean and covariance are defined as

$$\begin{aligned} w_0^{(m)} &= \tfrac{h^2-L}{h^2} \\ w_i^{(m)} &= \tfrac{1}{2h^2}, \\ w_i^{(c1)} &= \tfrac{1}{4h^2}, \\ w_i^{(c2)} &= \tfrac{h^2-1}{4h^4}, \quad i = 1, \cdots, 2L \end{aligned} \tag{5.13}$$

where $h \geq 1$ is the scalar central difference step size. If the random variables obey a Gaussian distribution, the optimal value of h is $\sqrt{3}$ [Van der Merwe, 2004].

As stated in [Anderson and Moore, 1979], the information matrix and information vector are the dual of the mean and covariance, so that the predicted information matrix $\hat{Y}_k$ and the information vector $\hat{y}_k$ are derived as:

$$\hat{y}_k = \hat{Y}_k \, \hat{x}_k \tag{5.14}$$

$$\hat{Y}_k = (\hat{P}_k)^{-1}. \tag{5.15}$$

5.2.3 Measurement Update

The measurement function of the nonlinear system is defined as

$$z_k = H(x_k) + n, \tag{5.16}$$

where z_k is the measurement and $n \sim \mathcal{N}(\overline{n}, R_n)$ is the Gaussian noise of the measurement.

The sigma points used for the measurement update are derived as:

$$\mathcal{X}_{i,k|k-1} = \begin{cases} \hat{x}_k, & i = 0 \\ \hat{x}_k + (h\sqrt{\hat{P}_k})_i, & i = 1, \cdots, L \\ \hat{x}_k - (h\sqrt{\hat{P}_k})_i, & i = L+1, \cdots, 2L \end{cases} \tag{5.17}$$

The predicted measurement points are obtained by transforming the sigma points through Eq. (5.16)

$$\mathcal{Z}_{i,k|k-1} = H(\mathcal{X}_{i,k|k-1}). \tag{5.18}$$

Furthermore, the mean and cross-covariance are derived by:

$$\hat{z}_k = \sum_{i=0}^{2L} w_i^m \mathcal{Z}_{i,k|k-1} \tag{5.19}$$

$$\hat{P}_{x_k z_k} = \sqrt{w_1^{(c1)} \hat{P}_k} (\mathcal{Z}_{1:L} - \mathcal{Z}_{L+1:2L})^T. \tag{5.20}$$

Finally, the measurement update of the information vector and the information matrix are derived as:

$$y_k = \hat{y}_k + \phi_k \tag{5.21}$$

$$Y_k = \hat{Y}_k + \Phi_k \tag{5.22}$$

where ϕ_k and Φ_k are information contribution terms for the information vector and matrix respectively, which can be derived by:

$$\phi_k = \hat{Y}_k \hat{P}_{x_k z_k} R_n^{-1} (z_k - \hat{z}_k + \hat{P}_{x_k z_k}^T \hat{y}_k) \tag{5.23}$$

$$\Phi_k = \hat{Y}_k \hat{P}_{x_k z_k} R_n^{-1} (\hat{P}_{x_k z_k})^T (\hat{Y}_k)^T. \tag{5.24}$$

The derivation of Eq. (5.23) and Eq. (5.24) are referred to [Lee, 2008a; Kim et al., 2008].

5.2.4 Global Information Fusion

For multiple sensor fusion, if the measurement noises between the sensors are uncorrelated, the measurement update for information fusion is simply expressed as a linear combination of the local information contribution terms [Chong, 1979]:

$$y_k = \hat{y}_k + \sum_{i=1}^{N} \phi_{i,k} \tag{5.25}$$

$$Y_k = \hat{Y}_k + \sum_{i=1}^{N} \Phi_{i,k}, \tag{5.26}$$

where N is the number of sensors. Eq. (5.25) and Eq. (5.26) show the main advantage of the information filters, which is the efficient measurement update. This superiority makes information filters more suitable for multiple sensor fusion than the Kalman filters. Note that the information matrix Y_k is the inverse of the covariance matrix P_k as shown in Eq. (5.15). When there is no prior information concerning the initial state, the Kalman filters are hard to cope with this situation since P_k is infinite. However, the information filters can deal with this special situation well with $Y_k = (P_k)^{-1} = 0$. Other comparisons between information filters and Kalman filters can be found in [Anderson and Moore, 1979].

5.3 Square-Root Central Difference Information Filter

The CDIF requires the square-root of the covariance to calculate the sigma-points in each discrete time update and measurement update, as shown in Eq. (5.8) and Eq. (5.17). The square-root operation is computationally expensive and demands that the covariance matrix must be positive semi-definite. To avoid the square-root operation and improve the numerical stability, we introduce the square-root central difference information filter (SRCDIF).

The square root form has important numerical advantages over the regular one: First, since the square-root of the covariance matrix is directly available, the SRCDIF saves computational cost for generating the sigma-points. Second, the numerical accuracy is improved because the condition number of the square root of the covariance matrix is only half of the covariance matrix [Anderson and Moore, 1979]. Third, the square-root filters can achieve twice the effective precision of the regular forms [Arasaratnam and Haykin, 2008]. Fourth, the symmetry and nonnegative properties of the covariance matrix are kept [Anderson and Moore, 1979].

5.3.1 SRCDIF for State Estimation

The SRCDIF benefits from three powerful matrix factorization techniques: *QR decomposition*, *Cholesky factor updating* and *efficient least squares*. In the following, we will use *qr*, *chol*, *cholupdate* to refer to the *QR decomposition*, *Cholesky decomposition*, and *Cholesky factor updating* respectively

- *QR decomposition*. In the CDIF, the square-root of the covariance matrix S is derived by *Cholesky decomposition* on P: $S = chol(P)^T$ where S is a lower triangular matrix and fulfills $P = SS^T$. If we know $P = AA^T$, the square-root factor S can be directly calculated from A by *QR decomposition*: $S = qr(A)^T$. If the matrix $A \in \mathbb{R}^{L\times N}$, then the computational complexity of a *QR decomposition* is $O(NL^2)$.

- *Cholesky factor updating*. If the original update of the covariance matrix is $P \pm uu^T$ and S is the Cholesky factor, then the rank 1 update of S is $S = cholupdate(S, u, \pm)$ where u is the update vector and ± means the positive (+) or negative (−) update. The positive update is usually numerical stable, but the negative update may destroy the positive definite property of S [Arasaratnam and Haykin, 2008; Seeger, 2004]. If u is a matrix, we can update each column of u one by one in a loop. For each column vector, the computational complexity is $O(L^2)$. This procedure can alternatively be implemented as $S = qr([S \;\; \pm u]^T)$ using *QR decomposition* without the loop updates.

- *Efficient least squares*. The least squares solution for the linear equation $Px = b$ can be solved efficiently using forward and back substitution if the Cholesky factor S is known and

---

Algorithm 11 SRCDIF for state estimation

```
• Initialization:
  $x_0 = E(x), S_{x0} = chol\left\{E\left((x - x_0)(x - x_0)^T\right)\right\}, S_v = \sqrt{R_v}$ and $S_n = \sqrt{R_n}$.
• For $k = 1, \cdots, \infty$:
    1. Generate sigma points for prediction:
       $x_{k-1}^{a_v} = \begin{bmatrix} x_{k-1} \\ \bar{v} \end{bmatrix}, \quad S_{k-1}^{a_v} = \begin{bmatrix} S_{x_{k-1}} & 0 \\ 0 & S_v \end{bmatrix}$    (5.27)
       $\mathcal{X}_{k-1}^{a_v} = \left[x_{k-1}^{a_v} \quad x_{k-1}^{a_v} + hS_{k-1}^{a_v} \quad x_{k-1}^{a_v} - hS_{k-1}^{a_v}\right]$    (5.28)
    2. Prediction equations:
       $\mathcal{X}_{k|k-1}^{x} = F(\mathcal{X}_{k-1}^{x}, \mathcal{X}_{k-1}^{v}, u_{k-1})$    (5.29)
       $\hat{x}_k = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{X}_{i,k|k-1}^{x}$    (5.30)
       $A = \sqrt{w_1^{(c_1)}}\left(\mathcal{X}_{1:L,k|k-1}^{x} - \mathcal{X}_{L+1:2L,k|k-1}^{x}\right)$    (5.31)
       $B = \sqrt{w_1^{(c_2)}}(\mathcal{X}_{1:L,k|k-1}^{x} + \mathcal{X}_{L+1:2L,k|k-1}^{x} - 2\mathcal{X}_{0,k|k-1}^{x})$    (5.32)
       $\hat{S}_{x_k} = qr\{[A \quad B]\}$    (5.33)
       $\hat{y}_k = \hat{S}_{x_k}^{-T}\left(\hat{S}_{x_k}^{-1}\hat{x}_k\right)$    (5.34)
       $\hat{S}_{y_k} = qr\left\{\hat{S}_{x_k}^{-1} I\right\}$    (5.35)
    3. Generate sigma points for measurement update:
       $\mathcal{X}_{k|k-1} = \left[\hat{x}_k \quad \hat{x}_k + h\hat{S}_{x_k} \quad \hat{x}_k - h\hat{S}_{x_k}\right]$    (5.36)
    4. Measurement update equations:
       $\mathcal{Z}_{k|k-1} = H(\mathcal{X}_{k|k-1})$    (5.37)
       $\hat{z}_k = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{Z}_{i,k|k-1}$    (5.38)
       $\hat{P}_{x_k z_k} = \sqrt{w_1^{c_1}}\hat{S}_{x_k}[\mathcal{Z}_{1:L,k|k-1} - \mathcal{Z}_{L+1:2L,k|k-1}]^T$    (5.39)
       $U = \hat{S}_{x_k}^{-T}\left(\hat{S}_{x_k}^{-1}\hat{P}_{x_k z_k}\right)S_n^{-T}$    (5.40)
       $y_k = \hat{y}_k + US_n^{-1}(z_k - \hat{z}_k + \hat{P}_{x_k z_k}^T \hat{y}_k)$    (5.41)
       $S_{y_k} = cholupdate\{\hat{S}_{y_k}, U, +\}$    (5.42)
```

---

satisfies $P = SS^T$. For example, we can solve the linear equation $Px = b$ by $x = S^{-T}(S^{-1}b)$. This operation has computational complexity $O(L^2)$.

The whole process is shown in Algorithm 11, where h is the scaling parameter, L is the dimension of the state, R_v and R_n are the process noise covariance and observation noise covariance respectively, $w_i^{(m)}$ and $w_i^{(c)}$ are weights calculated in Eq. (5.13), and I is the identity matrix.

In the prediction step, the *Cholesky factor* $\hat{S}_{x_k}$ is updated using *QR decomposition* on the weighted sigma points. This step replaces the $\hat{P}_k$ update in Eq. (5.12) and has the complexity $O(L^3)$. The information vector $\hat{y}_k = \hat{P}_k^{-1}\hat{x}_{x_k} = \hat{S}_{x_k}^{-T}\left(\hat{S}_{x_k}^{-1}\hat{x}_k\right)$ is derived using *efficient least squares* in Eq. (5.34). Because $\hat{S}_{x_k}$ is a square and triangular matrix, we can directly use back-substitution for solving $\hat{y}_k$ without the need for matrix inversion. The back substitution only requires $O(L^2)$. Next is the calculation of the square-root information matrix $\hat{S}_{y_k}$ in Eq. (5.35). This step requires a *QR decomposition* since $\hat{S}_{y_k}$ is a upper triangular matrix and $\hat{S}_{x_k}$ is a lower triangular matrix. $\hat{S}_{y_k}$ and $\hat{S}_{x_k}$ meet $\hat{S}_{y_k}^T\hat{S}_{y_k} = \hat{S}_{x_k}^{-T}\hat{S}_{x_k}^{-1}$. To avoid the inversion, here we use *efficient least squares* to solve $\hat{S}_{x_k}^{-1}$ as $\hat{S}_{x_k}^{-1}I$, where I is the identity matrix.

In the measurement update step, the updated information vector in Eq. (5.41) is derived from Eq. (5.21) as follows:

$$\begin{aligned} y_k &= \hat{y}_k + \hat{Y}_k\hat{P}_{x_kz_k}R_n^{-1}(z_k - \hat{z}_k + \hat{P}_{x_kz_k}^T\hat{y}_k) \\ &= \hat{y}_k + \hat{P}_k^{-1}\hat{P}_{x_kz_k}R_n^{-1}(z_k - \hat{z}_k + \hat{P}_{x_kz_k}^T\hat{y}_k) \\ &= \hat{y}_k + (\hat{S}_{x_k}\hat{S}_{x_k}^T)^{-1}\hat{P}_{x_kz_k}(S_nS_n^T)^{-1}(z_k - \hat{z}_k + \hat{P}_{x_kz_k}^T\hat{y}_k) \\ &= \hat{y}_k + \hat{S}_{x_k}^{-T}(\hat{S}_{x_k}^{-1}\hat{P}_{x_kz_k})S_n^{-T}S_n^{-1}(z_k - \hat{z}_k + \hat{P}_{x_kz_k}^T\hat{y}_k) \\ &= \hat{y}_k + US_n^{-1}(z_k - \hat{z}_k + \hat{P}_{x_kz_k}^T\hat{y}_k), \end{aligned} \tag{5.43}$$

where $U = \hat{S}_{x_k}^{-T}(\hat{S}_{x_k}^{-1}\hat{P}_{x_kz_k})S_n^{-T}$ as shown in Eq. (5.40). Since $\hat{S}_{x_k}$ and S_n are square and triangular matrices, y_k can be calulated using *efficient least squares* without the matrix inverse. The updated information matrix in Eq. (5.22) can be rewritten as:

$$\begin{aligned} Y_k &= \hat{Y}_k + \hat{Y}_k\hat{P}_{x_kz_k}R_n^{-1}(\hat{P}_{x_kz_k})^T(\hat{Y}_k)^T \\ &= \hat{Y}_k + \hat{Y}_k\hat{P}_{x_kz_k}S_n^{-T}S_n^{-1}(\hat{Y}_k\hat{P}_{x_kz_k})^T \\ &= \hat{Y}_k + \hat{Y}_k\hat{P}_{x_kz_k}S_n^{-T}(\hat{Y}_k\hat{P}_{x_kz_k}S_n^{-T})^T \\ &= \hat{Y}_k + UU^T. \end{aligned} \tag{5.44}$$

Because $\hat{S}_{y_k}$ is the Cholesky factor of the information matrix $\hat{Y}_k$, the updated Cholesky factor S_{y_k} of Y_k can be derived using the Cholesky update. If the observation dimension is M, the updated square-root information matrix S_{y_k} is calculated in Eq. (5.42) by applying an M-sequential Cholesky update to $\hat{S}_{y_k}$. The columns of matrix U are update vectors. This sequential Cholesky update only requires $O(L^2M)$.

5.3.2 SRCDIF for Multiple Sensor Fusion

In the case where information from multiple sensors is available, i.e., $N > 1$, we can fuse this using the Square-Root CDIF. For the i_{th} sensor, the information contribution for the information vector is

$$\phi_{i,k} = US_n^{-1}(z_k - \hat{z}_k + \hat{P}_{x_k z_k}^T \hat{y}_k) \tag{5.45}$$

where U is defined in Eq. (5.40). The information contribution for the square-root information matrix is

$$S_{i,\phi_k} = U. \tag{5.46}$$

The final estimated result is derived by:

$$y_k = \hat{y}_k + \sum_{i=0}^{N} \phi_{i,k} \tag{5.47}$$

$$S_{y_k} = cholupdate\{\hat{S}_{y_k}, [S_{1,\phi_k} \quad S_{2,\phi_k} \quad \cdots \quad S_{N,\phi_k}], +\}. \tag{5.48}$$

5.4 Square-Root Unscented Information Filter

In this section we consider the square-root implementation of the UIF. Because the UIF uses the *unscented transform* to calculate the sigma points, the architecture of the Square-Root unscented information filter (SRUIF) has few differences from the SRCDIF. As mentioned in Section 5.3, the main techniques behind the square-root form estimators are: *QR decomposition*, *Cholesky factor updating* and *efficient least squares*. We show how to use these in the SRUIF in the following.

The SRUIF is shown in Algorithm 12, where $\gamma = \sqrt{(\lambda + L)}$ in Eq. (5.50) is the composite scaling parameter, $\lambda = \alpha^2(L + \kappa) - L$, α and κ are scaling parameters that determine how far the sigma points spread from the mean value [Van der Merwe, 2004], L is the dimension of the state, R_v and R_n are process noise covariance and observation noise covariance respectively, $w_i^{(m)}$ and $w_i^{(c)}$ are weights calculated by $w_0^m = \frac{\lambda}{L+\lambda}$, $w_0^c = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta)$, $w_i^m = w_i^c = \frac{1}{2(L+\lambda)}$ $\quad i = 1, \cdots, 2L$, and $sign\{\}$ in Eq. (5.55) is the *signum* function.

We compare the SRUIF in Algorithm 12 to the SRCDIF in Algorithm 11. First, the SRUIF uses the *unscented transform* to calculate the sigma points in Eq. (5.50) and Eq. (5.58), where the scaling parameter becomes $\gamma = \sqrt{(\lambda + L)}$ and $\lambda = \alpha^2(L + \kappa) - L$. In contrast to only one scaling parameter h used in the SRCDIF, the SRUIF depends on three parameters λ, α and κ. Second, since the weight $w_0^{(c)}$ might be negative, we need an additional *cholupdate* to update the *Cholesky factor* $\hat{S}_{x_k}$ in Eq. (5.55), whereas the SRCDIF does not need this step since all weights used for the covariance update are positive. As we mentioned in Section 5.3.1, the negative update might destroy the positive definite property of the Cholesky factor, such that the SRCDIF is preferable to the SRUIF concerning the numerical stability. Finally, for multiple sensor fusion, the SRUIF is

---

Algorithm 12 SRUIF for state estimation

- Initialization:
  $x_0 = E(x)$, $S_{x0} = chol\left\{E\left((x - x_0)(x - x_0)^T\right)\right\}$, $S_v = \sqrt{R_v}$ and $S_n = \sqrt{R_n}$.
- For $k = 1, \cdots, \infty$:

1. Generate sigma points for prediction:

$$x_{k-1}^{a_v} = \begin{bmatrix} x_{k-1} \\ \overline{v} \end{bmatrix}, \quad S_{k-1}^{a_v} = \begin{bmatrix} S_{x_{k-1}} & 0 \\ 0 & S_v \end{bmatrix} \tag{5.49}$$

$$\mathcal{X}_{k-1}^{a_v} = \begin{bmatrix} x_{k-1}^{a_v} & x_{k-1}^{a_v} + \gamma S_{k-1}^{a_v} & x_{k-1}^{a_v} - \gamma S_{k-1}^{a_v} \end{bmatrix} \tag{5.50}$$

2. Prediction equations:

$$\mathcal{X}_{k|k-1}^{x} = F(\mathcal{X}_{k-1}^{x}, \mathcal{X}_{k-1}^{v}, u_{k-1}) \tag{5.51}$$

$$\hat{x}_k = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{X}_{i,k|k-1}^{x} \tag{5.52}$$

$$\hat{S}_{x_k} = qr\left\{ \sqrt{w_1^{(c)}} \left(\mathcal{X}_{1:2L,k|k-1}^{x} - \hat{x}_k\right) \right\} \tag{5.53}$$

$$C = \sqrt{w_0^{(c)}} \left(\mathcal{X}_0^{x} - \hat{x}_k\right) \tag{5.54}$$

$$\hat{S}_{x_k} = cholupdate\left\{\hat{S}_{x_k}, C, sign\{w_0^{(c)}\}\right\} \tag{5.55}$$

$$\hat{y}_k = \hat{S}_{x_k}^{-T} \left(\hat{S}_{x_k}^{-1} \hat{x}_k\right) \tag{5.56}$$

$$\hat{S}_{y_k} = qr\left\{\hat{S}_{x_k}^{-1} I\right\} \tag{5.57}$$

3. Generate sigma points for measurement update:

$$\mathcal{X}_{k|k-1} = \begin{bmatrix} \hat{x}_k & \hat{x}_k + \gamma \hat{S}_{x_k} & \hat{x}_k - \gamma \hat{S}_{x_k} \end{bmatrix} \tag{5.58}$$

4. Measurement update equations:

$$\mathcal{Z}_{k|k-1} = H(\mathcal{X}_{k|k-1}) \tag{5.59}$$

$$\hat{z}_k = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{Z}_{i,k|k-1} \tag{5.60}$$

$$\hat{P}_{x_k z_k} = \sum_{i=0}^{2L} w_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{x}_k^{-}][\mathcal{Z}_{i,k|k-1} - \hat{z}_k]^T \tag{5.61}$$

$$U = \hat{S}_{x_k}^{-T} \left(\hat{S}_{x_k}^{-1} \hat{P}_{x_k z_k}\right) S_n^{-T} \tag{5.62}$$

$$y_k = \hat{y}_k + U S_n^{-1} (z_k - \hat{z}_k + \hat{P}_{x_k z_k}^{T} \hat{y}_k) \tag{5.63}$$

$$S_{y_k} = cholupdate\{\hat{S}_{y_k}, U, +\} \tag{5.64}$$

---

equivalent to the SRCDIF in Eq. (5.47) and Eq. (5.48).

5.5 Experiment on Sensor Fusion

5.5.1 Space-Vehicle Tracking

To demonstrate the performance of the CDIF, UIF and their square-root forms SRCDIF and SRUIF, here we consider a classic space-vehicle reentry tracking problem, which was used in [Lee, 2008a; Julier and Uhlmann, 2004; Särkkä, 2008]. Two radars, which measure range and bearing, are used for tracking a high speed vehicle. The true trajectory of this vehicle is shown in Fig. 5.1.

The state space of the filter consists of the position (x_1 and x_2), the velocity (x_3 and x_4) and a parameter related to the aerodynamic force x_5. As described in [Julier and Uhlmann, 2004], the vehicle state dynamics for the discrete case are given by

$$\begin{aligned}
x_1(k+1) &= x_1(k) + \Delta t x_3(k) \\
x_2(k+1) &= x_2(k) + \Delta t x_4(k) \\
x_3(k+1) &= x_3(k) + \Delta t(D(k)x_3(k) + G(k)x_1(k)) + v_1 \\
x_4(k+1) &= x_4(k) + \Delta t(D(k)x_4(k) + G(k)x_2(k)) + v_2 \\
x_5(k+1) &= x_5(k) + \Delta t v_3,
\end{aligned} \tag{5.65}$$

where v_1, v_2 and v_3 are Gaussian process noises, $D(k)$ is the drag-related force, $G(k)$ is the gravity-related force, and $\Delta t = 0.1s$ is the sampling time. The force terms are given by

$$\begin{aligned}
D(k) &= \beta(k)V(k)exp\left\{\tfrac{R_0-R(k)}{H_0}\right\} \\
G(k) &= -\tfrac{Gm_0}{R^3(k)},
\end{aligned} \tag{5.66}$$

where $\beta(k) = \beta_0\ exp\{x_5(k)\}$, $R(k) = \sqrt{x_1^2(k) + x_2^2(k)}$ is the distance between the vehicle and the earth center, and $V(k) = \sqrt{x_3^2(k) + x_4^2(k)}$ is the vehicle's speed. The constants in Eq. (5.66) are defined as: $\beta_0 = -0.59783, H_0 = 13.406, Gm_0 = 3.9860 \times 10^5, R_0 = 6374$. The discrete process noise covariance in our simulation is defined by

$$R_v = diag(2.4064 \times 10^{-5}, 2.4064 \times 10^{-5}, 10^{-6}), \tag{5.67}$$

where *diag* means the diagonal matrix. The vehicle is tracked by two radars which are located at (x_s, y_s), where $s = 1, 2$, and the measurements model is

$$\begin{aligned}
r_s(k) &= \sqrt{(x_1(k) - x_s)^2 + (x_2(k) - y_s)^2} + e_{r,s} \\
\theta_s(k) &= tan^{-1}\left(\tfrac{x_2(k)-y_s}{x_1(k)-x_s}\right) + e_{\theta,s},
\end{aligned} \tag{5.68}$$

where $[e_{r,s}, e_{\theta,s}]^T \sim \mathcal{N}(0, R_{n,s})$ is the measurement noise. In the simulation, the radars are located

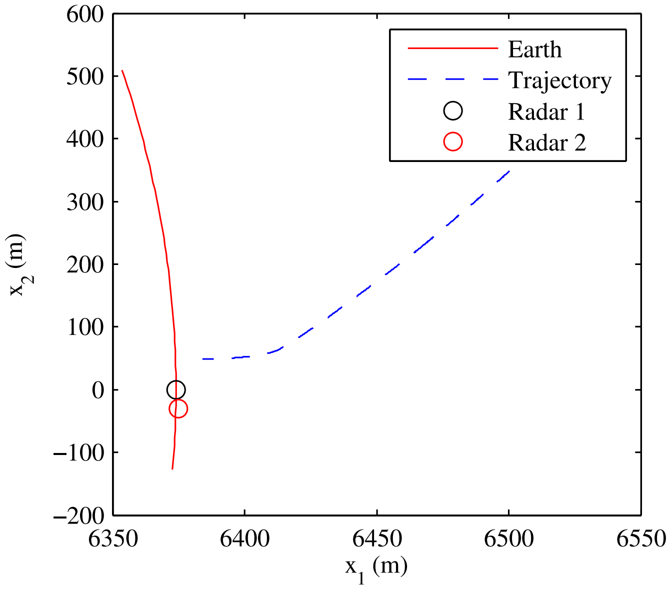


Figure 5.1: Earth surface, radar position and the real trajectory of the vehicle.

at $(x_1, y_1) = (6474, 0)$ and $(x_2, y_2) = (6475, -30)$, and their measurement noise variances are

$$\begin{aligned} R_{n,1} &= diag((1 \times 10^{-3})^2, (1.7 \times 10^{-4})^2) \\ R_{n,2} &= diag((2 \times 10^{-3})^2, (1.7 \times 10^{-4})^2). \end{aligned} \tag{5.69}$$

The initial true state and the covariance of the vehicle are given by

$$\begin{aligned} x_0 &= [6500.4, 349.14, -1.8093, -6.7967, 0.6932]^T \\ P_0 &= diag(10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 0), \end{aligned} \tag{5.70}$$

and the prior state and the covariance are given by

$$\begin{aligned} \hat{x}_0 &= [6500.4, 349.14, -1.8093, -6.7967, 0]^T \\ \hat{P}_0 &= diag(10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 1), \end{aligned} \tag{5.71}$$

which are the same as those used in [Julier and Uhlmann, 2004].

The time step Δt in Eq. (5.65) is set to $0.1s$, and measurements from both radars are received during each step, such that the observation frequency of both radars is $10Hz$.

The results of the simulation are derived from 100 Monte Carlo simulations, and summarized in Table 5.1, where UIFa, CDIFa, SRUIFa and SRCDIFa consider only the measurement from the first radar, and UIFb, CDIFb, SRUIFb and SRCDIFb consider measurements from both radars. The results indicate that, all filters have almost identical RMSE over time (at least to the fourth decimal place). This is to be expected, since the advantage of the square-root forms is that they have better numerical stability than the regular forms in the case of ill conditions, e.g., near perfect measurement (see our second experiment in Section 5.6). However, the square-root forms run slightly faster than the regular forms when two sensors are available, e.g., the SRCDIF has the lowest computational cost in this simulation. In addition, the filters can achieve more accurate results by fusing two sensors as we expected.

Table 5.1: Means (E) and standard deviations (STD) of RMSE values of the position and average run time in 100 Monte Carlo runs of the space-tracking problem

Number of Sensors	Method	E[RMSE]	STD[RMSE]	Average run time(s)
One	UIFa	0.0083	0.0007	4.0632
	CDIFa	0.0083	0.0007	3.5140
	SRUIFa	0.0083	0.0007	4.0667
	SRCDIFa	0.0083	0.0007	**3.5022**
Two	UIFb	0.0060	0.0005	5.3660
	CDIFb	0.0060	0.0005	4.6454
	SRUIFb	0.0060	0.0005	5.3364
	SRCDIFb	0.0060	0.0005	**4.5743**

5.5.2 Bearing-Only Tracking

In this section, we consider a nonlinear bearing-only tracking (BOT) problem using the UIF, CDIF, SRUIF and SRCDIF and compare their performances. The bearing-only tracking problem has become an important benchmark for different probability inference methods. By solving a BOT problem on a moving sensor platform, Bar-Shalom et al. [2001] analyzed the performance of the Taylor linearization in the EKF, Lin et al. [2002] compared the performance of the EKF, pseudo-measurement filter and particle filter, and Sadhu et al. [2004] proposed a new track-loss criterion for the comparison between the EKF and the square-root UKF. In addition, Hartikainen and Särkkä [2008] developed a toolbox which includes the comparison between the UKF, the EKF, and their smoothers by solving the BOT problem with static sensors.

Here we use the same system model as in [Hartikainen and Särkkä, 2008]. A moving target object is tracked by two static angular sensors. The discrete time update of the dynamic object on time step k is

$$\mathbf{x}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \mathbf{v}_{k-1} \tag{5.72}$$

where the system state is $\mathbf{x}_k = (x_k, y_k, \dot{x}_k, \dot{y}_k)^T$, which includes the target position (x_k, y_k) and velocity $(\dot{x}_k, \dot{y}_k)$. Δt is the time interval between time step k and $k-1$, which is set to $\Delta t = 0.01$ in the simulation. $\mathbf{v}_{k-1}$ is Gaussian noise with zero mean and the covariance is

$$\mathbf{R}_v = \begin{bmatrix} \frac{1}{3}\Delta t^3 & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{3}\Delta t^3 & 0 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 & 0 & \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & \Delta t \end{bmatrix} \beta \tag{5.73}$$

where β is the spectral density of the noise [Hartikainen and Särkkä, 2008] and set to $\beta = 0.1$ in our experiment. The target is tracked by sensors located at (x_s, y_s), where $s = 1, 2$ in the case of two sensors. The measurement model of the s_{th} sensor is defined as

$$\theta_s = tan^{-1}\left(\frac{y_k - y_s}{x_k - x_s}\right) + e_{\theta,s} \tag{5.74}$$

where $e_{\theta,s} \sim \mathcal{N}(0, R_{n,s})$ is the measurement noise of the s_{th} sensor. The sensors are located at $(x_1, y_1) = (-1, -2)$ and $(x_2, y_2) = (1, 1)$, and their measurement noise variances are $R_{n,1} = R_{n,2} = 0.05^2$. The initial prior state $\hat{\mathbf{x}}_0$ and the covariance $\hat{\mathbf{P}}_0$ are given by:

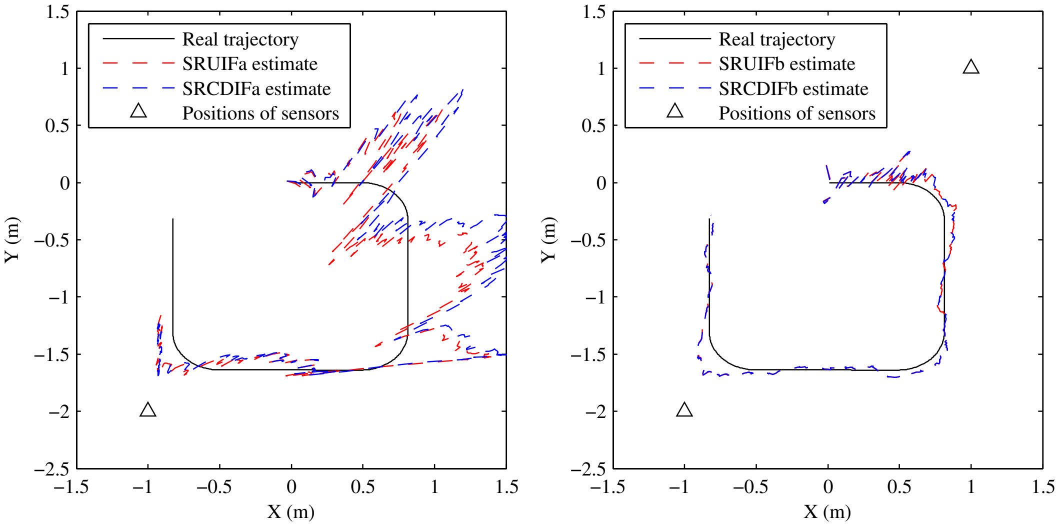
$$\hat{\mathbf{x}}_0 = [0,\ 0,\ 1,\ 0]^T \tag{5.75}$$

$$\hat{\mathbf{P}}_0 = diag(0.1, 0.1, 10, 10). \tag{5.76}$$

To achieve a curved trajectory the target has a randomized acceleration in our simulation [Hartikainen and Särkkä, 2008]. The estimated results from different filters are summarized in Table 5.2. It can be seen that the UIF and SRUIF have equal accuracy, as do the CDIF and the SRCDIF. Here we only show the comparison between the SRUIF and SRCDIF in Fig. 5.2. When only one sensor is available, the filters are hardly able to track the target and have very large RMSE errors as can be seen in Fig. 5.2a. Although the CDIFs still run faster than the UIFs in this simulation, the CDIFs have larger errors and covariances at the beginning of the trajectory. The filters achieve better results by fusing one more sensor which is shown in Fig. 5.2b.

Table 5.2: Means (E) and standard deviations (STD) of RMSE values of the position and average run time in 100 Monte Carlo runs of the bearing-only tracking

Number of Sensors	Method	E[RMSE]	STD[RMSE]	Average run time(s)
One	UIFa	0.6435	0.1364	0.5169
	CDIFa	0.6647	0.1638	**0.3512**
	SRUIFa	0.6435	0.1364	0.5042
	SRCDIFa	0.6647	0.1638	0.3596
Two	UIFb	0.1127	0.0277	0.8240
	CDIFb	0.1147	0.0294	0.5760
	SRUIFb	0.1127	0.0277	0.7952
	SRCDIFb	0.1147	0.0294	**0.5736**



(a) (b)

Figure 5.2: Comparison between the ground truth and the estimated trajectory for bearing-only tracking. (a): the comparison results when only one sensor is available. (b): the comparison results by fusing two sensors.

5.6 Experiment on Numerical Stability

To demonstrate the improved numerical characteristics of the proposed square-root filters, we consider a classic space-vehicle reentry tracking problem as shown in Section 5.5.1. In this experiment, The vehicle is tracked by a radar which is located at (x_s, y_s), and the measurement model is

$$\begin{aligned} r_s(k) &= \sqrt{(x_1(k) - x_s)^2 + (x_2(k) - y_s)^2} + e_{r,s}(k) \\ \theta_s(k) &= tan^{-1}\left(\frac{x_2(k)-y_s}{x_1(k)-x_s}\right) + e_{\theta,s}(k), \end{aligned} \tag{5.77}$$

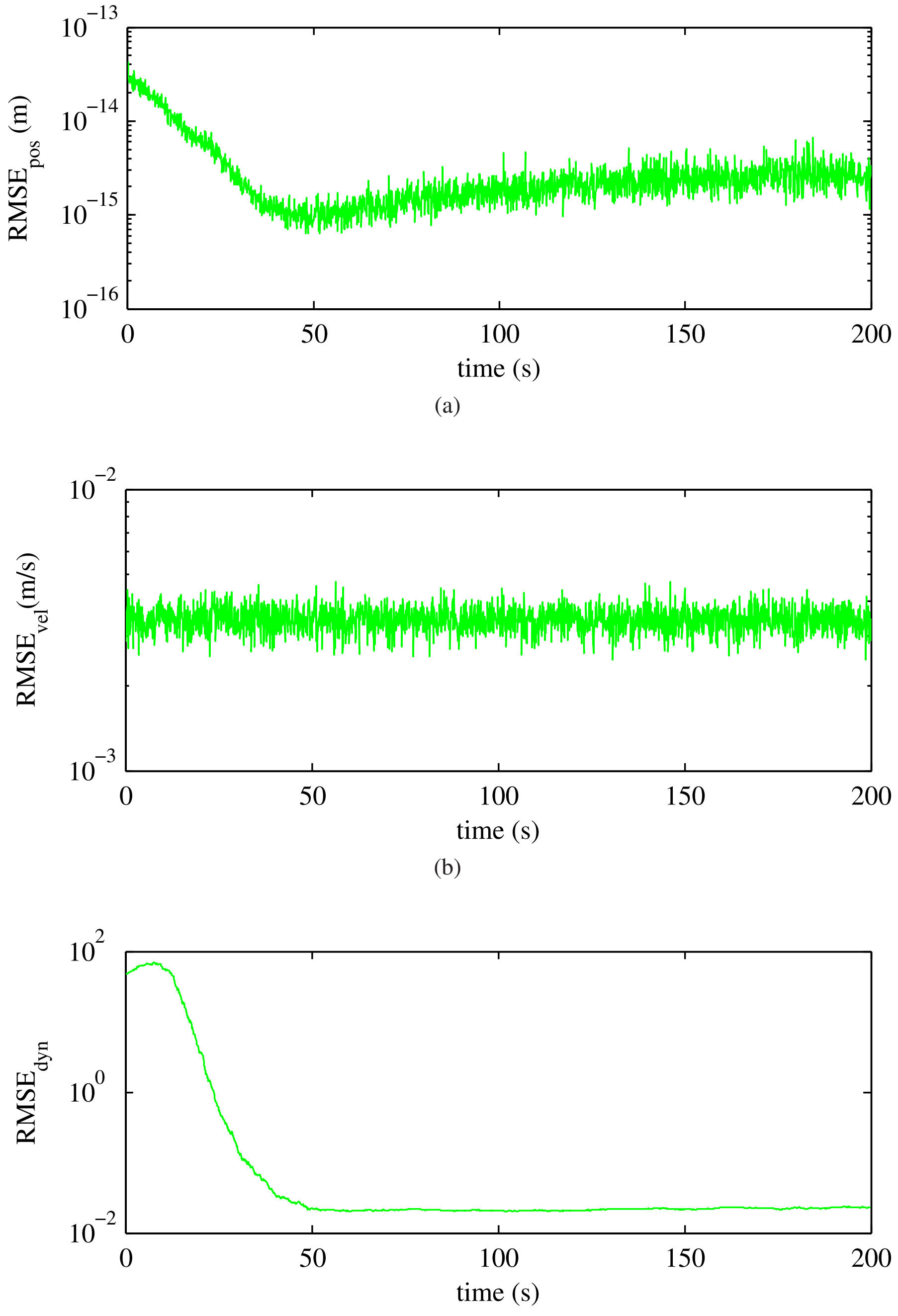
where $[e_{r,s}(k), e_{\theta,s}(k)]^T \sim \mathcal{N}(0, R_s(k))$ is the measurement noise. In the simulation, the radar is located at $(x_s, y_s) = (6474, 0)$ and the measurement noise variance is

$$R_s(k) = diag(e^2, e^2), \tag{5.78}$$

where e is a small number and meets $e \ll 1$, $1 + e \overset{r}{\neq} 1$ and $1 + e^2 \overset{r}{=} 1$. Here $\overset{r}{=}$ means equality due to rounding.

In addition, we set the scaling parameter $h = \sqrt{3}$ in CDIF and SRCDIF, $(\alpha = 0.001, \beta = 2, \kappa = -2)$ in UIF and SRUIF. The results of the simulation are derived from 100 Monte Carlo simulations.

It is well known that the cumulative effects of round-off errors eventually cause the estimated covariances to drift away from symmetry and positive definiteness [Howard and Jebara.,



(a)

(b)

(c)

Figure 5.3: Root mean square error (RMSE) in (a) position, (b) velocity and (c) aerodynamic force (x_5) estimations of the SRCDIF.

2005]. To illustrate the improved numerical characteristics of square root filters against the round off error, we set the measurement noise to be e^2 where e is the distance from 1.0 to the next larger double precision number as shown in Eq. (5.78) [Maybeck, 1979]. On the other hand, the near perfect measurement often causes the estimated covariance to be non-positive definite too [Arasaratnam and Haykin, 2008]. Since e is a very small number, it can be used to test the proposed square-root filters against the near perfect measurement as well. In the simulation, we set $e = 2^{-52}$ for a $32 bit$ system.

The simulation shows that only SRCDIF successfully run over all 100 Monte Carlo simulations as shown in Fig. 5.3, whereas CDIF, UIF and SRUIF failed to maintain the positive definiteness of the covariance matrix. Here the SRUIF can not enjoy the same numerical advantages as the SRCDIF, because the negative weight used in Eq. (5.55) destroyed the positive definiteness of resulting matrix.

5.7 Conclusion

In this chapter, a new central difference information filter (CDIF) algorithm for multiple sensor fusion and target tracking was presented. It is analogous to the UIF, but uses Stirling's interpolation instead of the unscented transform. Therefore, the CDIF only depends on one parameter (interval size) in contrast to three parameters which are required in the unscented transform. This makes the CDIF simpler, faster and easier tune than the UIF. Furthermore, we have introduced the square-root extensions of the CDIF and UIF. Although the proposed square-root forms have the same computational complexity as the regular ones, i.e., $O(L^3)$, the SRCDIF and SRUIF have better numerical properties, such as the improved numerical accuracy, double order precision and preservation of symmetry. In addition since the square-root of the covariance matrix is directly available, the SRCDIF and SRUIF can save computational costs in the step of sigma-point calculation. From the simulation, we conclude that the SRCDIF outperforms the others against the round off errors and near perfect measurements. The reason that SRUIF has failed in our simulation is that the negative weight used in the unscented transform causes non-positive eigenvalues of the resulting matrix. In the future, we plan to investigate their performances with different sensor network architectures [Lee, 2008b], and further improve the estimation accuracies, e.g., by combining the proposed filters with the adaptive consensus algorithm [Casbeer and Beard, 2009; Wang et al., 2010b].

Chapter 6

General Discussion

The main goal tackled in this thesis was how to improve the measurements in the Bayesian filtering framework for visual object tracking. We addressed three ways to improve the measurements, which are multiple visual cues, color invariant histograms and multiple sensor fusion. In this chapter, we will conclude the thesis, and evaluate the proposed methods and discuss the plans for future works.

6.1 Summary

Efficient and sufficient measurements of objects play a key role in visual object tracking. Visual object tracking is an important task for real world applications, e.g., intelligent surveillance, face recognition, smart room and intelligent robots. In order to track an object in image sequences recorded by cameras, A necessary prior step is to compute its appearance features, such as color, shape, texture, structure, etc. In the images, these appearance features are represented by visual cues which can be used as measurements of an object in a tracker. Because of geometric distortions and lighting changes, these visual cues usually are not always identical to the true appearance features. For instance, parallel lines in the real world are often not parallel in images, and the same object can have different colors under different lighting conditions. In addition, a single image usually provides only a partial information of objects, i.e., it may happen that only part of an object is visible. This partial information sometimes are not sufficient to distinguish the target object from other objects. Therefore, how to utilize efficient and sufficient visual cues as measurements for object tracking is an important and difficult problem. In order to obtain efficient and sufficient visual cues, we suggested to use multiple visual cues, color invariant features and multiple sensors, according to our main hypothesis mentioned in Chapter 1.

To illustrate how multiple visual cues can improve the measurements of object tracking, we selected lane tracking as a representative application, as shown in Chapter 3. In order to reveal the effectiveness of our method for combining multiple visual cues, we introduced two measurement model: one uses the multiple kernel density estimation to combine the color and intensity gradient (magnitude and direction) information, and the other one uses a simple combination of the color and edge (thresholded gradient magnitude) information. Clearly, the main differences between two measurement models are whether the gradient direction is used, whether the gradient magnitude is thresholded and whether the multiple kernel density is employed. As we expected, the experimental results show that the tracker using the former measurement can achieve better performance than the latter. Therefore, we have proved the idea that using multiple visual cues as measurements indeed can improve the performance of object tracking.

We given an evaluation framework of color invariant histograms to show how color invariant features can improve the measurements of object tracking in Chapter 4. Color histograms have been widely used for object tracking, due to their simplicity and effectiveness. However, they are susceptible to illumination changes. Therefore, it is necessary to investigate the color invariant properties of color histograms. The prior evaluations of color histograms mainly focus on applications concerning the classification. In our thesis, we evaluated the color invariant properties of color histograms for object tracking. The experiments was demonstrated on 13 open-access datasets, and 8 color histograms have been evaluated. As we expected, the results of our evaluations indicate that the trackers with color histograms that have color invariant properties, e.g., HSV, nRGB and Spherical, perform better than the ones that using color variant histograms, e.g., RGB. Consequently, our experiments prove that using color invariant features as measurements can improve the performance of object tracking.

We proposed new Bayes filters for multiple sensor fusion, which can improve the performance of object tracking by using the measurements from multiple sensors, as shown in Chapter 5. As discussed in Chapter 2, information filters have advantages for solving multiple sensor fusion problems, due to their simplicity in the measurement update step. In this thesis, we first introduced a new nonlinear information filter, which is called central difference information filter (CDIF). This CDIF has fewer predefined parameters than the unscented information filter (UIF) which has been introduced by [Lee, 2008a; Kim et al., 2008] recently. In addition, we introduced square-root extensions of the CDIF and UIF, which have better numerical characteristics. Finally, we tested the proposed information filters for solving sensor fusion problems in two simulations of object tracking: bearing-only tracking and reentry vehicle tracking. As we expected, the simulation results illustrate that the tracker can achieve better estimation accuracy by using multiple sensors. In addition, the square-root version of CDIF is more robust than the others against round-off errors and near perfect measurements for object tracking.

In this dissertation, we propose to improve the measurements in the Bayesian filtering framework for object tracking by using multiple visual cues, color invariant features and multiple sensors. This main hypothesis has been systematically tested and verified by a number of experiments, and we can conclude that the performance of object tracking indeed can be improved by using multiple visual cues, color invariant features and multiple sensors. Besides our achievements, there are also some limitations in our work, which will be analyzed in the following section.

6.2 Discussion

In the last section, we summarized the achievements in this dissertation. Now we would like to discuss the implications and limitations of our work.

For the fusion of multiple visual cues, we use a combination of color and shape together with multiple kernel density. We already showed that, for the application of lane tracking, our approach led to improvements. We want to point out that these improvements are not limited to the presented application. The possible extensions of our work are as follows: first, for objects which have a similar shape, i.e., parabolic or linear-parabolic shape, we can directly use our estimation method for tracking. For example, the human eyelid can be modeled by a parabolic shape [Liu et al., 2009], so we can use our method to detect and track eyelids, which is a very important process for iris recognition. Second, multiple kernel density estimation offers a natural way to estimate any shape from local visual cue elements, so it is also possible to use this method for estimating other shapes. Therefore, our idea can be employed in more applications, e.g., cell detection for medical care where cells can be modeled by circular shapes [Cao et al., 2009]. In contrast to our presented shape estimation, the color feature of the lane markers is very simple, so we did not need a complex method to estimate the color feature of lane markers. However, for some objects which require the color feature for identification, e.g., a colorful cup, the color feature can be very complex. In this case, we require a more elaborate method for color feature estimation from local visual cues. In addition, other features can be used for fusion, e.g., texture, which was shown to be important for tracking certain objects that usually have a specific pattern, e.g., zebras. Despite these possible extensions, we have shown the effectiveness of using multiple visual cues in the application of lane tracking, and our lane shape estimation method can be used for solving general shape estimation problems in other fields.

To evaluate color invariant features, we introduced a framework for evaluating color histograms. As far as we know, this is the first work in the literature to evaluate the color invariant properties of color histograms concerning the problem of object tracking! By analyzing the experimental results, we concluded that the tracker can achieve better result by using color histograms which have color invariant properties. In particular, we recommend three color invariant histograms for practical use, i.e., HSV, nRGB and Spherical color histograms. In addition, we used a diagonal-offset model, which is introduced by [Finlayson et al., 2005], to analyze the color invariant properties of color histograms. This model can also be used for analyzing the color invariant properties of other color invariant features, such as the features based on color differential geometry [Geusebroek et al., 2001], e.g., color-invariant and scale-invariant feature transform (C-SIFT) [Abdel-Hakim and Farag, 2006]. Since C-SIFT is invariant to both the scale and the color, the C-SIFT is usually more powerful than the color histograms. However, the C-SIFT requires more computational cost, which is a limitation for real time applications. In conclusion, our evaluation work can serve as recommendation to decide which color histograms should be used in practice for tracking.

Another important contribution of our work is the development of new Bayes filters to fuse multiple sensor information, i.e., central difference information filter (CDIF) and its square root form (SRCDIF), square root form of unscented information filter (SRUIF). Our new filters have the advantages compared to previous method for state estimation and sensor fusion, i.e., they require fewer predefined parameters and have better numerical stability. Information filters have a long history in the literature and many related real applications have been introduced [Fraser, 1967; Kaminski et al., 1971; Maybeck, 1979; Durrant-Whyte, 2001; Lee, 2008a]. Our new filters can also be utilized in all of these scenarios, e.g., multi-sensor data fusion using stereo camera, laser range finder and GPS receiver for vehicle localization [Wei et al., 2011]. Besides the sensor fusion, the proposed information filters can also be used for solving robotic simultaneous localization and mapping (SLAM) problems. For instance, Thrun et al. [2004] presented a SLAM framework using sparse extended information filter. Therefore, it is possible to use our new nonlinear information filters instead of the extended information filter. However, how to sparsify our new nonlinear information filters requires further analysis. Finally, we can see that the proposed new nonlinear information filters can be extensively used for solving state estimation and sensor fusion problems. A limitation of our proposed work is that we only show the performance of proposed filters by means of simulations. It will be interesting to demonstrate the proposed filters on the real sensor networks.

In summary, although we are aware of several limitations of our work as discussed above, we have successfully improved the measurements for object tracking by combining multiple visual cues, using color invariant histograms and fusing multiple sensors. Furthermore, we show that the new ideas inside of our work can be applied to other scenarios as well.

6.3 Future Works

Regarding the works we have done in this thesis, there are some open issues which can be studied in future. The possible extensions are discussed as follows.

For the lane shape estimation part, a very simple multiple lane model was used where the relations between multiple lanes are defined by the simple distance constraint. This simple model introduces a high dimensional state, i.e., the dimension of the state is three times of lane numbers. A better multiple lane model can be defined using the distance constraint and perspective analysis [Nieto et al., 2008]. In this way, the dimension of the state can be reduced. On the other hand, the inverse perspective mapping (IPM) images are used in our algorithm which offer a top-view of the road scene. The IPM algorithm has the assumption of the flat ground plane, so a constant pitch angle is used to calculate IPM images. However, the real situation in the urban environment is that the road plane is not flat. To solve this problem an online pitch angle estimation is necessary [Danescu and Nedevschi, 2009], which can guarantee the lanes are parallel in IPM images. In addition, the constant dividing line in the linear-parabolic model is the other limitation. It is better to have an adaptive solution.

For the color invariant histogram part, the evaluation is limited by the distance metric, the number of color spaces and databases. In future, we would like to investigate more color spaces and distance metrics on databases with various illumination conditions. In our experiment, the spatial information is included by using an isotropic kernel mask to the target region. It is possible to further investigate the performance of an anisotropic kernel [Khan et al., 2009] or multiple kernel solutions [Teuliere et al., 2009].

In the sensor fusion part, two simulations are implemented to demonstrate the performance of the proposed sigma point information filters for nonlinear estimation and sensor fusion. In future, we would like to run this framework on the real camera network and evaluate the performance of the proposed algorithms. This idea will be useful for real world applications, such as the intelligent surveillance using multiple cameras [Kulkarni et al., 2005].

Appendix A

Cholesky Factorization for Drawing Error Ellipses

This chapter explains how we draw the 2D error ellipse in Fig. 2.1 using Cholesky factorization.

We first define the error ellipse as:

$$\Gamma(\mu, \Sigma) = \left\{X : \ (X - \mu)^T \Sigma^{-1} (X - \mu) = 1\right\}, \tag{A.1}$$

where the covariance matrix Σ is a 2×2 positive definite matrix and the mean μ is a 2D state vector, $\Gamma(\mu, \Sigma)$ represents a collection of the sampling points X in the error ellipse.

Now we can draw $\Gamma(\mu, \Sigma)$ using a known collection of the sampling points Y from a unit ellipse $\Gamma(\mu_0, \Sigma_0)$. For a 2D state error ellipse, μ_0 and Σ_0 are defined by

$$\mu_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \ \Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{A.2}$$

then the ellipse can be represented by

$$X = LY + \mu, \tag{A.3}$$

where L is the lower triangle Cholesky factor of covariance matrix Σ, such that $\Sigma = LL^T$. The proof is as follows:

$$\begin{aligned} (X - \mu)^T \Sigma^{-1} (X - \mu) &= (LY + \mu - \mu)^T \Sigma^{-1} (LY + \mu - \mu) \\ &= (LY)^T (LL^T)^{-1} (LY) \\ &= Y^T L^T (L^T)^{-1} (L)^{-1} LY \\ &= Y^T Y \\ &= 1 \end{aligned} \tag{A.4}$$

Appendix B

Inverse Perspective Mapping

As we mentioned in Chapter 3, the inverse perspective mapping (IPM) is a method that transforms the original image to a top-view image. The IPM makes some assumptions, such as that intrinsic and extrinsic parameters of the camera are known and the ground plane is flat. In this chapter, we will give details about how to computer the IPM image.

The relationships between the world coordinates $X_wY_wZ_w$, IPM image coordinates X_gY_g, camera coordinates $X_cY_cZ_c$, and original image coordinates X_iY_i are shown in Fig. B.1. To derive the IPM image, we need first calibrate the camera, then get the homogeneous transformation matrices between the X_gY_g and X_iY_i. The detail of each step is presented as following:

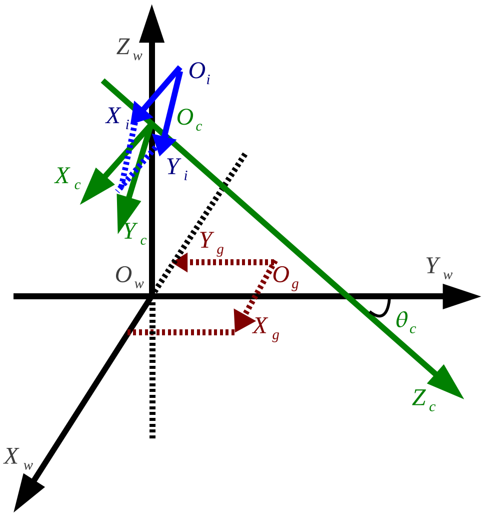


Figure B.1: The relationships between the world coordinates (black), inverse perspective mapping (IPM) image coordinates (red), camera coordinates (green) and image coordinates (blue).

- **Camera calibration**. The camera can be calibrated using the Matlab calibration toolbox

[Bouguet, 2010] or OpenCV library [Willowgarage, 2011]. One necessary step is we need one image that the checkerboard is on the ground $X_gO_gY_g$, so we can measure the pitch angle θ_c and the hight of the camera h from this special image. After the camera is calibrated, the intrinsic matrix M_{c2i} and extrinsic matrix M_{w2c} are derived by:

$$M_{c2i} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

$$M_{w2c} = \begin{bmatrix} R_{w2c} & R_{w2c}T_{w2c} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{B.2}$$

where f_x and f_y are the focal length expressed in units of horizontal and vertical pixels respectively, c_x and y_x are the coordinates of the principle point. R_{w2c} and T_{w2c} are the rotation matrix and translation vector respectively, which can be derived from the pitch angle θ_c and the hight of the camera h:

$$R_{w2c} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_c + \pi/2) & -sin(\theta_c + \pi/2) \\ 0 & sin(\theta_c + \pi/2) & cos(\theta_c + \pi/2) \end{bmatrix}, \; T_{w2c} = \begin{bmatrix} 0 \\ 0 \\ -h \end{bmatrix}. \tag{B.3}$$

Note that here the rotation angle is positive when the coordinate rotation is clockwise viewed from the anti-rotation-axis-direction.

- **From the ground plane coordinates** X_gY_g **to the world coordinates** $X_wY_wZ_w$. Assuming the coordinates of the original point O_g in the world coordinates $X_wY_wZ_w$ are $[x_{og}, y_{og}]^T$, we can calculate the rotation matrix R_{g2w} and translation vector T_{g2w} as follows:

$$R_{g2w} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\pi) & -sin(\pi) \\ 0 & sin(\pi) & cos(\pi) \end{bmatrix}, \; T_{g2w} = \begin{bmatrix} -x_{og} \\ -y_{og} \\ 0 \end{bmatrix}. \tag{B.4}$$

The transformation matrix M_{g2w} between the ground plane coordinates X_gY_g and the world coordinates $X_wY_wZ_w$ is

$$M_{g2w} = \begin{bmatrix} R_{g2w} & R_{g2w}T_{g2w} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{B.5}$$

- **Mapping**. Given one point P_g on the ground plane, we can calculate its image coordinate vector $P_i = [x_i, y_i, w, 1]^T$ in the homogeneous space by

$$P_i = M_{c2i}M_{w2c}M_{g2w}P_g. \tag{B.6}$$

The final two dimensional image coordinate vector $[u, v]^T$ are derived by $u = x_i/w$ and $v = y_i/w$.

- **Image distortion**. For many cameras we need take the image distortion into out account, such as wide angle cameras. In this case, we can not directly use the Eq. (B.6). The distortion must be removed. For a point $P_c = [x_c, y_c, z_c, 1]^T = M_{w2c}M_{g2w}P_g$ in the camera coordinates, the normalized image coordinate vector $P_n = [x_n, y_n]^T$ is calculated by $x_n = x_c/z_c$ and $y_n = y_c/z_c$. The distorted image coordinate vector $P_d = [x_d, y_d]^T$ is derived as:

$$P_d = \begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} r_d x_n + 2k_c(3)x_n y_n + k_c(4)(r^2 + 2x_n x_n) \\ r_d y_n + 2k_c(4)x_n y_n + k_c(3)(r^2 + 2y_n y_n) \end{bmatrix}, \tag{B.7}$$

  where k_c is a five dimensional vector containing both radial and tangential distortion coefficients, $r^2 = x_n^2 + y_n^2$ and $r_d = 1 + k_c(1)r^2 + k_c(2)r^4 + k_c(5)r^6$. Once the distortion applied, the distorted image pixel coordinate vector can be calculated by $P_i = M_{c2i}P_d$.

The IPM algorithm in fact is a mapping procedure between the ground plane and the image pixels, so a fast look-up table can be used to speed up this algorithm for real-time applications [Bergener and Bruckho, 1999].

Bibliography

Abdel-Hakim, A. and Farag, A. (2006). Csift: A sift descriptor with color invariant characteristics. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1978 – 1983. 94

Abramov, A., Aksoy, E. E., Drr, J., Wörgötter, F., and Dellen, B. (2010). 3d semantic representation of actions from efficient stereo-image-sequence segmentation on gpus. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission*. 56

Aghajan, H. and Cavallaro, A., editors (2009). *Multi-Camera Networks Principles and Applications*. Academic Press. 12, 15

Anderson, B. D. O. and Moore, J. B. (1979). *Optimal Filtering*. Eaglewood Cliffs, NJ: Prentice-Hall. 77, 78, 79

Arasaratnam, I. and Haykin, S. (2008). Square-root quadrature kalman filtering. *Signal Processing, IEEE Transactions on*, 56(6):2589 –2593. 16, 79, 90

Arasaratnam, I. and Haykin, S. (2009). Cubature kalman filters. *Automatic Control, IEEE Transactions on*, 54(6):1254 –1269. 74

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188. 19

Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation*. Wiley Interscience. 86

Baseski, E., Baunegaard With Jensen, L., Pugeault, N., Pilz, F., Pauwels, K., Van Hulle, M., Wörgötter, F., and Krüger, N. (2009). Road interpretation for driver assistance based on an early cognitive vision system. In *International Conference on Computer Vision Theory and Applications (VISSAP '09)*, pages 5–8. 53

Bergener, T. and Bruckho, C. (1999). Compensation of non-linear distortions in inverse-perspective mappings. Technical report, Institut für Neuroinformatik, Ruhr-Universität Bochum. 43, 101

Bertozzi, M. and Broggi, A. (1998). Gold: A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, 7(1):62–81. 42

Birchfield, S. (1998). Elliptical head tracking using intensity gradients and color histograms. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 232 –237. 14

Bouguet, J.-Y. (2010). http://www.vision.caltech.edu/bouguetj/calib_doc/. 100

Burghouts, G. J. and Geusebroek, J.-M. (2009). Performance evaluation of local colour invariants. *Computer Vision and Image Understanding*, 113:48–62. 15, 64

Cannons, K. (2008). A review of visual tracking. Technical report, York University. 59

Cao, G., Zhong, C., Li, L., and Dong, J. (2009). Detection of red blood cell in urine micrograph. In *Bioinformatics and Biomedical Engineering , 2009. ICBBE 2009. 3rd International Conference on*, pages 1 –4. 93

Casbeer, D. and Beard, R. (2009). Distributed information filtering using consensus filters. In *American Control Conference, 2009. ACC '09.*, pages 1882 –1887. 90

Chong, C. Y. (1979). Hierarchical estimation. In *in Proc. MIT/ONR Workshop on C3, Monterey, CA*. 78

Collins, R., Lipton, A., Fujiyoshi, H., and Kanade, T. (2001). Algorithms for cooperative multi-sensor surveillance. *Proceedings of the IEEE*, 89(10):1456 –1477. 15

Comaniciu, D. and Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603 –619. 66

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142 –149 vol.2. 14

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564 – 577. 59, 60, 64, 66

Dahyot, R. (2009). Statistical hough transform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(8):1502–1509. 12, 13, 16, 44, 45, 49

Danescu, R. and Nedevschi, S. (2009). Probabilistic lane tracking in difficult road scenarios using stereovision. *IEEE Transactions on Intelligent Transportation Systems*, 10(2):272–282. 42, 44, 57, 95

Duffner, S., Odobez, J.-M., and Ricci, E. (2009). Dynamic partitioned sampling for tracking with discriminative features. In *Proceedings of the British Maschine Vision Conference*. 39, 52

Durrant-Whyte, H. (2001). Multi sensor data fusion. Technical report, Australian Centre for Field Robotics, The University of Sydney NSW 2006. 15, 32, 94

Finlayson, G., Hordley, S., and Xu, R. (2005). Convex programming colour constancy with a diagonal-offset model. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III – 948–51. 60, 61, 93

Fraser, D. C. (1967). *A new technique for the optimal smoothing of data*. PhD thesis, Massachusetts Institute of Technology. 31, 94

Funt, B. and Finlayson, G. (1995). Color constant color indexing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(5):522 –529. 14

Geusebroek, J.-M., van den Boomgaard, R., Smeulders, A., and Geerts, H. (2001). Color invariance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(12):1338 –1350. 60, 64, 94

Gevers, T. and Smeulders, A. W. (1999). Color-based object recognition. *Pattern Recognition*, 32:453464. 14, 60, 63

Hartikainen, J. and Särkkä, S. (2008). Optimal filtering with kalman filters and smoothers a manual for matlab toolbox ekf/ukf. 86, 87

Howard, A. and Jebara., T. (2005). Square root propagation. Technical report, Columbia University. 88

Isard, M. and Blake, A. (1998). Condensationconditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28. 10.1023/A:1008078328650. 12

Ito, K. and Xiong, K. (2000). Gaussian filters for nonlinear filtering problems. *Automatic Control, IEEE Transactions on*, 45(5):910 –927. 26

Julier, S. and Uhlmann, J. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401 – 422. 84, 85

Julier, S., Uhlmann, J., and Durrant-Whyte, H. (1995). A new approach for filtering nonlinear systems. In *American Control Conference*. 24

Jung, C. and Kelber, C. (2005). An improved linear-parabolic model for lane following and curve detection. In *Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on*, pages 131 – 138. 16

Jung, C. R. and Kelber, C. R. (2004). A robust linear-parabolic model for lane following. In *Proc. 17th Brazilian Symposium on Computer Graphics and Image Processing*, pages 72–79. 16, 42, 43, 44, 47

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82 (1):3545. 19, 21

Kaminski, P., Bryson, A., J., and Schmidt, S. (1971). Discrete square root filtering: A survey of current techniques. *Automatic Control, IEEE Transactions on*, 16(6):727 – 736. 16, 94

Khan, Z., Gu, I.-H., and Backhouse, A. (2009). Joint particle filters and multi-mode anisotropic mean shift for robust tracking of video objects with partitioned areas. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 4077 –4080. 14, 60, 95

Kim, Y., Lee, J., Do, H., Kim, B., Tanikawa, T., Ohba, K., Lee, G., and Yun, S. (2008). Unscented information filtering method for reducing multiple sensor registration error. In *Multisensor Fusion and Integration for Intelligent Systems, IEEE International Conference on*, pages 326 –331. 32, 74, 78, 92

Kim, Z. (2008). Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):16–26. 41

Kitagawa, G. (1996). Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25. 35

Kulkarni, P., Ganesan, D., Shenoy, P., and Lu, Q. (2005). Senseye: a multi-tier camera sensor network. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA ’05, pages 229–238, New York, NY, USA. ACM. 95

Lee, D.-J. (2008a). Nonlinear estimation and multiple sensor fusion using unscented information filtering. *Signal Processing Letters, IEEE*, 15:861 –864. 16, 17, 32, 73, 74, 78, 84, 92, 94

Lee, D.-J. (2008b). Unscented information filtering for distributed estimation and multiple sensor fusion. In *AIAA Guidance, Navigation, and Control Conference*. 73, 90

Lim, K. H., Seng, K. P., Ang, L.-M., and Chin, S. W. (2009). Lane detection and kalman-based linear-parabolic lane tracking. In *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC ’09. International Conference on*, volume 2, pages 351 –354. 16, 42, 43, 44, 47

Lin, X., Kirubarajan, T., Bar-Shalom, Y., and Maskell, S. (2002). Comparison of ekf, pseudomeasurement, and particle filters for a bearing-only target tracking problem. In *in Proc. SPIE Conference of Signal and Data Processing of Small Targets*. 86

Liu, G., Wörgötter, F., and Markelić, I. (2010). Combining statistical hough transform and particle filter for robust lane detection and tracking. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 993 –997. 16, 52

Liu, G., Wörgötter, F., and Markelić, I. (2011a). Lane shape estimation using a partitioned particle filter for autonomous driving. In *Robotics and Automation, 2011. ICRA 2011. IEEE International Conference on*. 16, 52

Liu, G., Wörgötter, F., and Markelić, I. (2011b). Nonlinear estimation using central difference information filter. In *IEEE Workshop on Statistical Signal Processing*, pages 593–596. 17, 74

Liu, G., Wörgötter, F., and Markelić, I. (2012). The square-root unscented information filter for state estimation and sensor fusion. In *International Conference on Sensor Networks (SENSORNETS)*. 17

Liu, X., Song, Q., and Li, P. (2009). A parabolic detection algorithm based on kernel density estimation. In *5th International Conference on Intelligent Computing*, pages 405–412. 47, 49, 57, 93

Louw, M. S. (2004). Partitioned particle filtering for target tracking in video sequences. Master's thesis, University of cape town. 39

MacCormick, J. and Blake, A. (1999). A probabilistic exclusion principle for tracking multiple objects. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 572 –578 vol.1. 38, 39

MacCormick, J. and Blake, A. (2000). A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, 39(1):57–71. 16, 43, 49

MacCormick, J. and Isard, M. (2000). Partitioned sampling, articulated objects, and interface-quality hand tracking. In *European Conference on Computer Vision*. 52

Markelic, I., Kjaer-Nielsen, A., Pauwels, K., Jensen, L. B. W., Chumerin, N., Vidugiriene, A., Tamosiunaite, M., Hulle, M. V., Kruerger, N., Rotter, A., and Woergoetter, F. (2011). The driving school system: Learning basic driving skills from a teacher in a real car. *Intelligent Transportation Systems, IEEE Transactions on*, PP(99):1 –12. 53

Martinkauppi, B., Soriano, M., Huovinen, S., and Laaksonen, M. (2002). Face video database. In *Proc. First European Conference on Color in Graphics, Imaging and Vision (CGIV'2002)*. 67, 68

Maybeck, P. S. (1979). *Stochastic models, estimation, and control*. Academic Press. 19, 32, 74, 90, 94

McCall, J. C. and Trivedi, M. M. (2006). Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):20–37. 41, 42

Nieto, M., Salgado, L., Jaureguizar, F., and Arrospide, J. (2008). Robust multiple lane road modeling based on perspective analysis. In *ICIP*, pages 2396–2399. 57, 94

Nummiaro, K., Koller-meier, E., and Gool, L. V. (2002). A color-based particle filter. In *In First International Workshop on Generative Model Based Vision. In conjunction with ECCVO2, pages 53-60*, pages 53–60. 67

Nummiaro, K., Koller-Meier, E., and Gool, L. V. (2003). An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99 – 110. 67

Pérez, P., Hue, C., Vermaak, J., and Gangnet, M. (2002). Color-based probabilistic tracking. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Computer Vision ECCV 2002*, volume 2350 of *Lecture Notes in Computer Science*, pages 661–675. Springer Berlin / Heidelberg. 10.1007/3-540-47969-4_44. 14

Porikli, F. (2005). Integral histogram: a fast way to extract histograms in cartesian spaces. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 829 – 836 vol. 1. 59, 65, 66

Potter, J. and Stern., R. (1963). Statistical filtering of space navigation measurements. In *In AIAA Guidance Contr. Conference*. 74

Ross, D., Lim, J., Lin, R.-S., and Yang., M.-H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77:125–141. 68

Sadhu, S., Srinivasan, M., Mondal, S., and Ghoshal, T. (2004). Bearing only tracking using square root sigma point kalman filter. In *India Annual Conference, 2004. Proceedings of the IEEE INDICON 2004. First*, pages 66 – 69. 86

Särkkä, S. (2008). Unscented rauch–tung–striebel smoother. *Automatic Control, IEEE Transactions on*, 53(3):845 –849. 84

Seeger, M. (2004). Low rank updates for the cholesky decomposition. Technical report, EPFL. 79

Smith, A. R. (1978). Color gamut transform pairs. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, SIGGRAPH ’78, pages 12–19, New York, NY, USA. ACM. 62

Smith, K. and Gatica-perez, D. (2004). Order matters: a distributed sampling method for multi-object tracking. In *In Proc. BMVC*. 52

Southall, B. and Taylor, C. (2001). Stochastic road shape estimation. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 205 –212 vol.1. 44, 49

Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7:11–32. 14

Teuliere, C., Marchand, E., and Eck, L. (2009). A combination of particle filtering and deterministic approaches for multiple kernel tracking. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3948 –3954. 60, 95

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press. 20, 22, 25, 31, 35, 51

Thrun, S., Liu, Y., , Koller, D., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H. (2004). Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23:693. 94

van de Sande, K., Gevers, T., and Snoek, C. (2010). Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1582 –1596. 14, 15, 17, 60, 61, 62, 63

van de Weijer, J., Gevers, T., and Bagdanov, A. (2006). Boosting color saliency in image feature detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(1):150 –156. 14, 64

Van der Merwe, R. (2004). *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, OGI School of Science & Engineering, Oregon Health & Science University,. 25, 26, 29, 73, 74, 75, 76, 77, 82

von Kries, J. (1970). Influence of adaptation on the effects produced by luminous stimuli. In MacAdam, D., editor, *Sources of Color Vision*. MIT Press. 60, 61

Wang, L., Wang, N., and Zhu, H. (2010a). Consensus based distributed unscented information filtering for air mobile sensor networks. In *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, volume 2, pages 492 –495. 74

Wang, L., Zhang, Q., Zhu, H., and Shen, L. (2010b). Adaptive consensus fusion estimation for msn with communication delays and switching network topologies. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 2087 –2092. 73, 90

Wei, L., Cappelle, C., and Ruichek, Y. (2011). Unscented information filter based multi-sensor data fusion using stereo camera, laser range finder and gps receiver for vehicle localization. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1923 –1928. 94

Willowgarage (2011). http://opencv.willowgarage.com/wiki/. 100

Wu, Y., Hu, D., Wu, M., and Hu, X. (2005). Unscented kalman filtering for additive noise case: augmented versus nonaugmented. *Signal Processing Letters, IEEE*, 12(5):357 – 360. 26

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13. 11, 12

Zhou, Y., Xu, R., Hu, X., and Ye, Q. (2006). A robust lane detection and tracking method based on computer vision. *Measurement Science and Technology*, 17(4):736–745. 43, 50

Zhu, J., Zheng, N., Yuan, Z., Zhang, Q., Zhang, X., and He, Y. (2009). A slam algorithm based on the central difference kalman filter. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 123 –128. 74, 75

Zimmermann, K. (2008). *Fast Learnable Methods for Object Tracking*. PhD thesis, Czech Technical University. 11

Curriculum Vitae

Personal information

Name	Guoliang Liu
Birth date	November 1983
Birth place	Shandong (China)
Nationality	China

Academic history

2001-2005	B.Sc. in Physics, Shandong Normal University, Jinan, China
2005-2008	M.Eng. in Control Science and Engineering, National University of Defense Technology, Changsha, China
2008-2012	Research Assistant, Bernstein Center for Computational Neuroscience, Georg-August-Universität Göttingen, Germany