CIMAT

Centro de Investigación en Matemáticas

Thesis

Active Localization for Bearing-Only Sensor Using Stochastic Control Techniques

by

Oscar A Mar Rangel

Submitted to the Department of Computer Science in partial fulfillment of the requirements for the degree of

Maestro en Ciencias con Especialidad en Computación y Matemáticas Industriales

June 2012

A mi familia y amigos.
Ustedes hacen que el esfuerzo valga la pena.

Acknowledgements

First of all, I would like to thank my parents for all their unconditional support and care. Also, I want to express gratitude to my advisor Jean-Bernard Hayet, for all of his help and valuable advice. Additionally, thanks to you, my class partners, for sharing this wonderful and memorable experience. Finally I would like to say thanks to the *CONACyT* institution, for its financial support and for the opportunity.

Abstract

In this work we present an useful framework to address the problem of the bearing-only localization given noisy measurements of a mobile sensor (i.e robot). The difficulty of this problem recides on (1) the fact that the system is non-linear and (2) the estimation quality is highly dependent on the trajectory followed by the sensor. This framework is based on the use of approximated stochastic optimal control techniques (e.g. Assumed Certainty Equivalence Control, Open-Loop Feedback Control) to obtain the best control policies for the robot. Formulation and simulation results for different state estimation filters (e.g extended kalman filter, particle filter) and different optimization schemes are presented and discussed. The selected optimization schemes are such that can be solved using standard numerical optimization methods. The optimization is focused on the minimization of the estimation uncertainty, which can be measured with the covariance matrix determinant for instance, making it a good choice for the cost function. Additional terms were included in the cost function to improve the method performance and to somehow restrict the resulting behavior of the robot. Also an example application of the method to visual 3D recontruction, including experimental results, is presented. This application makes use of the space carving method [Kutulakos and Seitz 2000] to perform the reconstruction of a 3D object. The experiments were performed using a Nao humanoid robot, using its camera as the bearing-only sensor and employing a simplified motion model (i.e. differential drive) with a fixed linear speed.

Contents

1 Introduction

1.1 Motivation

There are many applications that require an observer, or a group of them, to determine the position of a specific target in the fastest and most accurate way possible, while, at the begining, the position of this target is unknown. For example in disaster situations, there might be injured people that need to be rescued and, for that, need to be localized first. Also, for military strategy and tactics, it is required to gather as much information about the enemy as possible, such as the location of enemy camps or approaching units. In fact, almost in any robotics application, it is required for the robot to determine the position of certain objects or "targets" in order to carry out the task or tasks that have been assigned to it.

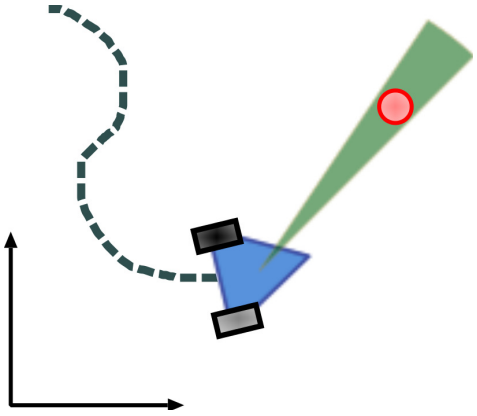

Figure 1.1: Mobile observer with a bearing-only sensor (e.g., a camera), detecting a target.

1.2 Problem Statement

Let us consider a mobile observer (e.g. a robot), with a bearing-only sensor, which is a sensor that can only give information of the relative orientation of the detected object (e.g. a camera, a rf-receiver). Suppose that this observer needs to detect and keep track of one or more targets in order to improve its knowledge of the environment or to estimate the location of a target. A trajectory and its associated controls have to be computed to get the most information from the environment.

The observation gives partial information of the state, so, in order to obtain an estimation of the position of certain object with respect to itself, the observer (robot) has to take several measurements of the object relative angle from different positions, so that it can compute the position by triangulation. Also, the measurement is a non-linear function of the current system state. For a given noise model on the bearing angle, the resulting uncertainty on the estimate is larger from farther positions. Figure 1.2 shows an example of the evolution of the estimation along time. The green ellipse indicates the uncertainty volume, which reduces as the robot moves, but this is highly dependent on the actual trajectory.

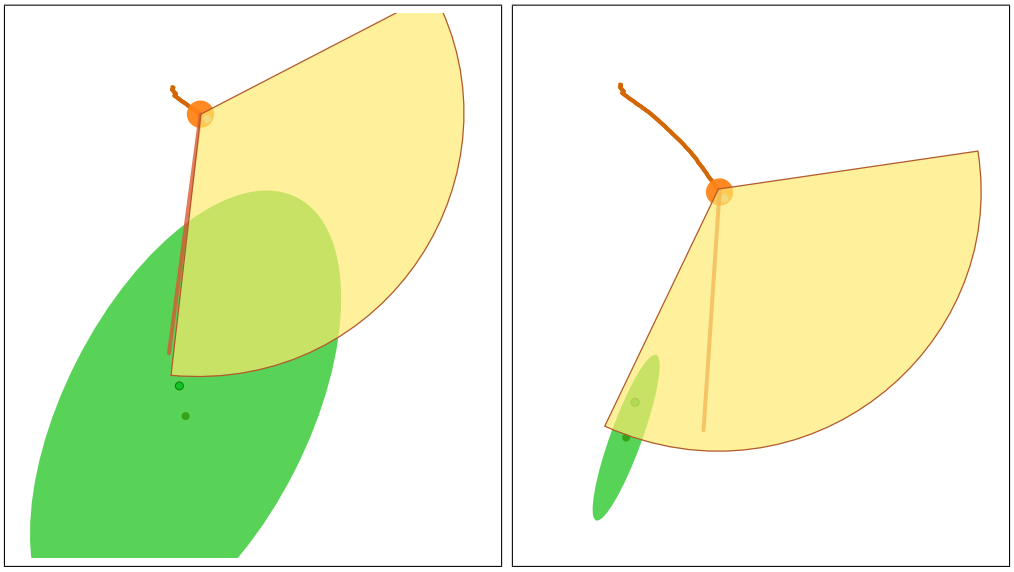

Figure 1.2: These are two stages in the estimation process, where the green ellipse represents the uncertainty in the estimation of the target possition. Is there some optimal way to move the robot so that the estimation goes for the best (e.g., such that the area of the ellipse decreases in the fastest way possible)?

1.2.1 Considerations

Measurements and movements are noisy, and we want to determine the sensor (robot) controls by taking into account these disturbances. We will define an objective function, at some horizon, that depends on the quality of estimation at this horizon. By definition, this estimation is not known in advance since observations have not taken place yet, thus this is a stochastic optimal control problem.

1.3 State of the Art

1.3.1 Bearing-Only Tracking

The bearing-only tracking (BOT), is now a classical, non trivial problem in the literature [Stansfield 1947; Aidala 1979; Fawcett 1988] of non-linear state estimation, and there have been several approaches that seek solving it in an optimal way. The problem is present in different contexts, fields and environments, like in militar or naval strategy and defense applications. Also in aeronautics applications, like Unmanned Aerial Systems (UAS), which are used for Intelligence, Surveillance and Reconnaissance (ISR) procedures, where targets in the ground have to be localized and tracked, fusing all sensor and self-localization data [Ponda 2008]. Another instance is in biology and wildlife research, for instance the study of fish tagged with radio transmitters [Tokekar et al. 2011]. There are several modalities as, for instance, the target might be moving, or might be assumed to be static. There might be several static sensors or only a mobile one. But all of them attempt to estimate the current position of a target or set of targets, by using a control policy that will improve as much as possible this estimation.

1.3.2 Partially Observable Markov Decision Processes

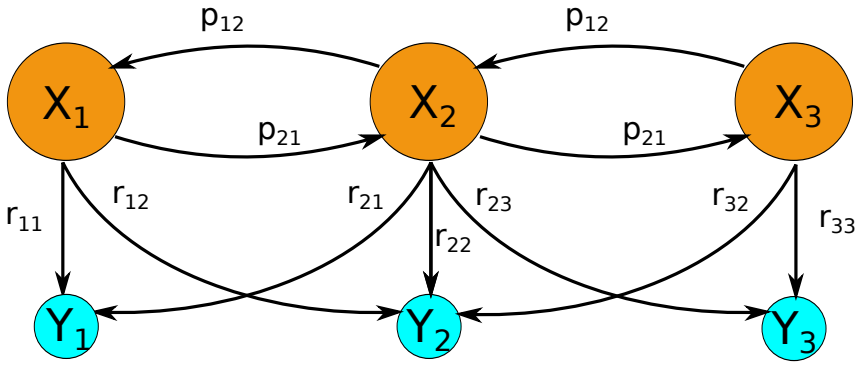

Figure 1.3: An example of a simple POMDP.

Partially Observable Markov Decision Processes (POMDP) [Smallwood and Sondik 1973], is a general approach that helps us solve planning problems in uncertain environments. On these approaches, the dynamics of the underlying system are modeled as a Hidden Markov Model (HMM), as depicted in Figure 1.3. In this model, the state itself is not observable, and it has to be estimated by indirect observations of it, in this case the noisy bearing measures. The sets of possible controls and states are discretised, and a number N of future states (horizon) is defined. If there are N future states remaining, the problem is to select a control sequence (policy) $U_k^N \triangleq \{u_k, u_{k+1}, ..., u_{k+N}\}$ that will optimize the performance of the system during the

remaining states. The problem is stated using a reward or cost functional, which depends on a belief distribution. The reward itself has to be estimated, because it depends on a probability distribution, and it is also not observable. The resulting problem is usually solved using dynamic programming. To this end, an expected maximum reward $V_N(\pi)$ is defined, this in terms of a recursive equation, where N is the remaining number of states, π is a belief space representation, expressed as a M dimensional vector $\{\pi_0, \pi_1, ..., \pi_M\}$, where π_i is the probability of being on a specific state i and M is the number of possible states. The first exact solution method proposed, according to [Lovejoy 1991], makes use of a property of the reward function, which states that for any finite horizon, $V_N(\pi)$ is piecewise linear and convex. However these methods have the major disadvantage of being computationally expensive, and in fact, intractable for most realistic problems [Lovejoy 1991].

There have been some research in the application of these methods to the BOT problem. For example, in [Trémois and Le Cadre 1999], the authors present a framework based on POMDP, in which they define the belief and the cost functional in terms of the Fisher Information Matrix (FIM), and express the problem in a dynamic programming form. They even propose an alternative exact algorithm to solve the POMDP problem, finally presenting results for a very coarse state-space discretisation. There have been several attempts to reduce the numerical burden related to these methods to make them more feasible for real applications. Some of these methods involve different types of approximations [Lovejoy 1991] and heuristics.

1.3.3 Enumeration and Brute Force Methods

Methods based on enumeration basically compute all the possible values (in a discrete space) of a *cost function* L, which is a function of the state that measures the performance of the method. This, to find the lowest (or highest) value of this cost function, and thus the best choice for a set of input parameters. For this particular problem, the search is performed in the space of controls, so the optimal control sequence U_k^N is to be found. These kind of methods are normally combinatorial, because in order to find the solution, the cost function has to be evaluated for all the possible combinations of controls and resulting states. In [Logothetis et al. 1997], the authors present an enumeration method for BOT using information theory measures as the cost function, such as the trace of Fisher Information Matrix or mutual information. They also propose an optimal prunning technique to avoid checking every possibility, and thus reducing the computational complexity. In [Tokekar et al. 2011], several enumeration methods are proposed and analysed, including a greedy method, which only takes in consideration one step ahead to decide the next configuration for the robot. They also propose another method using a min-max enumeration tree to find the optimal control sequence, taking in consideration worst-case observations. Because of their complexity, these methods also use only very coarse discretizations.

1.3.4 Stochastic Optimal Control Methods

Stochastic Control is a subfield of control theory, which makes use of methods and techniques to determine the best actions to carry out on a system for which only uncertain information is available and/or there is uncertainty on the result of the application of a specific action. The goal is to make the system reach a specific state in an efficient manner. On these methods, the uncertainty is expressed in the form of probabilities and an objective function (cost function) is defined in terms of random variables. Therefore L is now a function of the state X and the process noises $\mathcal{W}$.

A stochastic optimal control (closed-loop) for non-linear systems is, in general, complex in formulation and in computational requirements, normally because it requires, as in POMDP, the use of dynamic programming to solve recursive equations, which causes complexity to grow exponentially with the number of variables. However there are several possible approximated or simplified (sub-optimal) methods that help making these problems more tractable [Martinez and Soares 2002; Kappen 2007]. Among them, the Certainty Equivalence Control (CEC) and Open-Loop Feedback Control (OLFC) are the most common.

There are some important concepts in the control theory that help understand the differences between the kinds of stochastic control methods. Those concepts are: open/closed-loop and feedback [Bar-Shalom and Tse 1974]. An *open-loop control policy* states that the resulting control sequence will be obtained in terms of the initial state probability distribution only. This means that the error at the final state might be large, depending on the process noise. A *closed-loop control policy* implies that each element u_k of the resulting control sequence U_1^N will be computed in terms of all the measurements Y_1^N [Bar-Shalom and Sivan 1969], including the ones that have not been adquired yet, the latters in the form of an expected value (maybe the worst case ones). So, this way, all the available information is used. A *feedback control policy* makes use of the past measures Y_0^k, being k the current step, to obtain each control u_k. Therefore the possible errors in a state transition will be compensated with this next control, but assuming no additional information will be obtained. This last approach is sub-optimal, in the sense that certain future observations could have improved the overall performance of the controller, but those are not considered. An example of this kind of policy is the so called Open-Loop Feedback Control (OLFC) [Bar-Shalom and Sivan 1969].

This difference between closed-loop and feedback control, is very important to understand the concept of the *dual effect*, which states that the control currently applied to the system can affect the future state uncertainty. If the control can not affect the uncertainty, the system is called *neutral* [Feldbaum 1965; Bar-Shalom and Tse 1974]. If the controller takes advantage of this dual effect, it is considered as *actively adaptive control*. If the opposite is true, it is considered *passively adaptive control*. Some advantages of the actively adaptive control are that the controller has to exercise less of what is called *caution*, which is the ability reduce the negative effects of uncertainty on the system performance. However, as the closed-loop controller takes in consideration the fact that future measures will be taken, and actions will be realized with this information, it will exercise less caution. Another advantage is the active learning itself,

which is the improvement in the estimation with the "help" of the control.

Another important concept is the *certainty equivalence principle*, which states that under specific conditions, the stochastic optimal closed-loop control is equivalent to its deterministic counterpart. It is, assuming perfect state transitions, using the current state estimate $\hat{X}_k = E\{X_k|Y_k, U_{k-1}\}$ for the initial state [Bar-Shalom and Tse 1974]. This principle does not always hold. Specifically, it holds if and only if the system is neutral, which means considering future measures does not help improve the estimation. This is very related to the *separation principle*, which holds when the closed-loop control can be expressed in terms of the current state estimation, not necessarily in the same form of the deterministic case, so this principle is weaker than the certainty equivalence. This principle is applied in the Certainty Equivalence Control to simplify the control problem. It is sometimes used even if the principle does not hold, as a sub-optimal approach (Assumed Certainty Equivalence).

There is a subfield of stochastic control called *Adaptive Dual Control*, which takes explicitly into account the existing uncertainty into the control strategy [Unbehauen 2000], ensuring the dual effect will be present. There exist several strategies and methods related to this particular approach, which normally are similar to those of stochastic optimal control (CEC, OLFC), but with an uncertainty measure added to the cost function, in an explicit or implicit way.

Finally, another family of stochastic control methods are the ones based on Stochastic Approximation (SA), first proposed in [Robbins and Monro 1951], as a sub-optimal Stochastic Optimization method. This method seeks to find the minimum of the cost function using the *stochastic gradient* $\frac{\partial}{\partial X} L(X, \mathcal{W})$ of the cost funtion, which is a random function of the state X and the noise random variable W. A gradient descent method can be proposed in the following way: $X_{k+1} = X_k + \alpha_k \frac{\partial}{\partial X} L(X_k, \hat{\mathcal{W}}_k)$. If no analytic expression is available, a numeric approximation, such as the *Simultaneous Perturbation Stochastic Approximation* (SPSA) can be used [Spall 1992].

There has been some research focused in the application of stochastic control to the bearing-only tracking problem. For instance, in [Oshman and Davidson 1999], an optimization framework for the bearing-only target localization problem using the FIM is presented. They propose several solution methods, reformulating the cost function, for continuous and discrete time controls. The scheme they present does not take into account the uncertainty for the future steps so they handle the problem in a deterministic optimization fashion. Another example is [Skoglar et al. 2009]. In this work, the authors present several methods to solve the BOT problem are presented, applying both CEC and OLFC and making use of different information measures. In particular, an approximation of the differential entropy measure using Particle Filter and a SA gradient descent based algorithm are proposed. Also in [Singh et al. 2007], a method based in Particle Filter and SA, focusing on the convergence properties of this approach is proposed and analysed.

This current research work falls into the stochastic control category, which was chosen because it seems very promsing in terms of performance and flexibility, also, it is an active research area.

2

Kalman Filter Based Methods

2.1 Introduction

In this chapter, a first strategy to solve the tracking problem is studied. The following methods are based in an ACE approach, in the sense that deterministic (noiseless) observations are assumed. Even though the conventional CE control has no dual effect in the cases where it is optimal [Bar-Shalom and Tse 1974], in this approach we specifically set a measure of the system uncertainty as the cost function for the optimization scheme. And, actually, there is no intention of directly controlling the final robot state. The estimation part is done with a version of the Kalman filter adapted for the non-linear system. Results with two versions of the filter are presented, with an Extended Kalman Filter (EKF), and with an Unscented Kalman Filter (UKF). Also, the optimization scheme used to solve the problem and obtain the control sequence is explained.

2.2 Modeling

The following is the basic system modeling that will be used in most of the approaches, throughout this work. The system consists of a target slowly moving, in a stationary random walk in the xy-plane. The observer will be a mobile differential drive robot, also moving in the plane with constant linear speed, being the angular speed $U_k = \omega_k$ the controlled parameter. And, of course, the robot will have a bearing-only sensor, which will provide a measurement of the direction (angle) $Y_k = \phi_k$ in which the target is found, with respect to the robot orientation.

First of all, the total state will be represented with the variable X_k. It will be a vector containing all the required state information, in the current k time interval or step. For now, it

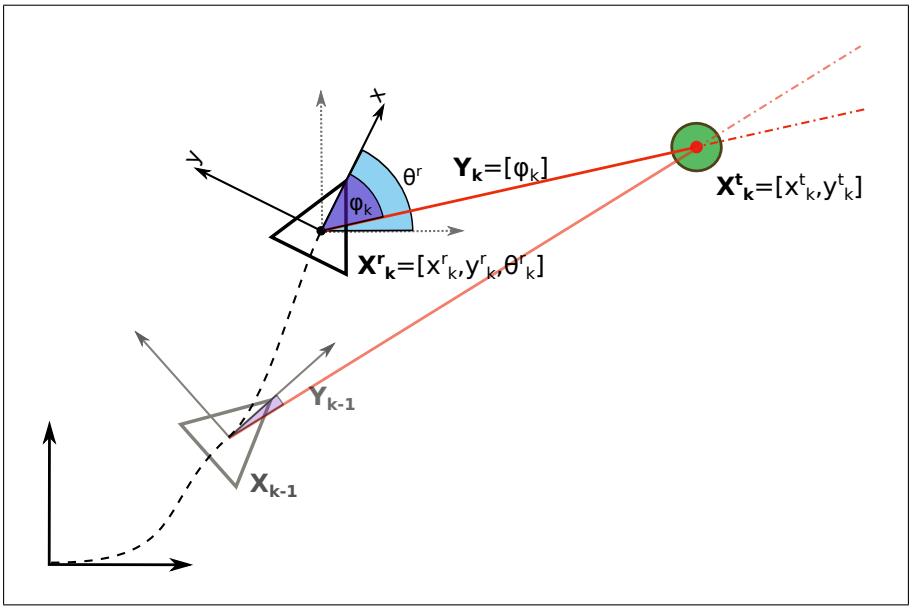

Figure 2.1: *System model diagram.* The darker triangle represents the robot in the k time interval. The green circle indicates the target position. The dashed black line represents the trajectory since the begining. The red line and the purple circle section represent the target detection from the robot (in the form of an angle relative to the robot orientation). The blue circle section depicts the current robot orientation θ^r.

will contain the robot pose and the target position in the plane, respectively, $X_k^r = [x_k^r, y_k^r, \theta_k^r]'$ and $X_k^t = [x_k^t, y_k^t]'$. Therefore, the total state vector X_k will be of the form:

$$X_k = \begin{bmatrix} X_k^t \\ X_k^r \end{bmatrix}. \tag{2.1}$$

The sub-index k might be dropped where not required, for notation simplicity.

The target and the robots motions are independent processes, therefore they can be modeled as independent systems.

For the target it will be:

$$X_{k+1}^t = f^t(X_k^t, \mathcal{W}_k^t) = X_k^t + \mathcal{W}_k^t \Delta T \tag{2.2}$$

where $\mathcal{W}_k$ is a random vector of the same dimension as X^t, and distributed according to $\mathcal{N}(0, \Sigma^t)$. ΔT is the sampling time.

In this case the robot will be assumed as noiseless in motion (i.e. its motion is deterministic), so we have:

$$X_{k+1}^r = f^r(X_k^r, \omega_k) = X_k^r + \begin{bmatrix} v\cos(\theta_k^r + \omega_k \Delta T)\Delta T \\ v\sin(\theta_k^r + \omega_k \Delta T)\Delta T \\ \omega_k \Delta T \end{bmatrix} \tag{2.3}$$

where v is the linear speed, which in this model it is taken as constant.

It is worth noting that, in this model, the control ω_k, is applied first to the orientation then to the position components, this is to observe an immediate effect to the robot position caused by this control.

The observation model states what should be the observation from the current state. The robot bearing observations are dependent of both the robot and the target states, i.e. the total state. This dependence is described by the expression in equation 2.4. It can also be observed in figure 2.1, where the observation is referred to as φ_k.

$$Y_k = g(X_k, \mathcal{W}_k^y) = \arctan_2(y^t - y^r, x^t - x^r) - \theta^r + \mathcal{W}_k^y \tag{2.4}$$

where $\arctan_2(y, x)$ is a function that returns the arc tangent of the quotient $\frac{y}{x}$, also using the signs of x and y to determine the quadrant of the result. And $\mathcal{W}_k^y \sim \mathcal{N}(0, \Sigma^y)$ is a normal additive random noise.

As can be deduced by the inspection of equations 2.3 - 2.4, the system is highly non-linear. Additionally, equation 2.4 shows the equal contribution of both states to the observation. This gives us a hint of why the robot motion is important to improve the target estimation. Also, it is clear that the process is partially observable, because there is a relative angle measure only and all the state variables have to be inferred from it.

2.3 Kalman Filter Review

As stated by [Thrun et al. 2005], state estimation addresses the problem of estimating quantities from sensor data that are not directly observable, but that can be inferred. A *recursive filter* is a type of signal filter that uses current noisy data to recursively improve a previous estimation of the internal state of a dynamic system. The *Kalman Filter* [Kalman 1960], which is based on the (probabilistic) Bayes Filter, is the most well known of this kind of filters, mainly because it has an exact closed form solution. But it is limited to linear systems with additive Gaussian noises.

2.3.1 The Bayes Filter

Bayes filtering is based on the application of Bayes rule to recursively calculate what is called a belief distribution, which is a distribution of the probability for the system of being in each state in a particular moment [Thrun et al. 2005].

$$p(X_k|Y_1^k, U_1^k) = \eta p(Y_k|X_k) \int p(X_k|X_{k-1}, U_k) p(X_{k-1}|Y_1^{k-1}, U_1^{k-1}) dX_{k-1} \tag{2.5}$$

where the notation $p(..|..)$ indicates conditional probability. The notation Y_1^k indicates the set of all Y^j random variables from Y^1, up to Y^k. The η is a normalization term, which is an integral (total probability law) of all the possible values of $p(X_k|Y_1^k, U_1^k)$, so that the result is actually a probability distribution. This is actually a recursive equation, because the resulting probability $p(X_k|Y_1^k, U_1^k)$ depends on its value in the previous step $k-1$.

The Bayes filter relies on the *Markov assumption* or the complete state assumtion. It postulates that past and future data are (conditionally) independent if the current state X_k is known. This is actually a very restrictive assumption and hardly accomplished. But, in fact, very accurate models are not always desired because of the computational complexity. Anyhow, this approach has proven to be surprisingly robust to this simplified modeling. On the other hand, the direct application of the Bayes filter approach, is rather unfeasible for general cases, mostly because of the integration steps that have to be carried out. This is where the Kalman Filter goes into action.

2.3.2 Kalman Filter

As already stated, the Kalman Filter (KF), is a closed-form solution of the Bayes Filter expression (check equation 2.5), for the linear additive gaussian case. The underlying linear system is expressed with the following equations:

- linear dynamic model

$$X_k = AX_{k-1} + BU_k + \mathcal{W}^A \tag{2.6}$$

  where U_k is the vector of controls applied to the system in the current time. A is a square matrix that represents the correspondences of the previous state with the current state. B is a matrix that states the contribution of the control vector to the new state. And $\mathcal{W}^A$ is a random noise vector (motion noise) distributed as $\mathcal{N}(0, \Sigma^A)$.

- linear measurement model

$$Y_k = MX_k + \mathcal{W}^M \tag{2.7}$$

  where M is a matrix that indicates the relation of the system state X_k and the measurement Y_k. $\mathcal{W}^M$ is a random noise vector (observation noise) distributed as $\mathcal{N}(0, \Sigma^M)$.

The KF algorithm is classically divided in two steps: *prediction* and *correction*. In the first one, the integration operation in equation 2.5, is applied in the form of an addition and a linear transformation. The correction step is a bit more complicated, it is the result of the previous resulting distribution conditioned to the measurement Y_k, which can be computed due the gaussian distributions properties. A summary of the equations is now presented.

- Prediction

$$\bar{X}_k^- = A\bar{X}_{k-1} + BU_k \tag{2.8}$$

$$\Sigma_k^- = A\Sigma_{k-1}A^T + \Sigma^A \tag{2.9}$$

- Correction

$$K = \Sigma_k^- M^T (M\Sigma_k^- M^T + \Sigma^M)^{-1} \tag{2.10}$$

$$\bar{X}_k = \bar{X}_k^- + K(Y_k - M\bar{X}_k^-) \tag{2.11}$$

$$\Sigma_k = (I - KM)\Sigma_k^- \tag{2.12}$$

In literature the expression $(Y_k - M\bar{X}_k^-)$ is usually called the innovation, because it indicates the information added by the observation. And the K is called the Kalman gain, because it indicates how much the innovation term is trusted. But, again, the standard KF, is limited to linear Gaussian systems, this is why some approximated versions for non-linear systems have been proposed.

2.3.3 Extended Kalman Filter

The first attempt to overcome the limitations of the KF is the Extended Kalman Filter, or, EKF, from now on. What this approach does is to approximate the non-linear model functions to linear functions using the first term of their Taylor expansions around the current estimation mean. This linearization of the model functions causes the approximated posterior beliefs distributions to remain Gaussian when the KF equations are applied. In order to perform this linearization by Taylor expansion, the Jacobian of the function has to be computed. This Jacobian is evaluated on the current estimation mean, which is the point that should give us minor linearization error. For the case of the state transition function, as it satisfies $f : \mathbb{R}^n \to \mathbb{R}^n$ (with n being the state dimension), its Jacobian J^f is a $n \times n$ matrix. In the case of the observation model, its Jacobian J^g is a $m \times n$ matrix, where m is the observation vector size (e.g. $m = 1$ in the current model). The Kalman Filter equations have to be adjusted to include the linearization procedure. The equations for the EKF are as follows:

- *Prediction.* The prediction step for the mean is performed applying directly the transition function without noise. It is on the covariance matrix where the Jacobian is used, replacing the matrix A of the linear KF.

$$\bar{X}_k^- = f(\bar{X}_{k-1}, U_k) \tag{2.13}$$

$$\Sigma_k^- = J_k^f \Sigma_{k-1} (J_k^f)^T + \Sigma^A \tag{2.14}$$

- *Correction.* Similarly, in the correction step, the Jacobian of the observation function J_k^g replaces the matrix M. Also, the innovation term is replaced by $[Y_k - g(\bar{X}_k^-)]$.

$$K = \Sigma_k^- (J_k^g)^T (J_k^g \Sigma_k^- (J_k^g)^T + \Sigma^M)^{-1} \tag{2.15}$$

$$\bar{X}_k = \bar{X}_k^- + K[Y_k - g(\bar{X}_k^-)] \tag{2.16}$$

$$\Sigma_k = (I - KJ_k^g)\Sigma_k^- \tag{2.17}$$

This method works fairly well for moderately non-linear functions or for small levels of noise, and it is vastly used in several fields. But, its performance obviously degrades as the non-linearity or noise increase. This may lead, in the worst case, to divergence of the method.

2.3.4 Unscented Kalman Filter

Another variant of the KF, not as popular as the EKF, is the Unscented Kalman Filter (UKF) [Julier and Uhlmann 1997]. The idea of this method is to approximate the first two moments of the posterior distribution (i.e mean and variance) directly, instead of approximating the non-linear function. The idea is to define a set of points to sample around the current distribution mean. These points, termed *Sigma Points*, are used to compute the *Unscented Transformation*, which is *a method for calculating the statistics of a random variable which undergoes a non-linear transformation* [Julier and Uhlmann 1997]. This method slightly resembles the Monte Carlo sampling techniques, but, in this case, the samples are deterministic. The distribution moments are computed by means of some weighted sum equations using the sigma points. $2n+1$ sigma points $\mathcal{X}^{(i)}$ and their weights $W^{(i)}$ have to be computed. In the simplest form of this method, these values are selected as follows:

- Sigma Points
  - for $i = 0$
  $$\mathcal{X}^{(0)} = \bar{X} \tag{2.18}$$
  - for $i = [1 : n]$
  $$\mathcal{X}^{(i)} = \bar{X} + [\sqrt{(n+\kappa)\Sigma_X}]_i \tag{2.19}$$
  - for $i = [n+1 : 2n]$
  $$\mathcal{X}^{(i)} = \bar{X} - [\sqrt{(n+\kappa)\Sigma_X}]_{i-n} \tag{2.20}$$

  where $[A]_k$ is an operator that returns the k-th column of the matrix A. $\kappa \in \mathbb{R}$ is a scale factor, and Σ_X is the covariance matrix. And $\sqrt{\cdot}$ is a matrix square root operator (e.g. a *Cholesky decomposition* [Press et al. 1992]).

- Weights
  - for $i = 0$
  $$W^{(0)} = \frac{\kappa}{n+\kappa} \tag{2.21}$$
  - for $i = [1 : 2n]$
  $$W^{(i)} = \frac{1}{2(n+\kappa)} \tag{2.22}$$

    the weights are normalized, thus their sum is 1.

Now, in order to compute the desired distribution moments, the following equations are used:

- Mean
$$\bar{X} = \sum_{j=0}^{2n} W^{(j)} \mathcal{X}^{(j)}. \tag{2.23}$$

- Covariance matrix

$$\Sigma_X = \sum_{j=0}^{2n} W^{(j)}(\mathcal{X}^{(j)} - \bar{X})(\mathcal{X}^{(j)} - \bar{X})^T. \tag{2.24}$$

By integrating the aforementioned equations (Unscented Transform) in the KF method, it is possible to develop a whole new approach for the estimation problem. In general, this method performs very well even for highly non-linear systems, having a performance comparable to a second order Gaussian Filter [Julier and Uhlmann 1997]. Also, it overcomes most of the EKF problems and limitations (e.g. the bias and divergence). But, of course, it is still an aproximated approach with inherent degradation of the information. More details of this variant of the KF can be consulted in [Julier and Uhlmann 1997].

2.4 Control Scheme

The control scheme selected for this first approach is similar to the CEC presented in [Skoglar et al. 2009], except that it directly uses the determinant of the estimation covariance matrix, instead of the information matrix. It consists in solving the following optimization problem:

$$\min_{U_{N-1}} \; L(X_N) = \det \Sigma_N \tag{2.25}$$

$$\begin{aligned} s.t. \qquad & u_k \in \mathcal{U} \\ & X_0 = \hat{X}_0 \\ & X^t_{k+1} = f^t(X^t_k, 0) \\ & X^r_{k+1} = f^r(X^r_k, u_k) \\ & y_k = g(X_k, 0) \\ & \Sigma_{k+1} = h(\Sigma_k) \end{aligned}$$

where Σ_N is the estimation covariance matrix at the final step $k = N$. $\mathcal{U}$ is the admissible control space. And $h(\Sigma_k)$ corresponds to the correction step equations for the covariance matrix of either version of the non-linear Kalman filter (e.g., equation 2.17 for the EKF).

The previous formulation is equivalent to performing a simulation of N steps of a Kalman Filter with the same initial parameters, for each candidate control sequence U_N.

2.4.1 Cost Function Terms

The original cost function of [Skoglar et al. 2009] was modified to directly include some control restrictions and to add some other terms to regularize the result and to improve other properties. Figure 2.2 depicts the kind of modifications performed on the original cost function. The control limits were included this way to avoid the use of *Constrained Optimization* (for further reading in optimization types and techniques, refer to [Nocedal 2006]), which normally is heavier and increases complexity in formulation. First of all, instead of using the $(\det \Sigma_N)$ value directly, a ratio between the first prediction step covariance matrix determinant $(\det \Sigma_1^-)$ and this $(\det \Sigma_N)$, was chosen, in order to have a "normalized" result.

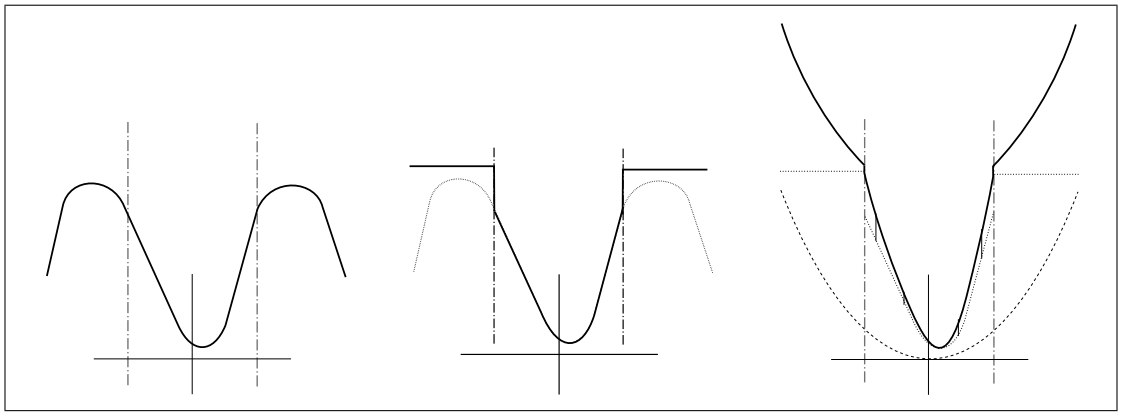

Figure 2.2: Cost function modifications to include restrictions and make it convex. *Left*, original cost function and control limits restrictions. *Center*, discontinuity added to avoid other local minima. *Right*, control vector squared norm added to make function convex.

Control Restrictions

Because of the problem definition, the control (angular speed) causes the cost function to be periodic. Therefore there is theoretically more than one possible solution for the problem. But usually the least effort solution is preferred, that is why we add a cost penalization to the control vector norm, which also serves as a Regularization term.

View Range Restrictions

Another important issue to take in consideration is the visual range of the robot, which is limited by the sensor physical properties. It is fundamental for the method to have the target always on sight. This is why a term penalizing the magnitude of the (expected) bearing measure φ_k^2 is also added. The term is quadratic, for smoothness. Also, as there will not be any correction in the real filter if the target is not detected, the $[\det \Sigma_k]$ value is conditioned to the successful detection of the target. This way, if the target is not in range, the cost function will not decrease.

Additional terms

So, in a similar way, it is possible to add multiple terms to this cost function in order to make the solution more robust and applicable to real life problems. Equation 2.26 shows the general idea for the cost function. For instance, it is possible to add a term penalizing the robot for being too close to the target, or to try make the robot keep the target centered on sight.

$$L(X_N) = \lambda_1 \frac{\det(\Sigma_N)}{\det \Sigma_1^-} + \lambda_2 \|U_N\|_2^2 + \lambda_3 \varphi_k^2 + \lambda_4 [(x^r - x^t)^2 + (y^r - y^t)^2]^{-\frac{1}{2}} + ... \tag{2.26}$$

where the λ_i are weights to ponderate the influence of each term in the resulting control.

2.5 Numerical Optimization Discussion

The optimization method used for the implementation of the following simulations and most of the experimental results, is a derivative-free method based on the Nelder-Mead simplex-reflexion [Nelder and Mead 1965]. Also known as the amoeba algorithm, it is not related to the simplex method used in linear programming. The advantage of these kind of methods is that they usually require fewer objective function evaluations than other derivative free methods like finite difference gradient-based methods. Also the latter are more susceptible of failing in the presence of noise [Nocedal 2006]. This is specially useful for objective functions with time-consuming evaluations. A disadvantage of this method is that its convergence properties have not been very well studied or proven.

2.5.1 The Nelder-Mead Simplex Method

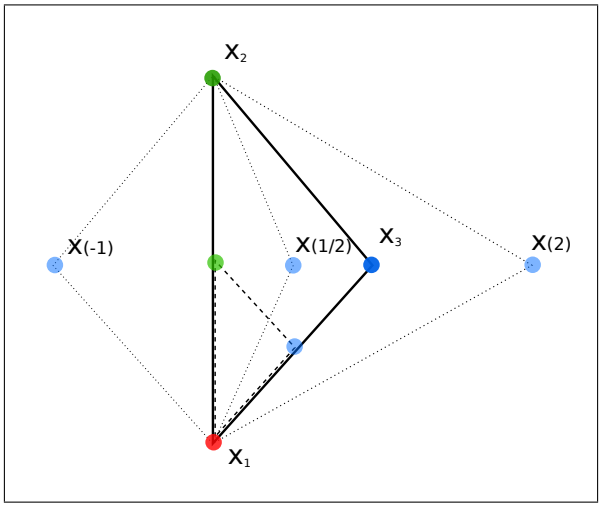

Figure 2.3: Nelder-Mead Simplex operations for a function in $\mathbb{R}^3$. Assuming the current simplex is composed of the points $[X_1, X_2, X_3]$ and the point with higher function evaluation is X_3, the possible operations are: Contraction $x(1/2)$. Expansion $x(2)$. Reflection $x(-1)$. And shrinkage (dashed line). Where the number in parenthesis indicate the scale of the applied operation.

This algorithm is very simple and consists in keeping in every step, $n+1$ points of interest in $\mathbb{R}^n$ (i.e. a simplex), and in every iteration it is required to replace the point with the worst function evaluation with a better one, using the following actions: reflecting, expanding, or contracting the simplex along the line joining the worst vertex with the centroid of the remaining vertices. If this fails, the simplex is shrinked, keeping only the best value vertex [Nocedal 2006]. This is repeated until a stop criteria, like a minimum size for the simplex, is fulfilled. In figure 2.3, an explanation of the method operations in $\mathbb{R}^3$ is depicted.

2.6 Simulation Results

2.6.1 Software Framework

Because of the simplifying assumptions in the model and the process, all the results in this chapter are in simulation only. The simulations were carried out with a simple graphic application (see figure 2.4) implemented in the Java™ programming language, using an open source graphics library and API called *Processing* [Reas and Fry 2010]. Also the mathematics library *commons-math*, of Apache, is used for the numerical optimization routines.

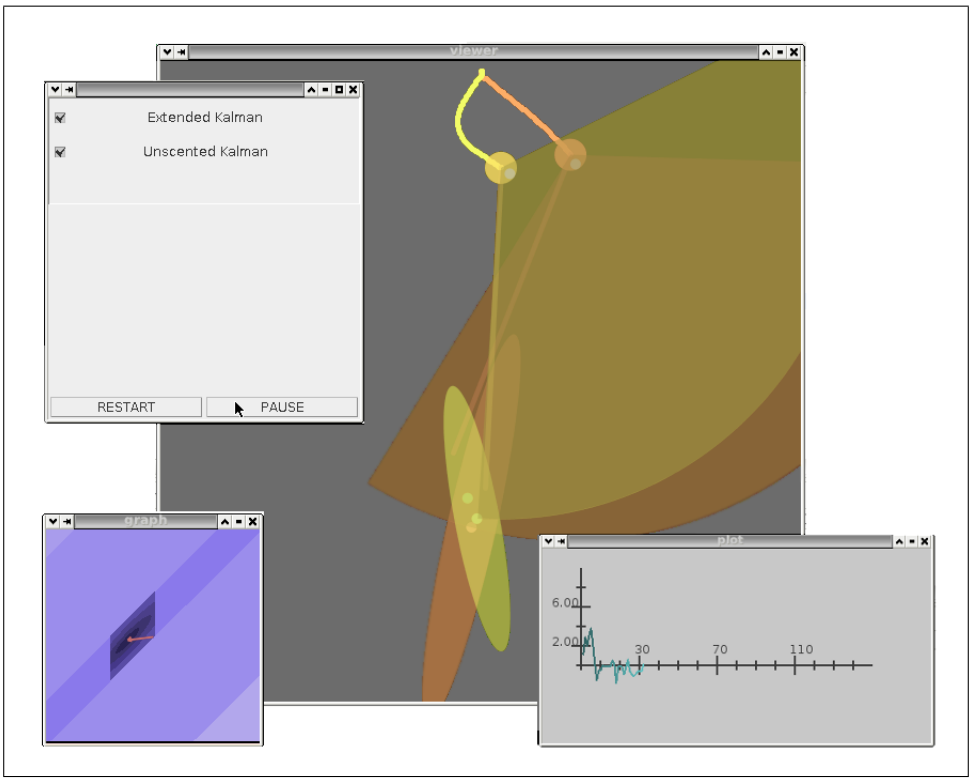

Figure 2.4: Developed Java Application preview. *Center:* visualization window. *Top-Left:* simulation control interface. *Bottom-Left:* Cost function contours graph. *Bottom-Right:* obtained control.

2.6.2 Extended Kalman Filter

In This section, the first set of simulation results are presented. These simulations were carried out using the EKF, and testing different modifications to the cost function parameters. For instance, in figures 2.5 - 2.6, the results of the control scheme using the $\det(\Sigma_N)$ term only, with the horizon $N = 1$ (i.e. a very "greedy" approach) are presented. As it can be observed, the overall error and uncertainty decrease as time goes on. Also, as this is an active, stochastic method, every resulting trajectory is different, depending on the actual noise realizations. But, as seen in figure 2.7, all of these trajectories have the same tendency, which is to get closer to the target, and to triangulate its position.

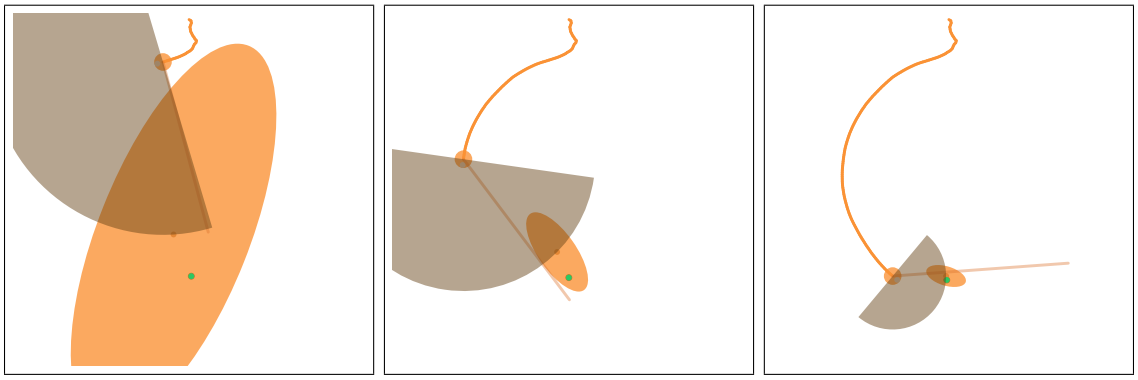

Figure 2.5: Robot trajectory, target position estimation and uncertainty ellipse using an EKF estimator, in different stages of the simulation ($k = \{30, 90, 150\}$). The robot combines two tendencies, to get closer to the target, and to move around the target to keep triangulating the position.

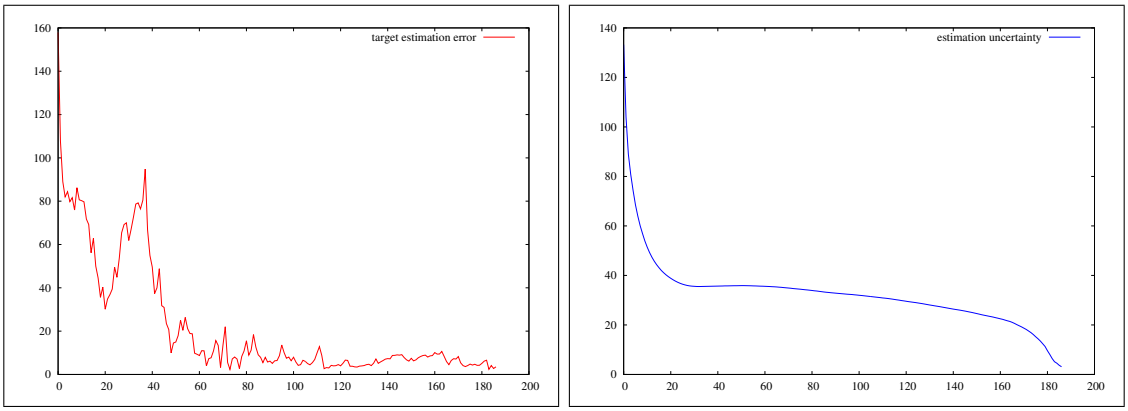

Figure 2.6: Simulation results for a single run. *Left:* Target position estimation error. *Right:* Uncertainty level (covariance matrix determinant). The x-axis in both plots is the iteration number k.

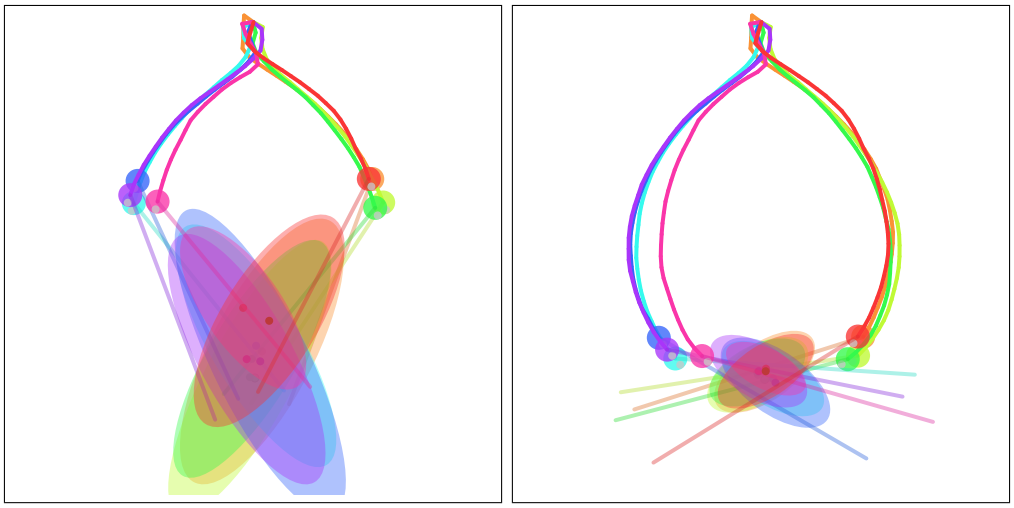

Figure 2.7: Two stages of several experiment realizations showing common resulting trajectories.

2.6.2.1 Analytic Solution

For comparison purposes, an analytic solution for this first approach (using EKF with $N = 1$) was obtained and tested. See appendix A, for a derivation of the analytic expression. As

depicted in figure 2.8, the resulting trajectory has similar tendencies as the previous numerical simulations. The use of this analytic solution is limited to the unconstrained, simplest form of the problem.

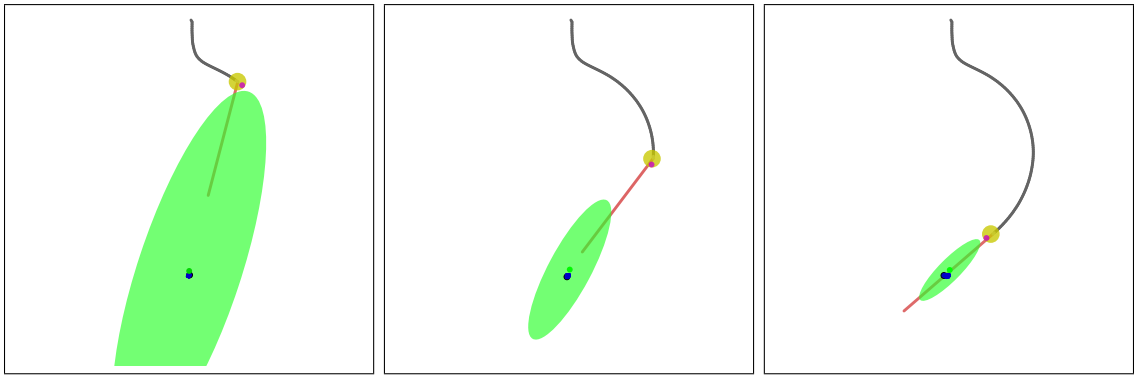

Figure 2.8: Simulation results using an analytic solution for the optimization scheme for horizon 1 (see Appendix A), using the covariance matrix determinant as the objective function only.

2.6.2.2 Horizon effects

Carried out simulations, shown in figure 2.9, indicate that for larger horizon sizes, the planner tendency is to get closer to the target in a faster way. This is congruent with theory and common sense, because with a faster approaching, the uncertainty due the angular error in observation will reduce faster.

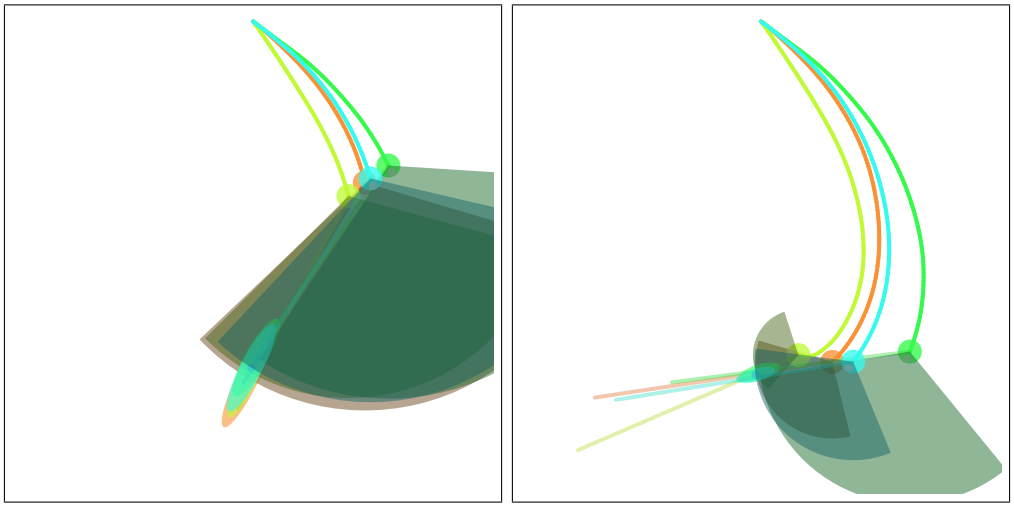

Figure 2.9: Simulation results for different horizon sizes. In increasing order the colors are: green, cyan, orange and yellow, for 1, 6, 11, 16 horizon sizes respectively.

2.6.3 Unscented Kalman Filter

The implementation of the UKF version of the planner was straight forward using the developed framework. Actually, the framework was coded using a generic *java interface* for the recursive

filter object, which allows us to exchange the target filter, and even simulate different types of filters at the same time, for comparison purposes.

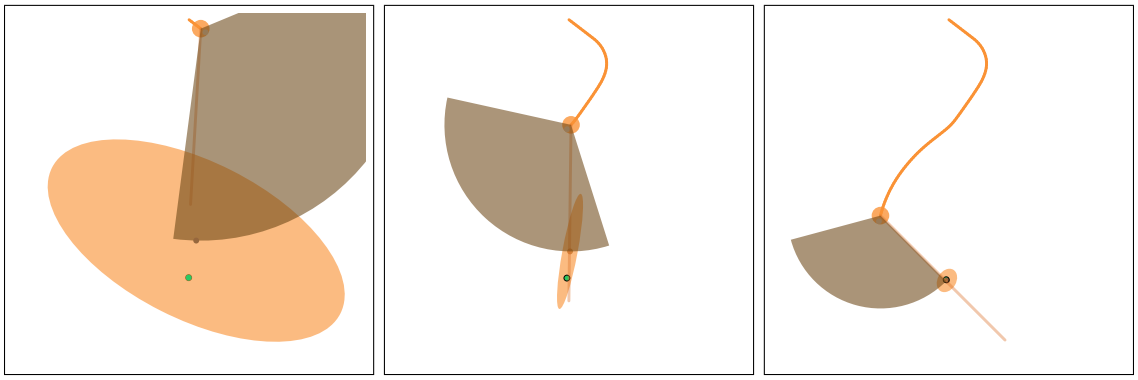

Figure 2.10: Robot trajectory and target position estimation and uncertainty ellipse using an UKF estimator, in different stages of the simulation ($k = \{5, 45, 85\}$).

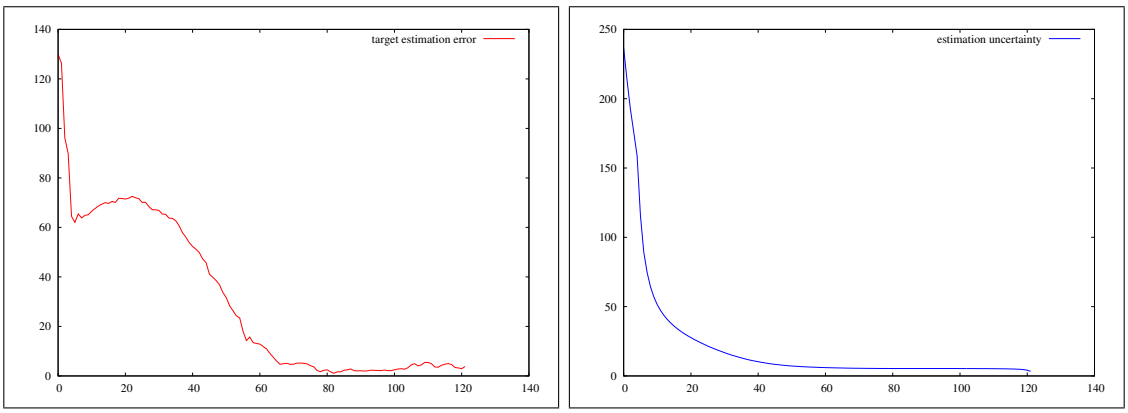

Figure 2.11: First simulation results for the UKF planner. *Left:* Target position estimation error. *Right:* Uncertainty level (covariance matrix determinant). The x-axis in both plots is the iteration number k.

In figure 2.10, the resulting trajectory and estimations for a first simulation are presented. Also in figure 2.11, the estimation errors and the uncertainty level are shown. The resulting trajectories are interesting, as most of them are S shaped. This behavior is attributed to the more accurate approximation of the filtered distributions and their moments far from the estimation mean, which might lead to different (and "better") optimal trajectories and controls. Actually, as seen in figure 2.11, the UKF planner obtains very low estimation errors and uncertainty levels. The disadvantage of this planner, is that it is slightly more complex in computational terms, because it implies $2n + 1$ evaluations of the non-linear functions, where n is the state dimension (5 in this case), this for the distribution estimations.

In comparison with the trajectory generated by the EKF planner, the ones of the UKF planner are more sinuous. Also, the estimation error and the uncertainty are lower at the end.

3

Particle Filter Based Methods

3.1 Gaussianity Limitations

The use of Gaussian models has the major disadvantage of being unable to handle multi-modal distributions. But also, there are other aspects that make the Gaussian models not well suited for several real-life applications. For instance, Gaussian distributions have support in all $x \in \mathbb{R}^n$, which is not true for some noise sources (i.e. real sensors have usually a bounded operation range). Also, Normal distributions are symmetric, and again, this does not always hold for real sensor noise distributions. So in order to overcome these limitations, it is necessary to use different kind of filters. One of the most common of these filters is the *Particle Filter*, which does not make any assumption about the shape or any other aspects regarding the involved distributions. Also, this filter handles very well the non-linearity of the underlying systems. So in this chapter, a Particle Filter version of the control scheme is studied.

3.2 Non-Gaussian Noise Models

As stated above, real systems are not always Gaussian, some of them might be approximately Gaussian, others not quite. In fact, most sensors have characteristic noise distributions. These distributions do not have to be symmetrical, or unimodal, not even smooth. They might be characterized by an empirical distribution obtained by experimentation.

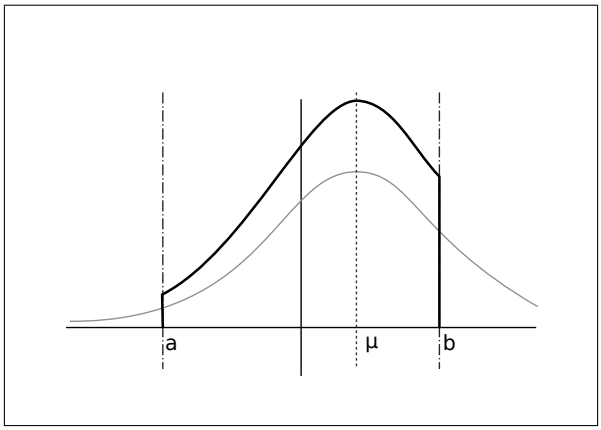

Figure 3.1: Truncated gaussian distribution density function. The parameters a and b are the minimum and maximum limits, respectively. μ is the mean value of the corresponding normal distribution.

3.2.1 Truncated Gaussian Distribution

A simple step to improve the modeling of a sensor is to take in consideration its operation range. It might be possible to model it as a truncated normal distribution, which basically behaves as a normal distribution with parameters μ and σ inside an interval $[a, b]$, and is zero valued outside this interval, being normalized to preserve the total area of 1. This distribution might have very different lookings, depending on its parameters (including the limits a and b), being possibly very asymmetrical. Figure 3.1 shows the form of this distribution and equation 3.1 expresses its density function. In [Robert 1995] some properties of this distribution and some sampling methods are explained.

$$f(z) = \begin{cases} c^{-1}(2\pi\sigma^2)^{-\frac{1}{2}} \exp(-(z-\mu)^2/2\sigma^2), & a < z < b \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

where

$$c = \int_a^b (2\pi\sigma^2)^{-\frac{1}{2}} \exp(-(x-\mu)^2/2\sigma^2)dx. \tag{3.2}$$

3.3 Particle Filter Review

Particle Filter (PF) is a non-parametric version of the Bayes Filter. It makes use of a set of weighted point masses (i.e. particles) to represent a probability distribution (depicted in Figure 3.2 as brown circles). It applies Monte Carlo sampling techniques, like the ones exposed in [Hastings 1970], to recursively approximate the filter posterior distribution using a set of particles drawn from the posterior of the previous iteration. For a comprehensive review of particle filter techniques, [Arulampalam et al. 2002] can be consulted.

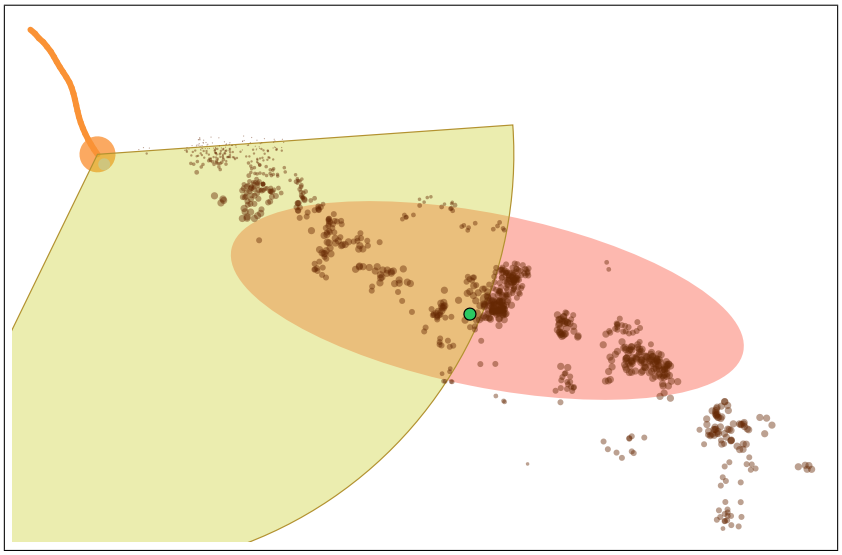

Figure 3.2: Particle Filter Simulation. A mobile robot (orange), estimating the position of a target (green). The filter particles (brown) and the uncertainty ellipse (red) are also shown. The size of the particles represent their weight.

3.3.1 Sequential Importance Sampling/Resampling

The Sequential Importance Resampling (SIR) Particle Filter is one of the most common variants of the PF [Arulampalam et al. 2002]. The SIR is an extension of the Sequential Importance Sampling method (SIS), which, basically, defines a method to handle the particle sampling and the weight computing. This method consists in generating a random sample to replace each particle, by using the transition model (defined in equations 2.1 to 2.3 in section 2.2) of the system, as follows:

$$X_k^i \sim P(X_k|X_{k-1} = X_{k-1}^i), \tag{3.3}$$

where the index $i \in [1 : M]$ indicates the particle number. M is the total quantity of particles. This is called the propagation or exploration step. Normally, it is easier to sample from this distribution, as the transition noise model is assumed to be known. After that, this new particle set does not reflect the actual belief distribution, so the weights of the particles have to be adjusted. This is accomplished by the following recursive equation:

$$w_k^i = \eta P(Y_k|X_k^i) w_{k-1}^i, \tag{3.4}$$

where w_k^i is the current weight of the i-th particle and η is a normalization factor given by:

$$\eta = \left[\sum_{j=1}^{M} P(Y_k|X_k^j) w_{k-1}^j \right]^{-1}. \tag{3.5}$$

These previous equations constitute the application of the *Importance Sampling* technique, very common in Monte Carlo methods [Hastings 1970]. Figure 3.3 gives a hint of how this

principle works. Also, in [Doucet et al. 2000], the author presents some alternative particle propagation distributions to reduce particle degeneration (i.e. $w_i = 0$), which is a problem inherent on this approach.

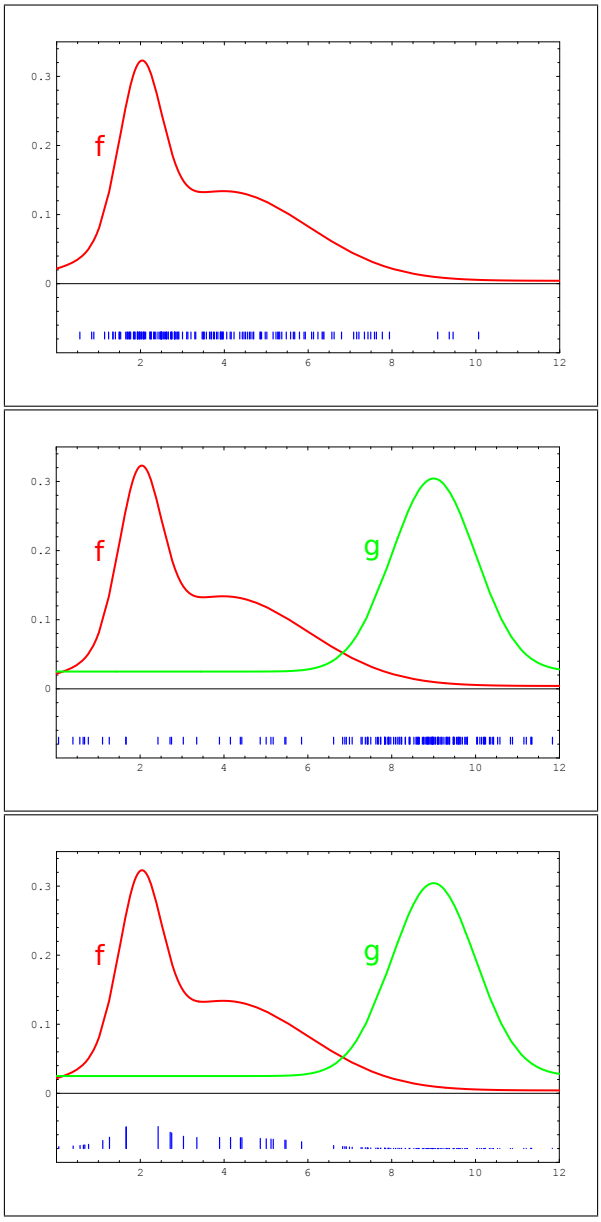

Figure 3.3: Importance Sampling principle. *Top:* Actual distribution f to be aproximated. *Center:* Distribution g to sample from, and obtained samples. *Bottom:* Final weighted samples representing the distribution f. In the case of the PF methods f corresponds to the belief distribution. [From Probabilistic Robotics, MIT Press].

After the weighting step, a state estimation can finally be obtained, and it is simply a weighted average of the particles, as follows:

$$\hat{X}_k = \frac{1}{M} \sum_{j=1}^{M} w_k^j X_k^j \tag{3.6}$$

Now, in order to avoid this particle degeneration problem (i.e. the tendency of weights to jointly tend to zero), an extension of the SIS approach is used, this makes use of an additional step, called *resampling*. This new approach is the SIR variant of the Particle Filter.

The resampling step consists on applying one more time the Monte Carlo principle to replace the current particle set. The sampling will be out of the original particle set (a discrete sampling) weighted by the weights of the particles. The advantage of this approach is that the new particles will have all the same weight (so degeneration is avoided), and only the particles with higher weights will survive. However, this also increase complexity, and restrict possible paralellizing schemes. On the other hand, resampling does not have to be done in every iteration. Some criteria can be defined to perform the resampling.

3.4 Control Scheme with PF

The following is a version of the control scheme used in the previous chapter, adapted for the use with the particle filter and non-additive noise distributions. A straight forward adaptation could have been used, by directly computing an estimation of the covariance matrix and use it as the cost function. But the problem with this is that because of the nature of the filter and the underlying approximated distribution in the form of particles, this covariance estimation might not be a very accurate indicator of the actual uncertainty. So, an approach similar to the one proposed in [Skoglar et al. 2009] is used. This approach consists in simulating M Kalman Filters, one for each particle in a subset of the current particle set, using a covariance matrix $\Sigma_k^{(i)}$ given by the filter covariance estimator and scaled by the particle weight $w^{(i)}$. Then the weighted sum of the covariance matrix determinants is used as a cost function. The author in [Skoglar et al. 2009] proposes a random simulation of the motion and observation noises, and a random selection of the particle subset.

$$\min_{U_{N-1}} \; L(X_N) = \sum_{i=1}^{M} \det \Sigma_N^{(i)} \tag{3.7}$$

$$\begin{aligned} s.t. \qquad & u_k \in \mathcal{U} \\ & X_0^{(i)} \sim P(X_0|I_0) \\ & X^{t(i)}_{k+1} = f^t(X^{t(i)}_k, \mathcal{W}^{t(i)}) \\ & \mathcal{W}^{t(i)} \sim \mathcal{D}^t(X_k^{(i)}) \\ & X^r_{k+1} = f^r(X^r_k, u_k) \\ & {\Sigma_{k+1}}^{(i)} = h({\Sigma_k}^{(i)}, {Y_k}^{(i)}) \\ & {Y_k}^{(i)} = g({X_k}^{(i)}, \mathcal{W}^{y(i)}) \\ & \mathcal{W}^{y(i)} \sim \mathcal{D}^y(X_k^{(i)}) \\ & {\Sigma_0}^{(i)} = w^{(i)}\Sigma_0 \end{aligned}$$

where $\mathcal{D}^t(.)$ and $\mathcal{D}^y(.)$ are non-additive generic noise probability distributions for the target motion and the sensor observation respectively. $h(.)$ includes the EKF correction equation (i.e.,

equation 2.17) that depends on the current observation Y_k. All the terms in this optimization scheme are explained with more detail in table 3.1.

Term	Description
$X_0^{(i)} \sim P(X_0 \mid I_0)$	The initial state for every simulated KF is obtained from the current estimation distribution (I_0 represents the initial information), it means that a particle can be used as sample.
$X^{t(i)}_{k+1} = f^t(X^{t(i)}_k, \mathcal{W}^{t(i)})$	The following target states are going to be simulated using the transition equations including the transition noise.
$\mathcal{W}^{t(i)} \sim \mathcal{D}^t(X_k^{(i)})$	How the transition noise is generated from an arbitrary distribution.
$X^r_{k+1} = f^r(X^r_k, u_k)$	The following robot states are going to be computed with the transition equation using the specific control u_k, which is the variable to be minimized.
${\Sigma_{k+1}}^{(i)} = h({\Sigma_k}^{(i)}, {Y_k}^{(i)})$	The covariance matrix has to be computed from the EKF equations.
${Y_k}^{(i)} = g({X_k}^{(i)}, \mathcal{W}^{y(i)})$	The observation ${Y_k}^{(i)}$ simulated from the observation model including a random noise $\mathcal{W}^{y(i)}$.
$\mathcal{W}^{y(i)} \sim \mathcal{D}^y(X_k^{(i)})$	The observation random noise generated from an arbitrary noise distribution.
${\Sigma_0}^{(i)} = w^{(i)}\Sigma_0$	Initial covariance matrix for each simulated EKF is initialized as the initial estimated covariance Σ_0 scaled by the particle weight.

Table 3.1: Optimization Scheme Terms.

3.5 Simulations and Results

The simulations were performed with the same software application as in the previous chapters. We simply adapted the cost function, and added an implementation of the Particle Filter, to carry out the new scheme. Also the visualization of the particles was added. Additionally, a simple parallelization scheme was implemented, taking advantage of the structure of the problem expressed in equation 3.7. It can be deduced that for every step, the particle motion is independent, so, the simulation of all the particles can be performed in parallel.

3.5.1 First Simulations

The first set of simulations consisted on the same type of setup as in the previous chapters. Similar conditions were used, such as the restriction and regularization terms, and similar noise parameters. It turned out that several types of resulting trajectories were observed (some examples are depicted in figure 3.5), some of them similar to the EKF patterns, other more resemblant to the paths obtained from the UKF planner. Figure 3.4 shows some indicators of the planner performance for one of the experiments. The performance levels are similar to the ones in the EKF version, but with more variation from one iteration to another.

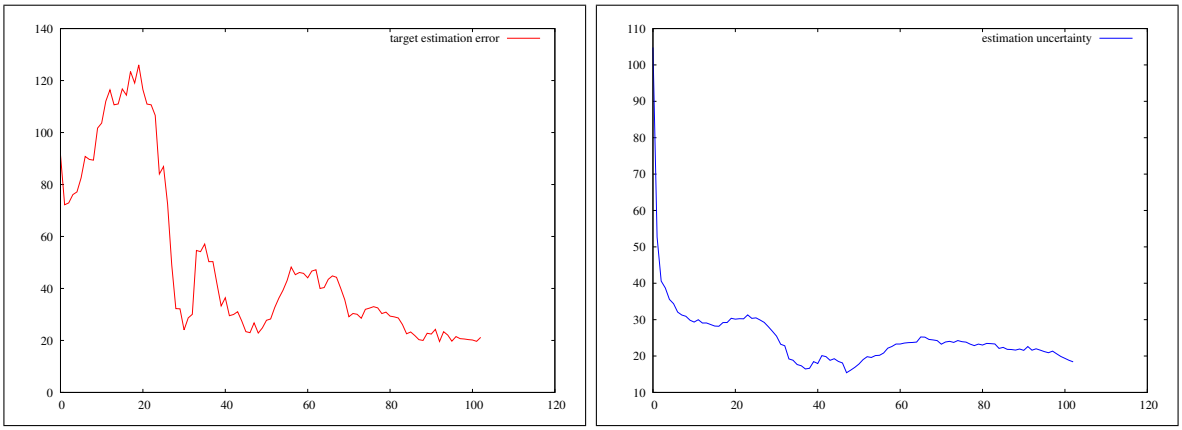

Figure 3.4: Results of a representative sample run of simulation. *Left:* Target position estimation error. *Right:* Uncertainty level (estimated covariance matrix determinant). The x-axis in both plots is the iteration number k.

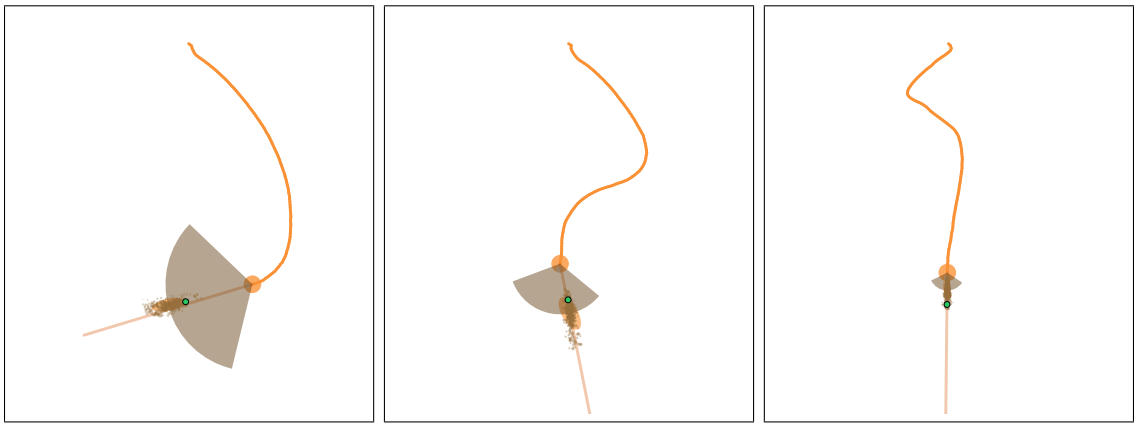

Figure 3.5: Different resulting trajectories, depending on the actual noise realizations. In all the experiments 400 particles and a horizon length of 2 were used. This approach presented a wider range of trajectory types. Some of these trajectories are similar to the ones of the EKF planner, others are more resemblant to the ones of the UKF planner.

3.5.2 Comparison of PF vs KF

In this section, some tests were performed to compare the performance of both methods presented so far. Both planners were executed simultaneously in the simulator, using the same noise parameters. The resulting trajectories for one of these experiments are shown in figure 3.6. Results basically show that the PF approach is more reactive to the actual observations, and due the randomness of the particles, the form of the obtained paths are more variable. Also, at the beginning it seems to be more conservative, but it actually seems to have a better estimation on the long run. This behavior is attributed to the fact that as the estimation improves, more of the particles agree on the actual system state, which causes a stronger response of the controller to certain policy.

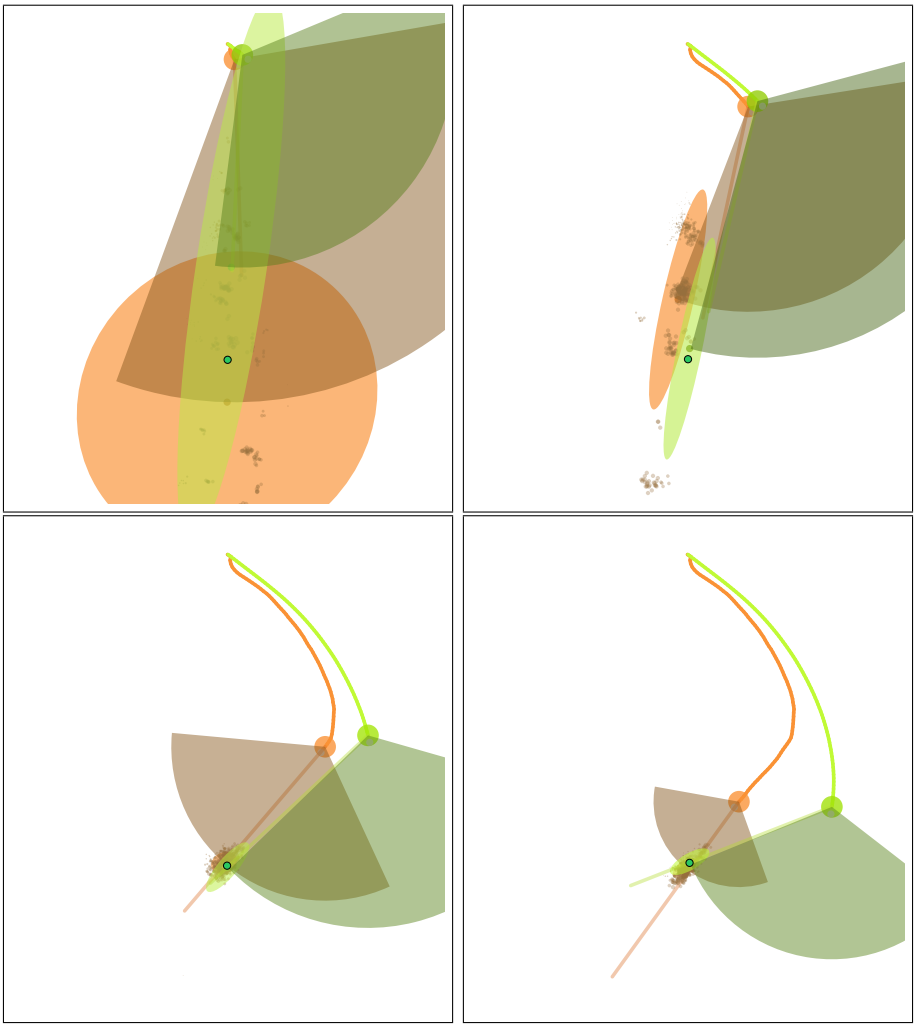

Figure 3.6: Simulation results for both PF (orange) and EKF (light green) planners. In this experiment the PF used 400 particles and the horizon length was 2. This simulation showed that the PF planner approached faster to the target and had a better estimation.

3.5.3 Number of Particles

An important parameter that affects directly in the overall planner and filter performance is the number of particles used in the filter. For instance, a higher number of particles, generates a smoother output trajectory. In figures 3.7 and 3.8, two simulations with different number of particles are shown. In these figures, it is observed how the trajectory is affected.

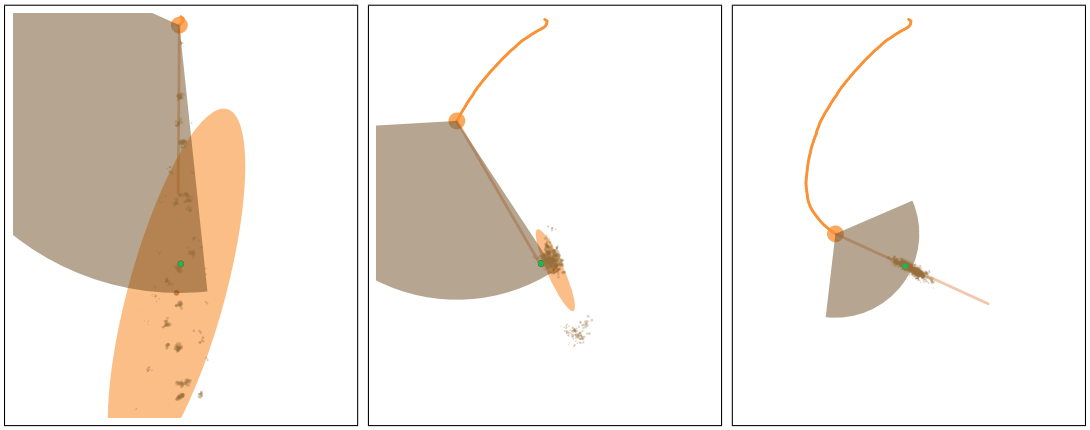

Figure 3.7: Particle Filter simulation results with 500 particles.

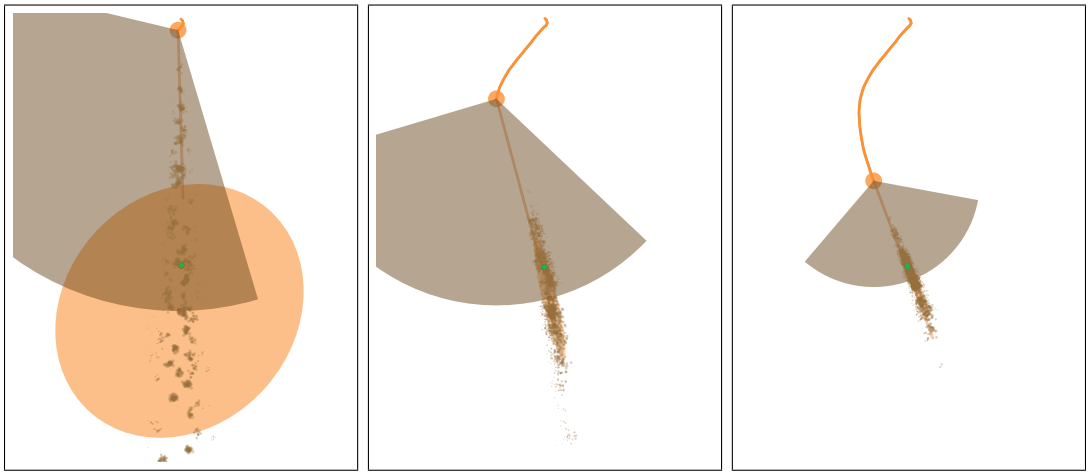

Figure 3.8: Particle Filter simulation results with 1500 particles.

Theoretically more particles should generate better results, but, it was observed that too many particles cause the filter to bias the estimated target position. This is caused by the triangular form of the observation noise distribution in the plane, which causes that more particles are drawn in the far side of the observation beam. These particles then cause a dragging of the estimation to this far side. This should be fixed by modifying the observation model to consider a maximum and minimum detection distance of the sensor. This is left as future work.

4

Application to Simultaneous Localization and Reconstruction of 3D Objects

4.1 Introduction

In this chapter, an original real-life application inspired in this present work is exposed. The application consists in the visual reconstruction (with monocular vision) and localization of an object, whose position and dimensions are unknown at the beginning. This is a more realistic setup than e.g., just assuming that the object position is initially known.

In order to complete the reconstruction task, it is required to have an estimation of the object position first, and then, to acquire images of the object from different angles. Also, an estimation of the object dimensions is necessary to make sure the contour of the object fits into the image. The realization of this real application implied developments in several technical aspects, including an adaptation of the current control scheme, the implementation of several image processing algorithms and the design and integration of a simple software architecture, for the interaction of all the system components. The reconstruction is performed by a NAO humanoid robotic platform, using a differential drive simplified locomotion controller.

4.2 Problem Description

In order to carry out this task, several practical issues have to be considered. For instance, besides the target bearing measurement, an additional object feature has to be obtained as an observation from the acquired images. This additional observation is the aperture angle (see figure 4.1) of the object projection in the camera. This supplementary observation is used to

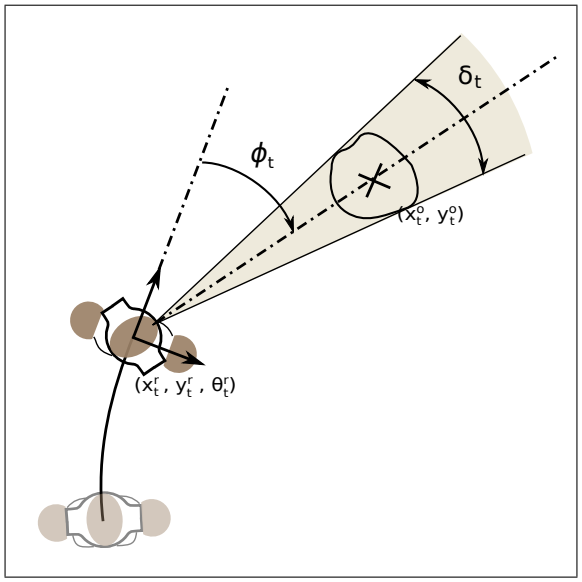

Figure 4.1: Bird-view of the reconstruction system. The location of the object to reconstruct is unknown at (x_0, y_0). Our aim is to control the robot so that the reconstruction of the object (which is initially not well localized) can be done with precision.

estimate the dimension of the object, which is approximated to a disk, and its radius (initially not known!) gives us a scale indicator. Also, as the approach assumes that the robot position is always known (deterministic), an additional visual localization system has to be used to accurately estimate the current robot position, which has to be updated in the state vector.

There are several points that need to be considered in order to perform an acceptable reconstruction. Among them, there are the following:

- The object has to be always on sight. If not, it has to be found.
- The whole object silhouette has to be always contained inside the image.
- The object must be seen from most of the possible angles.
- The images should be taken as still as possible, to avoid distorsion.

4.3 Reconstruction Oriented Control Scheme

To accomplish the previous points, several changes were made to the original modeling and to the control scheme. First, the additional angle aperture measurement δ^t was included in the previous observation model, extending the original observation Y_k to a two-dimensional vector. Also, the object (target) state X_k^t is augmented to include the radius, therefore passing from a punctual modelation to a disk-like model for the object. Now, on the control scheme side, the selected scheme was the one presented in chapter 2, using an EKF for the estimation and a couple of additional cost function terms, one to ensure the complete coverage of the object perimeter for its reconstruction, and the other to mantain a "safe" distance from the object, to avoid collisions and to make sure the whole object is going to be visible in the images.

4.3.1 Extended Model

Now, for convenience and clarity purposes, all the model equations are restated in this section.

- State Vector
  - robot state:
$$X_k^r = [x^r, y^r, \theta^r]' \tag{4.1}$$
  - target state:
$$X_k^t = [x^t, y^t, r^t]' \tag{4.2}$$
  - joint state:
$$X_k = \begin{bmatrix} X_k^t \\ X_k^r \end{bmatrix} \tag{4.3}$$
    where r^t is the target radius.
- Transition Model
  - *target:*
$$X_{k+1}^t = f^t(X_k^t, \mathcal{W}_k^t) = X_k^t + \mathcal{W}_k^t \Delta T \tag{4.4}$$
    where $\mathcal{W}_k$ is a random vector of the same dimension as X^t, and distributed according to $\mathcal{N}(0, \Sigma^t)$. And ΔT is the sampling time.
  - *robot:*
$$X_{k+1}^r = f^r(X_k^r, \omega_k) = X_k^r + \begin{bmatrix} v\cos(\theta^r + \omega_k \Delta T)\Delta T \\ v\sin(\theta^r + \omega_k \Delta T)\Delta T \\ \omega_k \Delta T \end{bmatrix} \tag{4.5}$$
    where v is the linear speed (constant). Again, this means that the robot motion is supposed deterministic.
- Observation Model
$$Y_k = g(X_k, \mathcal{W}_k^y) = \begin{bmatrix} \arctan_2(y^t - y^r, x^t - x^r) - \theta^r + \mathcal{W}_k^\theta \\ \arctan_2(r^t, \sqrt{(y^t - y^r)^2 + (x^t - x^r)^2}) + \mathcal{W}_k^r \end{bmatrix} \tag{4.6}$$
  where $\arctan_2(y, x)$ is a function that returns the arc tangent of the quotient $\frac{y}{x}$, also using the signs of x and y to determine the quadrant of the result. And $\mathcal{W}_k^y \sim \mathcal{N}(0, \Sigma^y)$ is a normal additive random noise vector composed of two elements $\mathcal{W}_k^y = [\mathcal{W}_k^\theta, \mathcal{W}_k^r]'$.

4.3.2 Added Cost Function Terms

In order to make the robot turn around the object to get a complete reconstruction, a new term was added. This new term τ_c is the number of angular sectors (bins) N_c of a discretised circle

around the object seen by the robot during the whole trajectory (as depicted in figure 4.2), divided by the total number of sectors N_0:

$$\tau_c = \lambda_c \frac{N_c}{N_0} \tag{4.7}$$

where λ_c is a constant scale parameter.

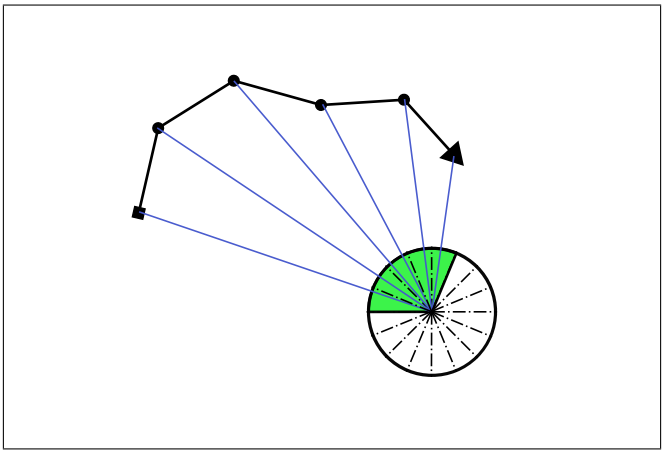

Figure 4.2: Coverage proportion cost function term. An imaginary circle is drawn arround the target, and it is divided into N_c angular sectors. The green area represents the angular sectors already seen by the robot during its trajectory. This is implemented by computing the angle of the rays drawn from the circle center to each trajectory point and checking in which angular sector it falls.

The other term τ_d added to the cost function was a repulsive potential inversely proportional to the squared distance from the robot to the object (see equation 4.8). Actually, this term is only activated when the distance goes below some threshold. In figure 4.3 simulations with two different min distance thresholds are shown. The observed effect is that the robot never goes closer than this distance to the target. This safe distance could be adapted in accordance to the estimated object radius, so it is ensured that the object will always fit into the image.

$$\tau_d = \lambda_d[(x^r - x^t)^2 + (y^r - y^t)^2]^{-1} \tag{4.8}$$

where λ_d is an adjustable scale parameter.

4.4 Robot Localization and Object Reconstruction

The Visual Localization and the object reconstruction methods used here, are complementary and independent of this work, and could be replaced by any other similar or more sophisticated techniques. Anyhow, assuming an accurate camera localization is crucial for the task of 3D modeling from multiple views. In this sense, our specific setup is aimed at estimating correct camera poses while simplifying the correspondence problem, which is a common issue while trying to determine which landmarks from the map correspond to the detected ones.

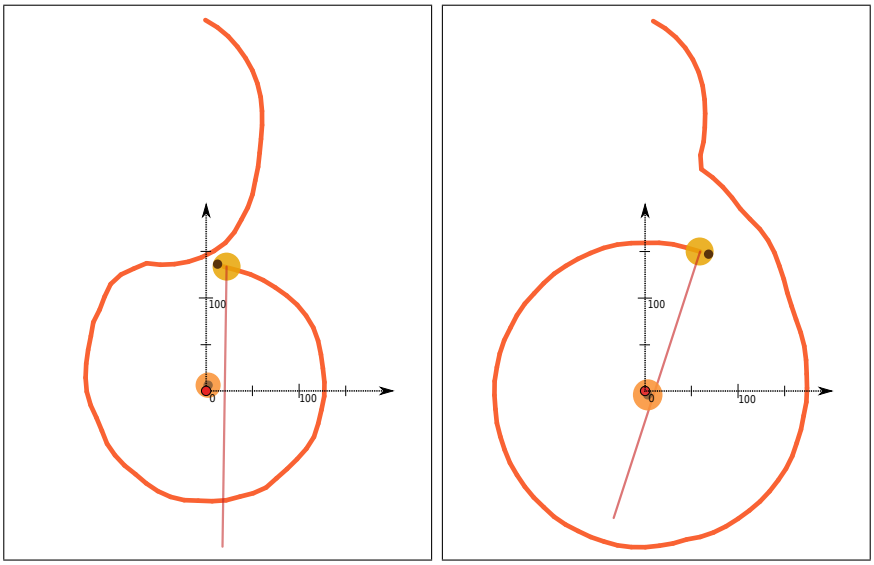

Figure 4.3: Simulations showing the trajectories generated for two different parameters for the safe distance cost function term. In these simulations the object is located in the coordinated system origin.

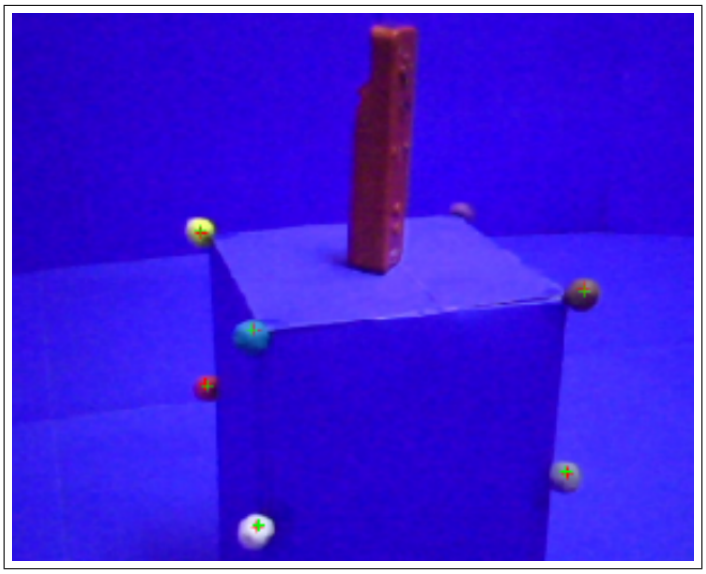

Figure 4.4: Visual localization. The visual localization system is based on the extraction of colored landmarks in the camera images. Crosses represent the centroid of the segmented colored blobs. Both the pixel positions of the centroids and their known 3D positions in the world feed the localization system.

4.4.1 Visual Localization

The localization method used in this application is based on the use of colored landmarks, detected in the robot images by simple color thresholding methods, where each landmark is identified by its color (see figure 4.4). Assuming the intrinsic parameters matrix K is available (it can be obtained with existing calibration tools), a standard method can be applied to minimize the landmarks reprojection error and estimate the camera localization pretty precisely. This is solved as a 2D-to-3D estimation problem using the *openCV* function *solvePnP*, which calculates

the camera pose (i.e. the camera extrinsic parameters) for each view of the object.

4.4.2 Space Carving Reconstruction

Regarding the 3D reconstruction of the object, a method based on shape from occluding boundaries known as space carving [Kutulakos and Seitz 2000] was used. Roughly, this method uses the camera projection matrix in order to reproject, towards the world, the area bounded by the silhouette of the observed object in the image. The camera projection matrix can be easily built as:

$$P = K[R_c|t_c], \tag{4.9}$$

where K is the matrix of intrinsic parameters of the camera. R_c is the rotation matrix and t_c is a translation vector that describe the camera configuration in the world.

From a set of multiple silhouettes with known camera matrices, a 3D model is finally recovered from the intersection of all reprojected silhouettes. This process, which resembles sculpting (carving), requires several views of the object usually acquired while the object is surrounded with the camera along a circular path. An initial bounding box is setted as parameter, but with the radius estimation, an adaptive bounding box could be computed, perhaps reducing the computing time of the reconstruction. This is not implemented yet, so it is left as future work.

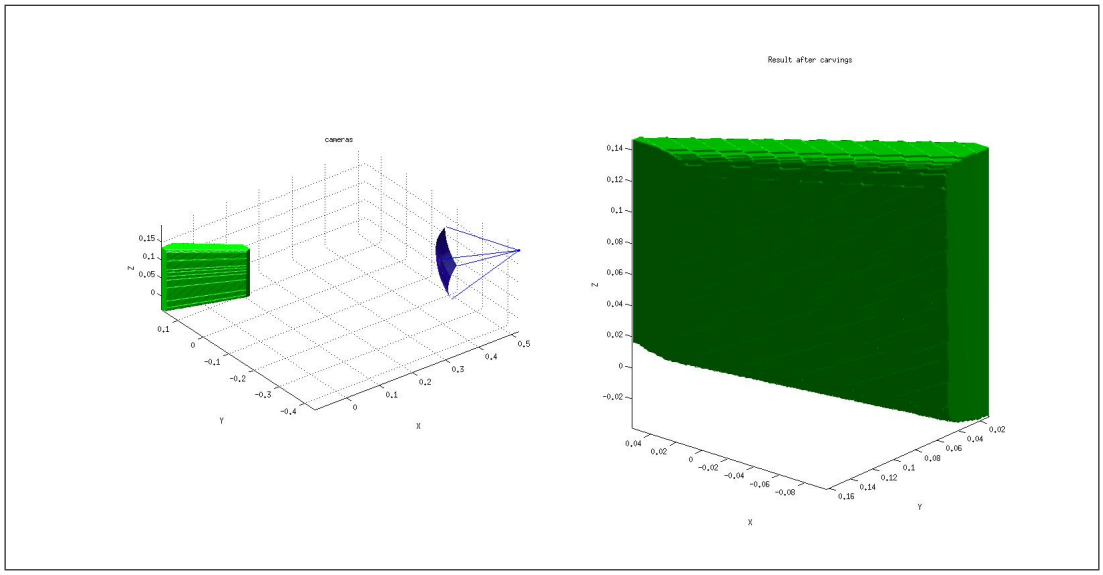

Figure 4.5: Space Carving first silhouette reprojection step. On the *left* the position from where the image was taken. On the *right* the generated solid shape. This first reconstruction is not accurate at all, and will be improved as more images are acquided and processed.

4.4.3 Extraction of Observation Vector Data

The actual extraction of the required observations is also an important task. To perform this process in a simple way, we suppose that the object can be segmented in the image from its background. As illustrated in Figure 4.6, by using standard color thresholding and morphological operations, we extract a blob corresponding to the object of interest. Then, along an horizontal axis, we project the blob centroid and the blob extremities, to get three points p_c (center) , p_l (left) and p_r (right). Then we generate the observations as:

$$Y_k = (\pm \arccos e_z^T K^{-T} K^{-1} p_r, \arccos p_l^T K^{-T} K^{-1} p_r)^T \tag{4.10}$$

where $e_z = (0, 0, 1)^T$, and the actual sign of the first element is determined by the horizontal position of the centroid. The $(.)^{-T}$ notation represents an inversion and transposition operations.

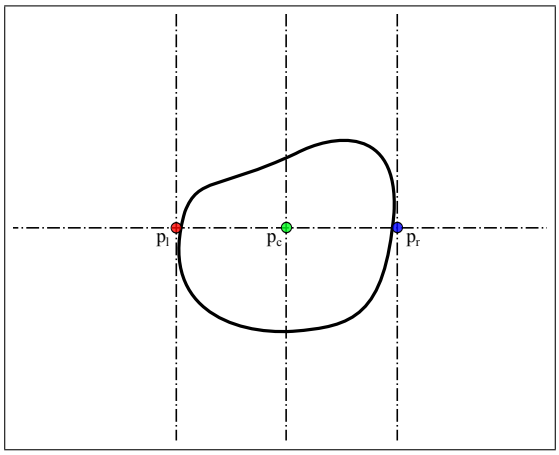

Figure 4.6: From image blobs to observations vector Y_t. Once a binary blob corresponding to the object has been segmented by applying color thresholding and morphological operations, the three points p_l, p_c and p_r are extracted to define the observation vector angles.

4.5 Implementation and Experiments Setup

The experiments were carried out onboard a Nao humanoid robotic platform. This robot is almost 0.6 meters tall, weights around 4.5 kilograms and has an intel atom @ 1.6 GHz processor, with a built-in *linux* OS, and a versatile multi language API (C++, Python, Java, MATLAB, Urbi, C, .Net). The Software Architecture uses a distributed server-host based scheme, and new custom functionalities can be added as modules (shared libraries). For this implementation, an image processing module to extract the required features was designed. We also implemented a simple position-based visual servoing approach to keep the target at the center of the image, in the same module. This module is accessed by an off-board modified version of the java framework (see section 2.6.1), which afterwards computes the optimal control sequence and sends it back to the robot.

The performed experiments consisted in the complete 3D reconstruction of three test

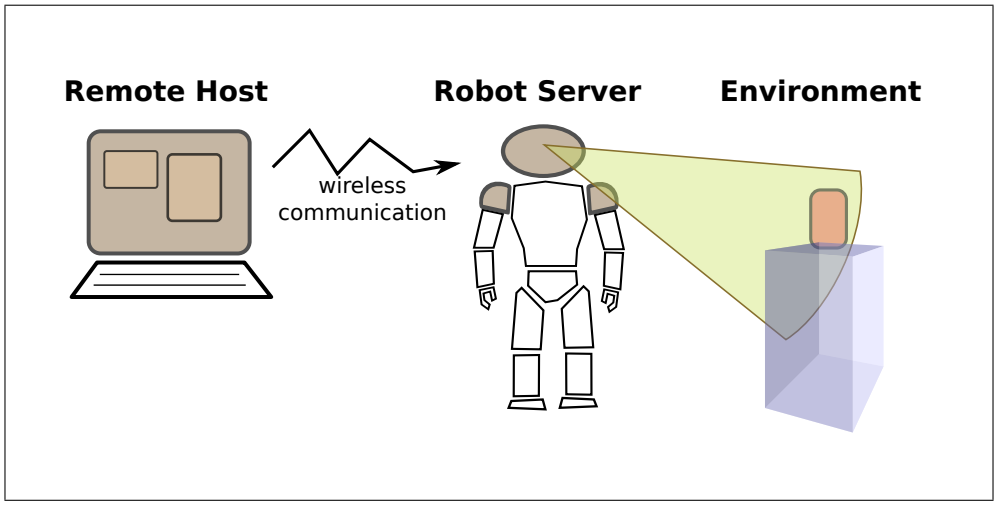

Figure 4.7: Experiment setup diagram. The robot platform (center) receives commands from the remote host (left) through a wireless link. Then the robot returns processed perception data from the environment. After that, the remote host computes the next control and sends it again to the robot.

objects by the robot, with the control computed on-line, off-board the robot using the actual measurements obtained by the segmentation module on-board the robot (see figure 4.7). The results were the generated trajectories, the obtained 3D models of the objects, and the estimation error plots.

4.6 Results

The experimental results are now presented. They are divided in three groups. The first set is obtained reconstructions, depicted in figure 4.8. The figure presents, for each object, five different views of the different objects as recorded by the camera of the robot, the extracted silhouette and a rendered view of the final 3D object reconstruction for the purposes of visual comparison. The next group is given by the estimated camera poses, and therefore, robot poses. They are displayed in figure 4.9. The parallelepiped represents the object base position ground truth (and which has to be estimated by the robot), and the taken snapshots appear as blue cones.

The last set of results is formed by the estimation errors (for the position and for the radius) for several trials of the experiment for one of the objects, with the purpose of showing the success and repeatability for different trials. First, the positions of the center of the object on the (x, y)-plane, as estimated by the active robot control, were recorded along the circular acquisition paths performed by the robot on each trial. This measure was compared against the final position of the center of the object obtained from the 3D reconstruction. The Euclidean distance between these two measures is shown as a function of the robot position on the left part of the figure 4.10. Note how initial instabilities present at the beginning of the walk tend to diminish as the path is completed. The plot on the right depicts a similar experiment considering the width of the object.

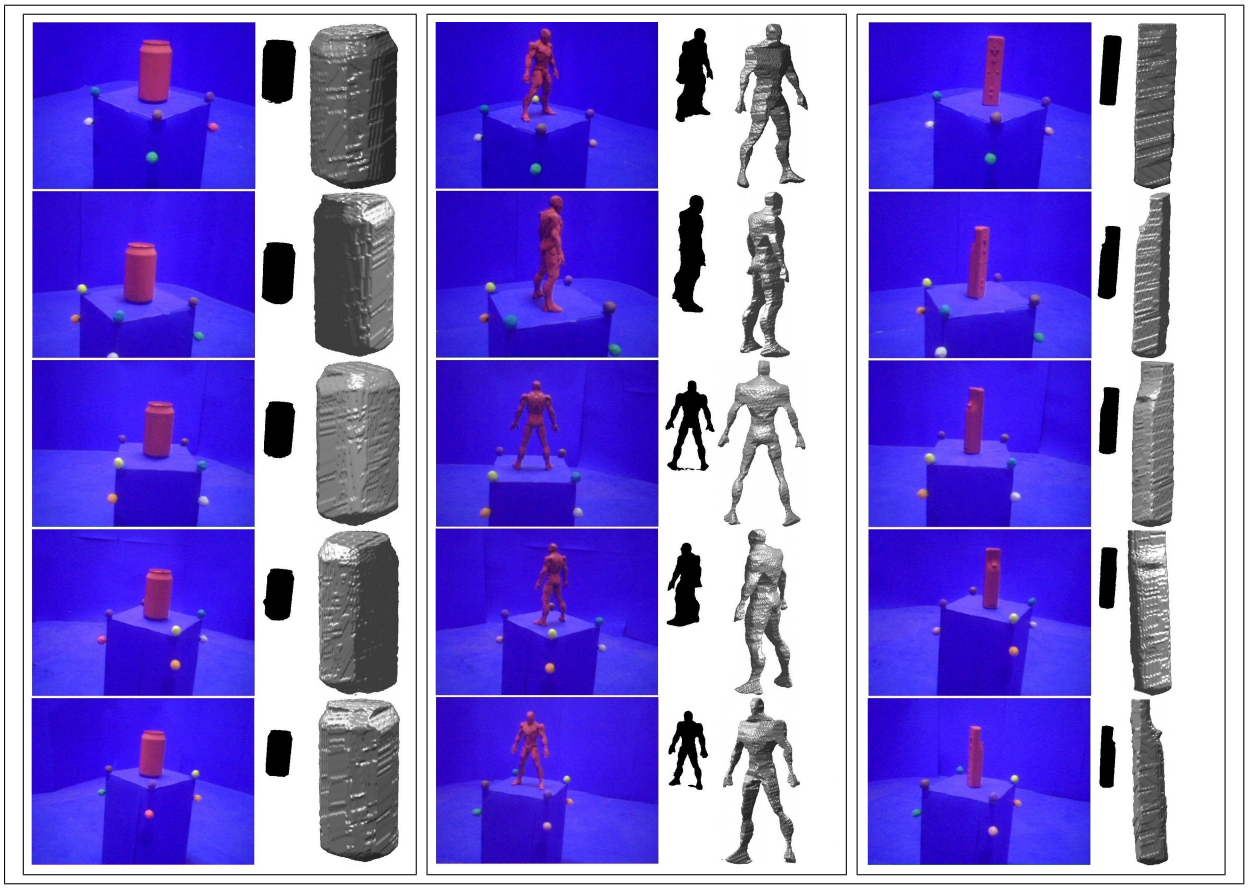

Figure 4.8: *3D reconstruction results.* The figure contains reconstruction results for three objects: an action figure (a), a remote control (b) and a can (c). For each object, example views are provided row-wise. The different views correspond to input images (as seen by the camera of the robot), extracted silhouettes (shown in black) and rendered views of the reconstructed 3D model. In the figure, only the silhouettes remain proportional in size with the input images, while the rendered views have been scaled for the purposes of visualization.

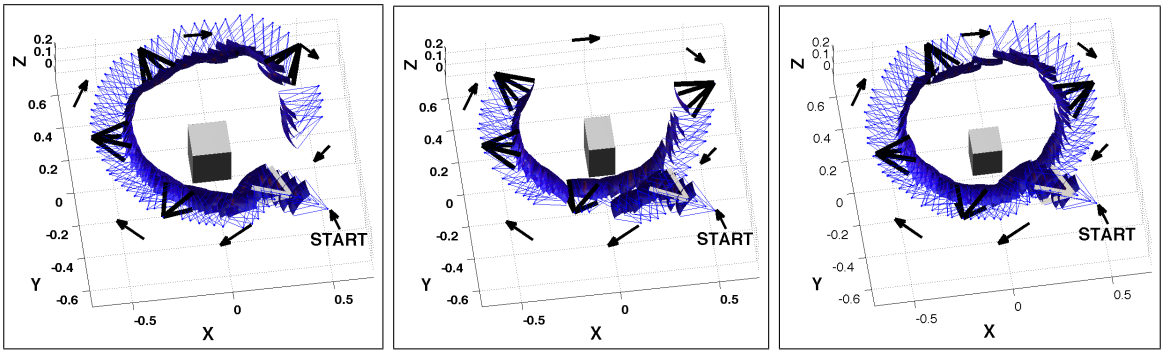

Figure 4.9: Generated trajectory and estimated camera positions for each reconstruction case. The viewpoints associated to the snapshots taken along these trajectories are depicted in blue. The five camera positions corresponding to the five views shown in Figure 4.8 have been highlighted in order to examplify the proximity of the camera to the object.

As it can be observed, the reconstructions are obtained with good quality, in particular in

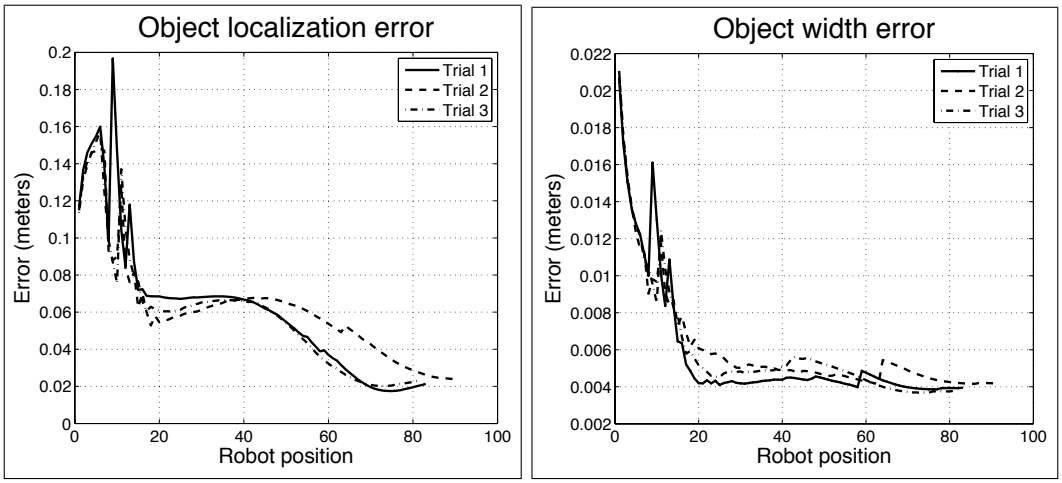

Figure 4.10: Error plots for object localization and object width. The two plots depict the error behavior for three reconstruction trials of the Wii remote control. The x-axis of the plots corresponds to the position of the robot (around the generated circular path). The plots reveal how both errors in object localization and object width diminish as the robot seems to finish surrounding the object.

the case of the action man figure (part *(a)* of the figure 4.8), where filiform parts (arms, legs) are well reconstructed, suggesting a successful coupling of the proposed robot control with the space carving methodology. Also, the obtained trajectories are quite similar to the simulation case, as the robot first makes position uncertainties lower by first getting closer to the target (for constant errors on the bearing angle, getting closer makes errors on positioning smaller) and by searching for positions favoring triangulation (hence, beginning to round up the object). Then, the aforementioned terms (section 4.3.2) make the robot complete the reconstruction objectives. Regarding the estimation error results, the position error ends up fluctuating around two centimeters, at the final of the experiment, this indicates an increasingly accurate estimation of the position of the object as the robot turns around it. The error converging towards four millimeters in the radius estimation reveals that not only the position of the object, but its scale, can also be estimated precisely as the camera of the robot acquires the relevant images for the 3D reconstruction method.

In general terms, the results for this application were very interesting and promising, primarily because of the level of success and the simplicity and flexibility of its implementation.

5

Conclusions and Future Work

In this work a sucessful implementation of an active tracking scheme for a bearing-only sensor system was developed. Several variants, using different types of state estimation filters (e.g. EKF, UKF), were studied. A couple of optimization schemes were tested and analysed. Also some simple non-gaussian system noise distributions were studied.

Several promising simulation and experimental results, including a real world application, are presented. The obtained simulation and experimental results include some plots of the overall system performance in terms of estimation error and uncertainty levels.

A flexible, quite generic, and useful software framework was developed, allowing extensive experimentation and testing of different system parameters and modalities. Additionally, actual interfacing with real experimental setups was also possible.

As it was shown, the approach is efficient in computer time terms, it was determined it is suitable for a variety of on-line applications. Even though the approach is suboptimal, as does not consider an infinite horizon, it can handle very well different scenarios with varying levels of uncertainty.

There is a lot of pending work for this interesting research field. Further experiments are required to determine the approach limitations and ultimate capabilities. Different types of system modelations (more complex) should be tested. In the final application, for instance, a more complex control model could be added. But it is definitely a very promising and vast research area.

References

Aidala, V. 1979. Kalman filter behavior in bearings-only tracking applications. *Aerospace and Electronic Systems, IEEE Transactions on AES-15,* 1 (January), 29–39.

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

Bar-Shalom, Y. and Sivan, R. 1969. On the optimal control of discrete-time linear systems with random parameters. *Automatic Control, IEEE Transactions on 14,* 1 (February), 3–8.

Bar-Shalom, Y. and Tse, E. 1974. Dual effect, certainty equivalence, and separation in stochastic control. *Automatic Control, IEEE Transactions on 19,* 5 (October), 494 – 500.

Doucet, A., Godsill, S., and Andrieu, C. 2000. On sequential monte carlo sampling methods for bayesian filtering. *Springer's Statistics and Computing 10,* 3, 197–208.

Fawcett, J. A. 1988. Effect of course maneuvers on bearings-only range estimation. *Acoustics, Speech and Signal Processing, IEEE Transactions on 36,* 8 (August), 1193 – 1199.

Feldbaum, A. A. 1965. *Optimal Control Systems*, Second ed. Vol. 2. New York: Academic Press.

Hastings, W. K. 1970. Monte carlo sampling methods using markov chains and their applications. *Oxford Journals' Biometrika 57,* 1 (January), 97–109.

Julier, S. J. and Uhlmann, J. K. 1997. A new extension of the kalman filter to nonlinear systems. *SPIE Proceedings - The International Society for Optical Engineering 3068,* 23, 182–193.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering ASME 82*, 35–45.

Kappen, H. J. 2007. An introduction to stochastic control theory, path integrals and reinforcement learning. In *COOPERATIVE BEHAVIOR IN NEURAL SYSTEMS: Ninth Granada Lectures*. American Institute of Physics, 149–181.

Kutulakos, K. N. and Seitz, S. M. 2000. A theory of shape by space carving. *International Journal of Computer Vision 38*, 199–218.

Logothetis, A., Isaksson, A., and Evans, R. J. 1997. An information theoretic approach to observer path design for bearings-only tracking. *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on 4*, 3132 – 3137.

Lovejoy, W. S. 1991. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research 28,* 1 (January), 47–65.

Martinez, L. and Soares, S. 2002. Comparison between closed-loop and partial open-loop feedback control policies in long term hydrothermal scheduling. *IEEE TRANSACTIONS ON POWER SYSTEMS 17,* 2 (May), 330–337.

Nelder, J. A. and Mead, R. 1965. A simplex method for function minimization. *The Computer Journal, Oxford Press 7,* 4, 308–313.

Nocedal, J. 2006. *Numerical Optimization*, Second ed. Operation Research and Financial Engineering Series. Springer.

Oshman, Y. and Davidson, P. 1999. Optimization of observer trajectories for bearings-only target localization. *Aerospace and Electronic Systems, IEEE Transactions on 35,* 3 (July), 892 – 902.

Ponda, S. S. 2008. Trajectory optimization for target localization using small unmanned aerial vehicles. M.S. thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press.

Reas, C. and Fry, B. 2010. *Getting Started with Processing.* O'Reilly.

Robbins, H. and Monro, S. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics 22,* 3 (September), 400–407.

Robert, C. 1995. Simulation of truncated normal variables. *Statistics and Computing*.

Singh, S. S., Kantas, N., Vo, B.-N., Doucet, A., and Evans, R. J. 2007. Simulation-based optimal sensor scheduling with application to observer trajectory planning. *Automatica, A Journal of IFAC, the International Federation of Automatic Control 43,* 5 (May), 817–830.

Skoglar, P., Orguner, U., and Gustafsson, F. 2009. On information measures based on particle mixture for optimal bearings-only tracking. In *Aerospace conference, 2009 IEEE.* IEEE, IEEE, 1–14.

Smallwood, R. D. and Sondik, E. J. 1973. The optimal control of partially observable markov processes over a finite horizon. *Operations Research, INFORMS 21,* 5 (September), 1071–1088.

Spall, J. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *Automatic Control, IEEE Transactions on 37,* 3 (March), 332 – 341.

Stansfield, R. G. 1947. Statistical theory of d.f. fixing. *Electrical Engineers - Part IIIA: Radiocommunication, Journal of the Institution of 94,* 15 (March), 762–770.

Thrun, S., Burgard, W., and Fox, D. 2005. *Probabilistic Robotics.* Intelligent robotics and autonomous agents. MIT Press.

Tokekar, P., Vander Hook, J., and Isler, V. 2011. Active target localization for bearing based robotic telemetry. In *Intelligent Robots and Systems (IROS), International Conference on.* IEEE, RSJ, 488 – 493.

Trémois, O. and Le Cadre, J. P. 1999. Optimal observer trajectory in bearings-only tracking for manouvering sources. *Radar, Sonar Navigation, IEE Proceedings 146,* 1 (February), 31–39.

UNBEHAUEN, H. 2000. Adaptive dual control systems: A survey. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000*. IEEE, AS-SPCC, 171 – 180.

A

EKF Analytic Solution Derivation

Sensor motion model, controlled by ω:

$$\begin{cases} x_t^s &= x_{t-1}^s + v\delta t \cos(\theta_{t-1}^s + \frac{1}{2}\omega\delta t) \\ y_t^s &= y_{t-1}^s + v\delta t \sin(\theta_{t-1}^s + \frac{1}{2}\omega\delta t) \\ \theta_t^s &= \theta_{t-1}^s + \omega\delta t \end{cases}$$

Target filtering on $\mathbf{X}_t$ through Kalman filtering:

$$\begin{cases} \bar{\mathbf{X}}_t^- &= \bar{\mathbf{X}}_{t-1}^+ \\ \mathbf{\Sigma}_t^- &= \mathbf{\Sigma}_{t-1}^+ \\ \mathbf{Y}_t &= h(\bar{\mathbf{X}}_t^-) \\ \bar{\mathbf{X}}_t^+ &= \bar{\mathbf{X}}_t^- \\ \mathbf{K}_t &= \mathbf{\Sigma}_t^- \mathbf{H}_t^T (\mathbf{H}_t \mathbf{\Sigma}_t^- \mathbf{H}_t^T + \sigma_M)^{-1} \\ \mathbf{\Sigma}_t^+ &= (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{\Sigma}_t^- \end{cases}$$

Observation model:

$$\begin{aligned} h(\mathbf{X}) &= \arctan\left(\frac{y_t - y_{t-1}^s - v\delta t \sin(\theta_{t-1}^s + \frac{1}{2}\omega\delta t)}{x_t - x_{t-1}^s - v\delta t \cos(\theta_{t-1}^s + \frac{1}{2}\omega\delta t)}\right) - \theta_{t-1}^s - \omega\delta t \\ &= \arctan\left(\frac{\Delta y}{\Delta x}\right) - \theta_{t-1}^s - \omega\delta t \end{aligned}$$

where $\Delta y = y_t - y_{t-1}^s - v\delta t \sin(\theta_{t-1}^s + \frac{1}{2}\omega\delta t)$ and Δx defined in a similar way. The corresponding Jacobian is given by:

$$\begin{aligned}\mathbf{H}_t &= \frac{1}{1+(\frac{\Delta y}{\Delta x})^2}[-\frac{\Delta y}{(\Delta x)^2}, \frac{1}{\Delta x}] \\ &= \frac{1}{(\Delta x)^2+(\Delta y)^2}[-\Delta y, \Delta x] \\ &= \frac{1}{\rho^2}[-\Delta y, \Delta x] \\ &= \frac{1}{\rho}[-\sin\mu, \cos\mu]\end{aligned}$$

where we define

$$\begin{cases} \rho^2 &= (\Delta x)^2 + (\Delta y)^2, \\ \mu &= \arctan(\frac{\Delta y}{\Delta x}). \end{cases}$$

By using the SVD of the covariance matrix

$$\mathbf{\Sigma}_t^- = \begin{pmatrix} v_{xx} & v_{xy} \\ v_{xy} & v_{yy} \end{pmatrix} = \begin{pmatrix} \cos\nu & -\sin\nu \\ \sin\nu & \cos\nu \end{pmatrix} \begin{pmatrix} \lambda_+ & 0 \\ 0 & \lambda_- \end{pmatrix} \begin{pmatrix} \cos\nu & \sin\nu \\ -\sin\nu & \cos\nu \end{pmatrix}$$

$$\mathbf{\Sigma}_t^- = \begin{pmatrix} \lambda_+\cos\nu & -\lambda_-\sin\nu \\ \lambda_+\sin\nu & \lambda_-\cos\nu \end{pmatrix} \begin{pmatrix} \cos\nu & \sin\nu \\ -\sin\nu & \cos\nu \end{pmatrix} = \begin{pmatrix} \lambda_+\cos^2\nu + \lambda_-\sin^2\nu & (\lambda_+ - \lambda_-)\sin\nu\cos\nu \\ (\lambda_+ - \lambda_-)\sin\nu\cos\nu & \lambda_+\sin^2\nu + \lambda_-\cos^2\nu \end{pmatrix}$$

which can be developed in a more compact way as

$$\mathbf{\Sigma}_t^- = \lambda_-\mathbf{I} + (\lambda_+ - \lambda_-)\begin{pmatrix} \cos^2\nu & \sin\nu\cos\nu \\ \sin\nu\cos\nu & \sin^2\nu \end{pmatrix}.$$

Let us examine the Kalman gain denominator (a scalar):

$$\begin{aligned} den &= \mathbf{H}_t\mathbf{\Sigma}_t^-\mathbf{H}_t^T + \sigma_M \\ &= \frac{1}{\rho^2}[-\sin\mu, \cos\mu]\begin{pmatrix} v_{xx} & v_{xy} \\ v_{xy} & v_{yy} \end{pmatrix}[-\sin\mu, \cos\mu]^T + \sigma_M \end{aligned}$$

and since

$$[-\sin\mu, \cos\mu]\begin{pmatrix} \cos\nu & -\sin\nu \\ \sin\nu & \cos\nu \end{pmatrix} = [\sin(\nu-\mu), \cos(\nu-\mu)],$$

then

$$den = \frac{1}{\rho^2}(\lambda_+\sin^2(\nu-\mu) + \lambda_-\cos^2(\nu-\mu)) + \sigma_M.$$

Note that since

$$\mathbf{\Sigma}_t^+ = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{\Sigma}_t^-$$

the determinant evolves as:

$$\det \mathbf{\Sigma}_t^+ = \det(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t) \det \mathbf{\Sigma}_t^-.$$

Let us note the decreasing factor of the covariance matrix as

$$\eta(\omega) = \det(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t),$$

It is a function of the control through the gain and Jacobian matrices. Let us also note

$$\mathbf{Q} = \mathbf{I} - \mathbf{K}_t\mathbf{H}_t.$$

Then it is clear that

$$\mathbf{Q} = \mathbf{I} - \frac{1}{den}\mathbf{\Sigma}_t^-\mathbf{H}_t^T\mathbf{H}_t.$$

By using the previous expressions

$$\mathbf{Q} = \mathbf{I} - \tfrac{1}{den}\tfrac{1}{\rho^2}\begin{pmatrix} v_{xx} & v_{xy} \\ v_{xy} & v_{yy} \end{pmatrix}\begin{pmatrix} \sin^2\mu & -\sin\mu\cos\mu \\ -\sin\mu\cos\mu & \cos^2\mu \end{pmatrix}.$$

$$\mathbf{Q} = \mathbf{I} - \tfrac{1}{den}\tfrac{1}{\rho^2}(\lambda_-\mathbf{I} + (\lambda_+ - \lambda_-)\begin{pmatrix} \cos^2\nu & \sin\nu\cos\nu \\ \sin\nu\cos\nu & \sin^2\nu \end{pmatrix})\begin{pmatrix} \sin^2\mu & -\sin\mu\cos\mu \\ -\sin\mu\cos\mu & \cos^2\mu \end{pmatrix}$$

By defining

$$D = (\lambda_+\sin^2(\nu-\mu) + \lambda_-\cos^2(\nu-\mu)) + \sigma_M\rho^2,$$

then

$$\begin{aligned}\mathbf{Q} = & \mathbf{I} - \tfrac{1}{D}\lambda_-\begin{pmatrix} \sin^2\mu & -\sin\mu\cos\mu \\ -\sin\mu\cos\mu & \cos^2\mu \end{pmatrix} \\ & -\tfrac{1}{D}(\lambda_+ - \lambda_-)\begin{pmatrix} \cos^2\nu & \sin\nu\cos\nu \\ \sin\nu\cos\nu & \sin^2\nu \end{pmatrix}\begin{pmatrix} \sin^2\mu & -\sin\mu\cos\mu \\ -\sin\mu\cos\mu & \cos^2\mu \end{pmatrix}\end{aligned}$$

$$\begin{aligned}\mathbf{Q} = & \mathbf{I} - \tfrac{1}{D}\lambda_-\begin{pmatrix} \sin^2\mu & -\sin\mu\cos\mu \\ -\sin\mu\cos\mu & \cos^2\mu \end{pmatrix} \\ & -\tfrac{1}{D}(\lambda_+ - \lambda_-)\begin{pmatrix} \cos\nu\sin\mu\sin(\mu-\nu) & -\cos\mu\cos\nu\sin(\mu-\nu) \\ \sin\mu\sin\nu\sin(\mu-\nu) & -\sin\nu\cos\mu\sin(\mu-\nu) \end{pmatrix}\end{aligned}$$

$$\mathbf{Q} = \tfrac{1}{D}(D\mathbf{I} - \lambda_-\begin{pmatrix} \sin^2\mu & -\sin\mu\cos\mu \\ -\sin\mu\cos\mu & \cos^2\mu \end{pmatrix} - (\lambda_+ - \lambda_-)\sin(\mu-\nu)\begin{pmatrix} \cos\nu\sin\mu & -\cos\mu\cos\nu \\ \sin\mu\sin\nu & -\sin\nu\cos\mu \end{pmatrix}).$$

Then:

$$\begin{aligned} D\mathbf{Q}_{12} &= +\lambda_- \cos\mu \sin\mu + (\lambda_+ - \lambda_-)\sin(\mu-\nu)\cos\mu\cos\nu \\ D\mathbf{Q}_{21} &= +\lambda_- \cos\mu \sin\mu - (\lambda_+ - \lambda_-)\sin(\mu-\nu)\sin\mu\sin\nu \end{aligned}$$

$$\begin{aligned} D\mathbf{Q}_{11} &= D - \lambda_- \sin^2\mu - (\lambda_+ - \lambda_-)\sin(\mu-\nu)\cos\nu\sin\mu \\ &= \lambda_+ \sin^2(\nu-\mu) + \lambda_- \cos^2(\nu-\mu) + \sigma_M\rho^2 - \lambda_- \sin^2\mu - (\lambda_+ - \lambda_-)\sin(\mu-\nu)\cos\nu\sin\mu \\ &= \lambda_+ \sin(\mu-\nu)(\sin(\mu-\nu) - \cos\nu\sin\mu) + \lambda_-(1 - \sin^2(\nu-\mu) - \sin^2\mu + \sin(\mu-\nu)\cos\nu\sin\mu) + \sigma_M \\ &= -\lambda_+ \sin(\mu-\nu)\sin(\nu)\cos\mu + \lambda_-(1 - \sin^2(\nu-\mu) - \sin^2\mu + \sin(\mu-\nu)\cos\nu\sin\mu) + \sigma_M\rho^2 \\ &= -\lambda_+ \sin(\mu-\nu)\sin(\nu)\cos\mu + \lambda_-(1 - \sin(\mu-\nu)(\sin(\mu-\nu) - \cos\nu\sin\mu) - \sin^2\mu) + \sigma_M\rho^2 \\ &= -\lambda_+ \sin(\mu-\nu)\sin(\nu)\cos\mu + \lambda_-(1 + \sin(\mu-\nu)\sin(\nu)\cos\mu - \sin^2\mu) + \sigma_M\rho^2 \\ &= -\lambda_+ \sin(\mu-\nu)\sin(\nu)\cos\mu + \lambda_-\cos\mu(\cos\mu + \sin\nu\sin(\mu-\nu)) + \sigma_M\rho^2 \\ &= \lambda_- \cos^2\mu - (\lambda_+ - \lambda_-)\sin\nu\cos\mu\sin(\mu-\nu) + \sigma_M\rho^2 \\ &= \cos\mu(\lambda_- \cos\mu - (\lambda_+ - \lambda_-)\sin\nu\sin(\mu-\nu)) + \sigma_M\rho^2 \\ &= \tfrac{\cos\mu}{\sin\mu} D\mathbf{Q}_{21} + \sigma_M\rho^2. \end{aligned}$$

Similarly,

$$\begin{aligned} D\mathbf{Q}_{22} &= D - \lambda_- \cos^2\mu + (\lambda_+ - \lambda_-)\sin(\mu-\nu)\cos\mu\sin\nu + \sigma_M\rho^2 \\ &= \lambda_+ \sin^2(\nu-\mu) + \lambda_- \cos^2(\nu-\mu) - \lambda_- \cos^2\mu + (\lambda_+ - \lambda_-)\sin(\mu-\nu)\cos\mu\sin\nu + \sigma_M\rho^2 \\ &= \lambda_+ \sin(\mu-\nu)\sin\mu\cos\nu + \lambda_-(1 - \sin^2(\mu-\nu) - \cos^2\mu - \sin(\mu-\nu)\cos\mu\sin\nu) + \sigma_M\rho^2 \\ &= \lambda_+ \sin(\mu-\nu)\sin\mu\cos\nu + \lambda_-\sin\mu(\sin\mu - \cos\nu\sin(\mu-\nu)) + \sigma_M\rho^2 \\ &= \sin\mu((\lambda_+ - \lambda_-)\sin(\mu-\nu)\cos\nu + \lambda_-\sin\mu) + \sigma_M\rho^2 \\ &= \tfrac{\sin\mu}{\cos\mu} D\mathbf{Q}_{12} + \sigma_M\rho^2 \end{aligned}$$

Then

$$\begin{aligned} D^2(\mathbf{Q}_{22}\mathbf{Q}_{11} - \mathbf{Q}_{12}\mathbf{Q}_{21}) &= (\tfrac{\cos\mu}{\sin\mu} D\mathbf{Q}_{21} + \sigma_M\rho^2)(\tfrac{\sin\mu}{\cos\mu} D\mathbf{Q}_{12} + \sigma_M\rho^2) - D^2\mathbf{Q}_{12}\mathbf{Q}_{21} \\ &= \sigma_M\rho^2(\tfrac{\cos\mu}{\sin\mu} D\mathbf{Q}_{21} + \tfrac{\sin\mu}{\cos\mu} D\mathbf{Q}_{12} + \sigma_M\rho^2) \\ &= \sigma_M\rho^2(\lambda_- \cos^2\mu - (\lambda_+ - \lambda_-)\sin(\mu-\nu)\cos\mu\sin\nu + \\ &\quad \lambda_- \sin^2\mu + (\lambda_+ - \lambda_-)\sin(\mu-\nu)\sin\mu\cos\nu) + \sigma_M\rho^2) \\ &= \sigma_M\rho^2(\lambda_- + (\lambda_+ - \lambda_-)\sin^2(\mu-\nu) + \sigma_M\rho^2) \\ &= \sigma_M\rho^2 D \end{aligned}$$

We deduce

$$\begin{aligned} \eta(\omega) = \det\mathbf{Q} &= \tfrac{\sigma_M\rho^2}{D} \\ &= \tfrac{\sigma_M\rho^2}{(\lambda_+ \sin^2(\nu-\mu) + \lambda_- \cos^2(\nu-\mu)) + \sigma_M\rho^2} \\ &= \tfrac{1}{1 + \frac{(\lambda_+ \sin^2(\nu-\mu) + \lambda_- \cos^2(\nu-\mu))}{\sigma_M\rho^2}} \end{aligned} .$$

Hence to minimize $\det\mathbf{Q}$, we want to **maximize**

$$\begin{aligned}\frac{(\lambda_+ \sin^2(\nu-\mu)+\lambda_- \cos^2(\nu-\mu))}{\sigma_M \rho^2} &= \frac{\lambda_+ -(\lambda_+ -\lambda_-)\cos^2(\mu-\nu)}{\sigma_M \rho^2} \\ &= f(\omega).\end{aligned}$$

Here we find what one can observe and guess intuitively: to decrease the error on the position of the target, one can either makes $\nu - \mu$ to $\frac{\pi}{2}$ and/or makes ρ tends to zero. We recall that ν is the angle of the direction of largest variance on the estimation, μ is the slope of the line joining the sensor to the target. Making the difference of these two directions tending to $\frac{\pi}{2}$ means that we look for a position with the most orthogonal direction to ν.

Both ρ and μ are controlled through ω. Let us denote $\Lambda = \lambda_+ - \lambda_-$, then

$$f(\omega) = \frac{\lambda_+ - \Lambda \cos^2(\mu(\omega) - \nu)}{\sigma_M \rho^2(\omega)}, \tag{A.1}$$

with

$$\mu(\omega) = \arctan\left(\frac{y_t - y^s_{t-1} - v\delta t \sin(\theta^s_{t-1} + \frac{1}{2}\omega\delta t)}{x_t - x^s_{t-1} - v\delta t \cos(\theta^s_{t-1} + \frac{1}{2}\omega\delta t)}\right).$$

The remarkable expression from Eq. A.1 can be transformed to make the control appear more clearly :

$$f(\omega) = \frac{\lambda_+ - \Lambda \frac{1}{1+(\frac{\tan\mu - \tan\nu}{1+\tan\mu\tan\nu})^2}}{\sigma_M \rho^2(\omega)},$$

$$\begin{aligned}f(\omega) &= \frac{\lambda_+ - \Lambda \frac{1}{1+(\frac{\Delta y\cos\nu - \Delta x\sin\nu}{\Delta x\cos\nu + \Delta y\sin\nu})^2}}{\sigma_M \rho^2(\omega)}, \\ &= \frac{1}{\sigma_M \rho^2(\omega)}\left(\lambda_+ - \Lambda \frac{(\Delta x\cos\nu + \Delta y\sin\nu)^2}{(\Delta x\cos\nu + \Delta y\sin\nu)^2 + (\Delta y\cos\nu - \Delta x\sin\nu)^2}\right) \\ &= \frac{1}{\sigma_M \rho^2(\omega)}\left(\lambda_+ - \Lambda \frac{(\Delta x\cos\nu + \Delta y\sin\nu)^2}{\rho^2(\omega)}\right. \\ &= \frac{1}{\sigma_M \rho^4(\omega)}(\lambda_+(\Delta x\sin\nu - \Delta y\cos\nu)^2 + \lambda_-(\Delta x\cos\nu + \Delta y\sin\nu)^2) \\ &= \frac{1}{\sigma_M \rho^4(\omega)}(\lambda_+((\Delta x_0 - v\delta t\cos(\theta^s_{t-1} + \tfrac{1}{2}\omega\delta t))\sin\nu - (\Delta y_0 - v\delta_t \sin(\theta^s_{t-1} + \tfrac{1}{2}\omega\delta t))\cos\nu)^2 \\ &\quad +\lambda_-((\Delta x_0 - v\delta t\cos(\theta^s_{t-1} + \tfrac{1}{2}\omega\delta t))\cos\nu + (\Delta y_0 - v\delta_t \sin(\theta^s_{t-1} + \tfrac{1}{2}\omega\delta t))\sin\nu)^2) \\ &= \frac{1}{\sigma_M \rho^4(\omega)}(\lambda_+(\rho_0\sin(\nu - \mu_0) + v\delta_t \sin(\theta^s_{t-1} + \tfrac{1}{2}\omega\delta t - \nu))^2 \\ &\quad +\lambda_-(\rho_0\cos(\nu - \mu_0) - v\delta_t \cos(\theta^s_{t-1} + \tfrac{1}{2}\omega\delta t - \nu))^2)\end{aligned}$$

where:

$$\begin{cases}\Delta x_0 &= x_t - x^s_{t-1} \\ \Delta y_0 &= y_t - y^s_{t-1} \\ \mu_0 &= \arctan\frac{\Delta y_0}{\Delta x_0} \\ \rho_0^2 &= (\Delta y_0)^2 + (\Delta x_0)^2.\end{cases}$$

Note that

$$\begin{aligned}\rho^2 &= (\Delta x_0 - v\delta t\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t))^2 + (\Delta y_0 - v\delta_t\sin(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t))^2\\ &= \rho_0^2 + v^2\delta t^2 - 2\rho_0 v\delta t\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\mu_0)\end{aligned}$$

Finally

$$\begin{aligned}f(\omega) &= \tfrac{1}{\sigma_M[\rho_0^2+v^2\delta t^2-2\rho_0 v\delta t\cos(\theta^s_{t-1}+\frac{1}{2}\omega\delta t-\mu_0)]^2}(\lambda_+(\rho_0\sin(\nu-\mu_0)+v\delta_t\sin(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu))^2\\ &\quad +\lambda_-(\rho_0\cos(\nu-\mu_0)-v\delta_t\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu))^2).\end{aligned}$$

Now, if $\delta t << 1$, we can forget terms in δt^2

$$\tfrac{1}{[\rho_0^2+v^2\delta t^2-2\rho_0 v\delta t\cos(\theta^s_{t-1}+\frac{1}{2}\omega\delta t-\mu_0)]^2} \approx \tfrac{1}{\rho_0^4}(1+4\tfrac{v\delta t}{\rho_0}\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\mu_0)).$$

Let us call N the numerator, it can be written

$$\begin{aligned}N &= \rho_0^2(\lambda_+\sin^2(\nu-\mu_0)+\lambda_-\cos^2(\nu-\mu_0)) + 2v\delta t\rho_0(\lambda_+\sin(\nu-\mu_0)\sin(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu)-\\ &\quad \lambda_-\cos(\nu-\mu_0)\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu)),\\ &= a + 2v\delta t\rho_0 d(\sin\alpha\sin(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu)-\cos\alpha\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu))\\ &= a - 2v\delta t\rho_0 d\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu+\alpha)\end{aligned}$$

where $\alpha = \arctan(\frac{\lambda_+\sin(\nu-\mu_0)}{\lambda_-\cos(\nu-\mu_0)}) = \arctan(\frac{\lambda_+}{\lambda_-}\tan(\nu-\mu_0))$, $d = \sqrt{\lambda_+^2\sin^2(\nu-\mu_0)+\lambda_-^2\cos^2(\nu-\mu_0)}$, and $a = \rho_0^2(\lambda_+\sin^2(\nu-\mu_0)+\lambda_-\cos^2(\nu-\mu_0))$.

Then,

$$\begin{aligned}f(\omega) &\approx \tfrac{a}{\sigma_M\rho_0^4}(1-2v\delta t\tfrac{\rho_0}{a}d\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu+\alpha))(1+4\tfrac{v\delta t}{\rho_0}\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\mu_0))\\ &\approx \tfrac{a}{\sigma_M\rho_0^4}(1-2v\delta t\tfrac{\rho_0}{a}d\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu+\alpha)+4\tfrac{v\delta t}{\rho_0}\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\mu_0))\\ &\approx \tfrac{a}{\sigma_M\rho_0^4}(1-2v\delta t(\tfrac{\rho_0}{a}d\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\nu+\alpha)-2\tfrac{1}{\rho_0}\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\mu_0)))\\ &\approx \tfrac{a}{\sigma_M\rho_0^4}(1-2v\delta t(\tfrac{\rho_0}{a}d\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t)\cos(\nu-\alpha)+\tfrac{\rho_0}{a}d\sin(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t)\sin(\nu-\alpha)-\\ &\quad 2\tfrac{1}{\rho_0}\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t)\cos(\mu_0)-2\tfrac{1}{\rho_0}\sin(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t)\sin(\mu_0))).\end{aligned}$$

Define

$$e = \frac{\rho_0}{a}d\cos(\nu-\alpha)-2\frac{1}{\rho_0}\cos(\mu_0),$$

and

$$f = \frac{\rho_0}{a}d\sin(\nu-\alpha)-2\frac{1}{\rho_0}\sin(\mu_0),$$

and $\phi = \arctan\frac{f}{e}$, $r = \sqrt{e^2+f^2}$

$$f(\omega) \approx \tfrac{a}{\sigma_M\rho_0^4}(1-2vr\delta t\cos(\theta^s_{t-1}+\tfrac{1}{2}\omega\delta t-\phi))$$

With this expression, the optimal control problem to maximize $f(\omega)$ is resolved by minimizing the cosine.