Visual Tracking with Filtering Algorithms

Botond A. Bócsi Lehel Csató
Department of Mathematics and Computer Science
Babeş-Bolyai University
Kogalniceanu str. 1, RO-400084, Cluj-Napoca
Email: **bocsiboti@gmail.com lehel.csato@cs.ubbcluj.ro**

Abstract

We present a comparative study of object tracking algorithms using filtering methods. We detail the underlying model assumptions the different algorithms use, measure their operation performance, and compare them in real environmental settings. The comparison is based on several different criteria, including both the computational time and the performance of the tracker.

We study a restricted family of methods, called filters. We compare the Kalman filter*,* unscented Kalman filter *and the* particle filtering *methods. Based on real-world settings, some conclusions are drawn about the usability of the algorithms. We outline the conditions when a given algorithm becomes efficient.*

Keywords: object tracking, unscented Kalman filter, particle filter

1 Introduction

The use of efficient object tracking algorithms in video processing became an essential task in many fields of the modern life. Almost all robots, security cameras, industrial camera systems need to locate and track specific objects. Many tracking techniques are available, each of them with its strong and weak points respectively. An obvious deciding factor is the performance of the hardware equipment that lies behind the cameras: powerful equipment can "afford" a more sophisticated algorithm and conversely: when we have limited computational power at our disposal, we have to balance the parameters so that the algorithm eventually terminates in a convenient time. Aside from the above issues, when selecting a method we must consider the conditions where the tracking will be done, like luminance, *etc.*Ideally we want to look at the object we want to track, the specifics of the motions, and the "background" environment. Based on the considerations above, we aim to pick the most adequate algorithm.

In this article we study a particular class of methods: the *filter-based methods*. The name *filter* originates from physics: it is a device that removes unwanted components *e.g.*from electronic signals. It is used in estimation theory since the 1940s [5], with the meaning of the removal of the noise from a dynamical system perspective. Not filter–based but still relevant category of visual tracking algorithms was introduced by Isard and Blake [7, 8], and called *condensation algorithm*. This method does is significantly more coplex than filtering-based methods since it necessarily needs a contour based representation, which in turn require rather complex image processing techniques.

We mention that there is a significant difference between localizing, or identifying an object and respectively tracking it: identifying an object in a real-world environment is mostly a question of ontology and object recognition, while tracking is the detection of *changes* relevant to the object we track. We emphasize that in this paper we do not address the problem of object recognition, rather we focus on the tracking of already identified objects.[1]

The structure of the paper is the following: next we give a short introduction to dynamic systems, followed by a detailed description of the mathematical models used in tracking algorithms and we present the Kalman filter, the unscented Kalman filter, and the particle filter respectively. We then present the experimental results and conclude the paper with presenting the strong and weak points of the presented algorithms.

2 Tracking algorithms

In this section we give a short introduction to *dynamic systems* [6]. A dynamic system is an assemble of elements which attributes of interest are changing in time. In each time the state of the systems depends on the previous states; if it depends on the n previous states, it can be described

[1] If one wants to track 2D blobs through progressive image, a blob based tracking is advised, well explained by Collins [1].

by an n ordered differential equation. Very often the evolution of the state in time cannot be told explicitly. The main difficulty in the field of dynamic systems is that they are very hard predictable in time, a small change in the initial conditions could have immense effects on the future states. Besides that, it is often difficult - or impossible - to measure explicitly the state of the system and only indirect measurement can be done.

Sometimes one does not have an arbitrary precise device and cannot measure arbitrary precise the output of the system, so the measurements are perturbated with some noise (measurement noise). Noise can also appear if one does not know accurate enough the exact lows which govern the system (operation noise).

Object tracking – as viewed here – is the prediction of latent states in a dynamic system: the position of the object is the latent state of the system and the image received from the camera is the measured observational data. We next describe three mathematical models that are used in dynamical systems and here are used for object tracking.

2.1 Kalman Filter

The Kalman filter is the optimal filter that estimates the state of a linear dynamic system when only noisy measurements are available. Due to analytic tractability we require that both operation and measurement noises to be Gaussian. The system was developed by Rudolf Kalman [9] in 1960. One of the firsts applications of the filter was the estimation of the trajectory for the NASA Apollo program in 1969 [5].

For a brief introduction to the Kalman filter, one should consult Welch and Bishop [17] and details on the filter are *e.g.*in books by Andrews and Grewal [5], or Haykin [6]. These books contain the rigorous deduction of the equations which describe the operation of the filter. The state-space model is the following:

$$x_k = \mathbf{F}_k x_{k-1} + \mathbf{B}_k u_k + w_k \quad (1)$$
$$z_k = \mathbf{H}_k x_k + v_k, \quad (2)$$

where x_k represents the internal state of the system; z_k is the measurement data; u_k is the control or external input; $\mathbf{F}_k$ is the time-transition matrix; $\mathbf{H}_k$ is the connection between the measurements z_k and the system states x_k. If we assume that we can interfere with the system, then we can do it using the control input: its effect is linear and characterized by the matrix $\mathbf{B}_k$. There are two noise processes in the dynamical system: (1) w_k is the operation noise and (2) v_k is the measurement noise. These have to be *additive* and *white*. Another restriction of eq. (2) is that the noise must be Gaussian with zero mean. In real applications we usually assume a stationary system, meaning that the matrices $\mathbf{F}$, $\mathbf{H}$, $\mathbf{B}$, do not depend on k, therefore we omit the time index.

The operation of the filter is split into two steps:

1 the **prediction** step, based solely on the previous estimated latent state x_{k-1}:

$$x_k^- = \mathbf{F}_k x_{k-1}^+ + \mathbf{B} u_k \quad (3)$$
$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1}^+ \mathbf{F}_k^T + \mathbf{Q}, \quad (4)$$

where the first line is the most probable estimation of the latent state given the previous estimation x_{k-1}^+ and the external input. The second line tells us the *covariance matrix* of the new latent state given the covariance of the previous state $\mathbf{P}_{k-1}^+$, and the known covariance matrix of the operation noise.

2 the **update** step, when the new measurement data z_k are included into the estimation process:

$$x_k^+ = x_k^- + \mathbf{K}_k \left(z_k - \mathbf{H}_k x_k^-\right) \quad (5)$$
$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k^- \quad (6)$$
$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R})^{-1}, \quad (7)$$

where x_k^+ is the estimation based on the system equations from the prediction step eq. (4) and the actual observations z_k. The matrix $\mathbf{R}$ is the covariance matrix corresponding to the measurement process. The matrix $\mathbf{K}_k$ is called the *Kalman gain*, it controls the effect that the current measurement has over the new state.

It is important that we see that the filter is an online estimator: the new values are generated based only on the previous state estimation and the *current measurement*. It means that in operation-time we do not need to store neither the preceding states nor the history of the measurement data.

A second observation is related to eq. (4): by increasing $\mathbf{Q}$, $\mathbf{P}_k^-$ also increases, so the weight of the last state becomes less important if the uncertainty of the model grows.

Thirdly, from eq. (7) we see that if the measurement noise $\mathbf{R}$ tends to infinity, then $\mathbf{K}_k$ tends to zero, meaning that the new measurement has no effect on the estimated state. On the other hand, if $\mathbf{R}$ tends to zero then $\mathbf{K}_k$ tends to $\mathbf{H}_k^{-1}$ and the previous state is neglected.

2.2 Unscented Kalman Filter

The requirement of linearity and the limitations of the noise processes are the main disadvantages of the Kalman filter. To alleviate the constraint of linearity, a statistical approach has been introduced by Uhlmann [15], called the *unscented Kalman filter*, presented in more detail by Julier and Uhlmann [14]. In their extension procedure they used the intuition that it is easier to approximate a probability distribution than a non-linear function. In order to achieve this, they used the so-called *sigma points* to replace the covariance matrices of the latent states $\mathbf{P}_k$. The sigma-points

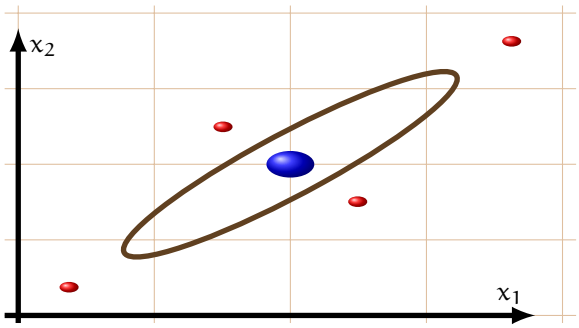

Figure 1. Sigma points

are chosen so that they describe the first and the second moments – the mean and the covariance respectively – of the current system state. It is assumed that the distribution is Gaussian. A good estimation of the posterior distribution of the states is found by propagating only the sigma points through the system equations and this does not need to be linear. The mean and the covariance matrices of the system state vector are estimated by calculating the statistics of the new points. It was shown [14] that information that is of higher order than 2, *e.g.*from Taylor expansion, can also be captured about the posterior distribution by using this method. As mentioned earlier, the state-space model is not necessarily linear:

$$\begin{aligned} x_k &= f(x_{k-1}) + w_k \\ z_k &= h(x_k) + v_k, \end{aligned}$$

where $f()$ and $h()$ are arbitrary non-linear functions and w_k and v_k are white noise processes.

The important step in the unscented Kalman filter framework is the sigma point setting, denoted here with $\mathcal{X}$. The method of the sigma point generation is given by the following equations (the illustration is in Figure 1.):

$$\begin{aligned} \mathcal{X}_0 &= x_k \\ \mathcal{X}_i &= x_k + \left(\sqrt{(n+\lambda)\mathbf{P}_x}\right)_i \quad i = \overline{1,n} \quad (8) \\ \mathcal{X}_{n-i+1} &= x_k - \left(\sqrt{(n+\lambda)\mathbf{P}_x}\right)_{n-i+1}, \quad (9) \end{aligned}$$

where the indexed bracket $(\dots)_i$ represents the ith row of the matrix and

$$\lambda = \alpha^2(n+\kappa) - n.$$

Parameters α and κ stand for scaling, they provide extra degree of freedom to the model. Through α one can underestimate or overestimate the significance of the covariance: the sigma points will be placed closer or farther from the means. It is usually good to set it to a small number, *e.g.*$\alpha = 10^{-3}$. Usually $\kappa = 0$, see [14]. In Figure 1. we give an illustration of sigma points in two dimensional case. The large – blue – dot represents the original state of the system, the ellipse around that point stands for the covariance, and the small – red – dots are the generated the sigma points, where we placed them far from the center for illustration.

Eqs. (8) and (9) contain the square root of a matrix, this is a matrix $\mathbf{B}$ such that $\mathbf{P}_k = \mathbf{B}\mathbf{B}^T$. We note that $\mathbf{P}_x$ always can be decomposed in a square-root form [11], the Cholesky decomposition is suggested to be employed [14]. Each sigma point has a weight W_i:

$$\begin{aligned} W_0^{(m)} &= \frac{\lambda}{n+\lambda} \quad (10) \\ W_0^{(c)} &= \frac{\lambda}{n+\lambda} + (1-\alpha^2+\beta) \quad i = \overline{1,2n} \quad (11) \\ W_i^{(m)} &= W_i^{(c)} = \frac{1}{2(n+\lambda)}, \quad (12) \end{aligned}$$

where β encodes prior information about the distribution of states x. $\beta = 2$ is used for a Gaussian distribution [16]. There are $2n+1$ sigma points, with n the dimension of the states. The superscript (m) stands for the weights when the estimated values of the state and measurement are calculated, whereas W with superscript (c) are used in covariance and correlation determination.

The operation of the filter has the following mathematical formulation:

- **time update**:

$$\begin{aligned} \mathcal{X}_{k|k-1} &= f(\mathcal{X}_{k-1}) \\ \hat{x}_k^- &= \sum_{i=0}^{2n} W_i^{(m)} (\mathcal{X}_{k|k-1})_i \\ \mathbf{P}_k^- &= \sum_{i=0}^{2n} W_i^{(c)} [\mathcal{X}_{k|k-1} - \hat{x}_k^-][\mathcal{X}_{k|k-1} - \hat{x}_k^-]^T \\ \mathcal{Z}_{k|k-1} &= h(\mathcal{X}_{k|k-1}) \\ \hat{z}_k^- &= \sum_{i=0}^{2n} W_i^{(m)} (\mathcal{Z}_{k|k-1})_i \end{aligned}$$

- **measurement update**:

$$\begin{aligned} \mathbf{P}_{\hat{z}_k,\hat{z}_k} &= \sum_{i=0}^{2n} W_i^{(c)} [\mathcal{Z}_{k|k-1} - \hat{z}_k^-][\mathcal{Z}_{k|k-1} - \hat{z}_k^-]^T \\ \mathbf{P}_{\hat{x}_k,\hat{z}_k} &= \sum_{i=0}^{2n} W_i^{(c)} [\mathcal{X}_{k|k-1} - \hat{x}_k^-][\mathcal{Z}_{k|k-1} - \hat{z}_k^-]^T \\ \mathbf{K} &= \mathbf{P}_{\hat{x}_k,\hat{z}_k} \mathbf{P}_{\hat{z}_k,\hat{z}_k}^{-1} \\ x_k &= \hat{x}_k^- + \mathbf{K}(z_k - \hat{z}_k) \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}\mathbf{P}_{\hat{z}_k,\hat{z}_k}\mathbf{K}^T, \end{aligned}$$

where $\hat{x}_k^-$ and $\hat{z}_k^-$ represents the prior values of the internal states and the measured data respectively; $\mathbf{P}_k^-$ is the prior covariance of $\hat{x}_k^-$; and $\mathbf{P}_{\hat{z}_k,\hat{z}_k}$ and $\mathbf{P}_{\hat{x}_k,\hat{z}_k}$ are the empirical correlation matrices. The matrix $\mathbf{K}$ is analogous to the Kalman gain from eq. (7). It is influenced by the correlation matrices: as the variance of $\hat{z}_k^-$ grows, the effect of the new measurement (z_k) diminishes. The correlation between $\hat{x}_k^-$ and $\hat{z}_k^-$ has a converse effect.

2.3 Particle Filters

Particle filtering is an efficient method for estimating unknown probability distributions by simulation. It is a generalization of the above framework, including the non-linear functions we want to estimate – discussed previously – and the non-Gaussian characteristics of the noise. The method was introduced by Gordon [4] in 1993, and detailed in [3]. It is also known as *sequential Monte Carlo method* [13].

The basic idea of the algorithm is that any distribution can be represented as a *weighted particle set*, consisting of pairs $\langle x_k^{(i)}, w_k^{(i)} \rangle$, where $x_k^{(i)}$ is a possible value of the system state and $w_k^{(i)}$ is its plausibility. In particle filtering we generate randomly a set that represents the prior distribution. The posterior distribution is defined by a new particle set and it is determined by sampling form $p(x_k|x_{k-1})$, describing the dynamics of the system. The measurement update step is achieved by setting the weights according to the measurement data. The relation between the measurement data and the latent states is given by the conditional $p(z_k|x_k)$.

Implementing the particle filter in its basic form is unstable: it frequently happens that all weights are almost zero except one which tends to one. In these circumstances the system becomes highly unstable and unusable. To eliminate this effect, importance resampling is advised [2]: a point is chosen form the previous set proportional with the probability expressed by its weight.

Another major issue is the determination of the sample size in order to achieve the best performance. Liu and Kong [10] claim that an efficient set size can be obtained from the weights of the points by the following formula:

$$N_{eff} = \frac{N_s}{1 + var(w_k)},$$

where N_s is the original set size. It is obvious that $N_{eff} \leqslant N_s$, meaning that the sample size decreases in time. To get rid of this effect one has to set a lower limit for the effective sample size.

The operation of the particle filter is presented in Algorithm 1: at each step we aim to obtain a weighted particle set S_k that approximates the posterior distribution of the system state. We assume that the prior distribution is known and given as an other weighted particle set S_{k-1} in the initialization step – line 1. The resampling happens in lines 5-12. First an index is chosen, an index j is sampled with the probability $w_{k-1}^{(j)}$ (line 6)[2]. After selecting a point, adding system dynamics is applied (line 7); this corresponds to the time update step in the previous sections.

Algorithm 1 Particle filter

1: **Input:** $S_{k-1} = \left\{ \langle x_{k-1}^{(i)}, w_{k-1}^{(i)} \rangle | i = \overline{1,n} \right\}$
2:
3: $S_k \leftarrow \emptyset$
4: $\alpha \leftarrow 0$
5: **for** $i \leftarrow \overline{1,n}$ **do**
6: $\quad j \sim p(j|w_{k-1})$
7: $\quad x_k^{(i)} \sim p(x_k|x_{k-1} = x_{k-1}^{(j)})$
8: $\quad w_k^{(i)} \sim p(z_k|x_k = x_k^{(j)})$
9:
10: $\quad \alpha \leftarrow \alpha + w_k^{(i)}$
11: $\quad S_k \leftarrow S_k \cup \{\langle x_k^{(i)}, w_k^{(i)} \rangle\}$
12: **end for**
13:
14: **for** $i \leftarrow \overline{1,n}$ **do**
15: $\quad w_k^{(i)} \leftarrow w_k^{(i)} / \alpha$
16: **end for**
17:
18: **return** S_k

The new measurement data does not affect directly the sample points, rather it changes the weights: they are proportional to their plausibility in the view of the current measurement – line 8. To define a valid probability distribution by the weighted particle set, the sum of the weights has to be one, so normalization is needed, operation performed in lines 14 - 16.

Before presenting the experimental results, we mention a third method to tackle nonlinearity: the *extended Kalman filter* or EKF [12]. EKF alleviates the linearity constraint by considering a nonlinear function, similar to the unscented case and *linearising* via the Taylor expansion to the first order. This method has several drawbacks compared to the extensions given above: the cost of computing the Jacobian in each step is high; this type of linearization can produce large errors, specifically when the model is highly non-linear and the higher order terms in the Taylor series expansion become important. This extension is not pursued any further due to its inadequacy in real applications. Usually we do not know the system, so we cannot perform the linearization.

[2] From the point of view of efficiency, importance resampling (index choosing) can be done in $O(n)$ time by storing the cumulative weights.

3 Experimental Results

Although the actual representation of the object is independent from the tracking algorithm, they cannot be separated. For instance, the *condensation* algorithm, mentioned in Section 1, allows only contour-based representation. We did not use contours, we choose to represent the tracking object by image histogram. The histogram allows us to track arbitrary objects without being obliged to deal with the computational costs of the image recognition.

In the histogram representation an image is represented as a point from a 3×256 dimensional space, where 3 stands for the three color components of the image and 256 is the discretization of the unit interval representing the intensity.

By using histograms, all spacial information of the picture is lost, but by choosing the right object – *e.g.*a monochrome ball – this does not affect the measured characteristics of the algorithms and we gain in reduced computational costs, specifically when the histogram differences are evaluated.[3]

For evaluating the performance of a tracking algorithm, we developed a framework that compares the methods. Using a computer screen, the move of a ball in a rectangular area was simulated. This moving ball had to be tracked by the algorithm, that only saw the rectangular area in which the ball was moving. The color of the ball (black) and the color of the background (white) were selected to be in contrast. We aimed to minimize the effects of the environment, although this was inevitable, due to the altering illumination. The ball had different types of motions: linear, nonlinear,[4] with different speeds and different levels of additive noises could be added to the ball. We conducted 40 experiments, each taking ten minutes. We used a camera with resolution 200×160 that was capable of 22 frame per second.

Five algorithms were tested: Kalman filter (KF), unscented Kalman filter (UKF), particle filter with 10 sample points (PF_{10}), particle filter with 30 sample points (PF_{30}), particle filter with 70 sample points (PF_{70}). During the experiments we measured the following features of the tracking algorithms (1) the speed in frames per second, meaning that the algorithm could look at the screen the given number of times in a second; (2) the number of times the algorithm lost the object it tracked; (3) the time to find the ball – explicit ball searching has not been used.

Table 1 contains general data about the performance of the algorithms: *fps* - frame per second; *error* - difference between the real position of the ball and the estimated value; *lost* - how many times was the ball lost; *found* - how much time it took to find the ball again after loosing it.

[3]We used $\mathbf{L_2}$ norm to compute distances between histograms.

[4]We implemented bouncing dynamics, meaning that the movement was always nonlinear.

Table 1. Performance of the algorithms

Alg.	fps	error	lost	found
PF_{10}	15.81	11.31	1.25	6.25
PF_{30}	10.71	12.28	11.75	5.67
PF_{70}	7.24	15.69	18.25	4.41
KF	10.28	8.57	7.5	8.02
UKF	10.19	8.52	4.0	8.87

Regarding the speeds of the algorithms, the PF has an extra degree of freedom, because it can be scaled by the number of the sample points, as a result it could be faster or slower then the KF and the UKF. We see that the KF and UKF had almost the same speed. The KF and the UKF offer a more reliable estimation about the position of the tracking object. In this evaluation the most important characteristic of an algorithm is the number of times it lost the ball. In this regard the PF_{10} gave surprisingly good results, followed by the UKF and KF.

A second important characteristic of the algorithm is the time required to find the object when it has been lost. The KF and the UKF do not possess implicit searching, so they are quite unimpressing when considering this measure. In the case of the PF this characteristic depends on the number size of the particle set. By increasing the number particles, the frequency of loosing the ball increases, but this is compensated by the fact that the time to find it again is getting smaller. This behaviour can be explained with the fact that the sample points are more spread around the center so there is a higher chance to find the lost object.

Table 2. Loosing the object

Alg.	without noise	with noise
PF_{10}	1.5	1.0
PF_{30}	11.5	12.0
PF_{70}	20.5	16.0
KF	5.5	9.5
UKF	3.0	5.0

Table 2. shows statistics about the frequency of losing the ball when (1) the motion of the ball was straight: without noise; or (2) when it has been distorted with noise in the motion. The experiments we conducted showed that adding noise to the motion did not have significant effect on the PF, moreover the PF_{70} was more efficient with noise added then without it. This is not true about the KF and UKF: they suffered a remarkable fall in performance. The UFK obtained better result than the KF, but this was predictable from the state-space model of the methods: the merely linear model was less powerful.

Table 3. contains data about how many times was the ball lost when linear and non-linear motion was used. These

Table 3. Loosing the object

Alg.	linear motion	non-linear motion
PF_{10}	1.0	1.5
PF_{30}	12.5	11.0
PF_{70}	20.0	16.5
KF	2.0	13.0
UKF	1.5	6.5

results are very similar to Table 2: we conclude that the type of the motion was not relevant for the PF, but it could be seen in the KF-UKF comparison. A significant fall in performance was detected for KF, whereas the UKF proved to be more robust also in this aspect.

4 Conclusions

A first conclusion is that – with most real-world situations – there is no universally good tracking algorithm: we cannot expect a single method to work equally well in all conditions. This means that when selecting from the methods, we have to consider the environment in which the method will be used.

The Kalman filter and the Unscented Kalman filter provide the best estimation for the position of the object but they often fail if non-linearity or non-Gaussian noise appears. Comparing only these two, the unscented solution proved to be a better choice in all cases we tested. To tackle non-linearity and noisy motion, the particle filter can be used. It gives less accurate estimations but non-linearity and noise does not have significant effects on its performance. Forthermore, it can be made more flexible using a suitable sample size. An obvious remark is that we have to trade computational costs for accuracy.

We mention that we aimed a real-world setting, our goal is to embed the algorithm into a robot: the different illuminations, the camera and the use of a computer screen are all settings where calibrating a method might be difficult.

Further on we would like to benchmark the algorithm using real robots where the there are control signals and the noises are not independent of the states. It would be interesting to see the performance of the unscented Kalman filter and comparing it to a particle filter in a situation when *e.g.*a robot aims to find the ball using a stand-alone environment: all the code and the sensors are on the robot.

Acknowledgement

The authors acknowledge the support of the Romanian Ministry of Education, grants CEEX 1473/2006 and PN2 2240/2007.

References

[1] R. Collins. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recognition (CVPR'03)*. IEEE, June 2003.

[2] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2005.

[3] D. Fox. Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[4] N. Gordon, J. Salmond, and A. Smith. A novel approach to non-linear/non-gaussian bayesian state estimation. *IEEE Preceedings on Radar and Signap Processing*, pages 107–113, 1993.

[5] M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB*. John Wilney and Sons, Inc., second edition, 2001.

[6] S. Haykin. *Kalman Filtering and Neural Networks*. John Wilney and Sons, Inc., 2001.

[7] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. on Computer Vision*, volume 1, pages 343–356, Cambridge UK, 1996.

[8] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. In *Int. J. Computer Vision*, volume 1, pages 5–28, 1998.

[9] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[10] A. Kong, J. Lius, and W. Wong. Sequential imputations and bayesian missing data problems. *American Statistical Association 89*, pages 278–288, 1994.

[11] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, second edition, 1992.

[12] M. I. Ribeiro. *Kalman and Extended Kalman Filters: Concept, Derivation and Properties*. Institute for Systems and Robotics, Instituto Superior Tecnico, February 2004.

[13] C. P. Robert and G. Casella. *Monte Carlo Methods*. Springer, second edition, 2004.

[14] J. J. Simon and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management II*, 1997.

[15] J. K. Uhlmann. *Dynamic map building and localization:New theoretical foundations*. PhD thesis, New theoretical foundations, 1995.

[16] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan. The unscented particle filter. In T. G. D. T. K. Leen and V. Tresp, editors, *Advances in Neural Information Processing Systems (NIPS13)*. MIT Press, December 2000.

[17] G. Welch and G. Bishop. An introduction to the kalman filter. In *TR*, 1995.