UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

BAYESIAN HANDLING OF UNCERTAINTY FOR MOBILE ROBOTS

TESIS PARA OPTAR AL GRADO DE DOCTOR EN INGENIERÍA ELÉCTRICA

PABLO ALEXIS GUERRERO PÉREZ

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
Martin Adams
Fabio Ramos
Miguel Torres Torriti

SANTIAGO DE CHILE
AGOSTO 2011

A mi amada esposa, Alejandra, y mis adorados hijos, Martín, Daniela y Francisca.

Resumen

El manejo de incerteza es de gran importancia en el campo de la Robótica Móvil. La correcta estimación y reducción de la incerteza respecto al estado de un robot y su entorno pueden tener un tremendo impacto en el nivel de éxito del robot al realizar una tarea. Esta tesis aborda los problemas de la correcta estimación y la reducción de esta incerteza.

El problema de estimar correctamente la incerteza es abordado a través del desarrollo de un método innovador, llamado *Heteroscedastic-Gaussian-Process Extended Kalman Filter* (HGP-EKF), el cual es una mejora del método existente llamado *Gaussian-Process Extended Kalman Filter* (GP-EKF). Las principales contribuciones de HGP-EKF en comparación con GP-EKF son las siguientes: la regresión y uso de la varianza expresada en los datos de entrenamiento como una función del estado que se está estimando, la habilidad de aprender matrices de covarianza no diagonales, y el uso de parámetros adicionales que pueden entregar pistas para mejorar los modelos observacional y del proceso.

El problema de reducir la incerteza mencionada es abordado mediante el desarrollo de un método de visión activa, llamado *Task-Oriented Active Vision* (Visión Activa Orientada a la Tarea). La principal contribución del paradigma de la visión activa orientada a la tarea, en comparación con otros métodos probabilísticos existentes en la literatura, es que ella intenta explícitamente reducir los componentes de la incerteza acerca del estado más relevantes para la ejecución de la tarea actual. Esta reducción focalizada de la incerteza acerca del estado se logra a través de la consideración de una función de valor relacionada con la tarea que se está ejecutando. Los resultados obtenidos muestran que los métodos de visón activa orientados a la tarea tienen un mejor rendimiento que aquellos basados en la teoría de la información cuando la tarea no es la reducción de la incerteza en sí misma.

En conclusión, se ha demostrado que el correcto uso de la incerteza puede mejorar el rendimiento de un robot en términos de estimar el estado y ejecutar la tarea actual.

Abstract

Handling the existence of uncertainty is a key issue in the field of mobile robotics. Both correctly estimating and reducing the amount of uncertainty about the state of a robot and its environment can potentially have a tremendous impact on the performance of the robot in a determined task.
This thesis addresses precisely these problems of correctly estimating and reducing this uncertainty.

The problem of correctly estimating uncertainty is addressed by developing a novel method, called *Heteroscedastic-Gaussian-Process Extended Kalman Filter* (HGP-EKF), which is an improvement over the existent *Gaussian-Process Extended Kalman Filter* (GP-EKF) method. The main contributions of HGP-EKF in comparison to GP-EKF are the following: the regression and use of the variance expressed in the training data as a function of the state being estimated, the ability to learn non-diagonal covariance matrices, and the use of additional parameters that can offer clues to improve both the process and the observational models.

The problem of reducing the aforementioned uncertainty is addressed by developing an active-vision method. This active-vision method is called *Task-Oriented Active Vision*. The main contribution of the paradigm of Task-Oriented Active Vision in comparison with other probabilistic methods discussed in the literature is that it explicitly intends to reduce the components of the uncertainty in the state that are most relevant to the task being executed. This focused reduction of the uncertainty in the state is achieved by considering a value function that is related to the task being executed. The results obtained show that Task-Oriented Active Vision methods have a better performance than the information-theory based ones when the task is not the reduction of uncertainty itself.
In conclusion, it has been shown that the correct handling of uncertainty can improve the performance of a robot in terms of both estimating the state and executing the selected task.

Acknowledgements

I would like to thank my supervisor Javier Ruiz del Solar for several reasons. Javier helped me to discover my interest in research and taught me what research is about. He was always proposing new and interesting challenges to me and gave me the freedom to develop my own ideas, which in my opinion has been crucial for my development as a researcher. I would also like to thank him for creating the Robotics Laboratory and the robot-soccer team at the Universidad de Chile, which has inspired and motivated me for all of these years. I would also like to thank several professors at the Universidad de Chile for giving me such a high quality education and teaching me to think rigorously.

It is also important to mention the valuable corrections I received not only from Javier but also from the other three reviewers of this Thesis: Martin Adams, Fabio Ramos and Miguel Torres. I thank them all because their corrections made this Thesis improve noticeably.

I would like to thank the Australian Center for Field Robotics (ACFR), at the University of Sydney, for giving me the opportunity to do an internship there. My internship was a very enriching experience and a significant part of this Thesis would not have been possible without the knowledge I gained while working with the ACFR. I would especially like to thank my internship supervisor, Fabio Ramos. Fabio is a very patient and wise supervisor, who has the ability to perfectly balance the interchange of ideas and teaching. Fabio generously gave me very wise advice and taught me almost everything I know about Gaussian Processes. I would also like to thank Francisco Zubizarreta for being such a good project partner and Muhammad Esa Attia for his professional and kind help.

Many of the people I met during my undergraduate and PhD studies at the Universidad de Chile made a contribution to my experience, but there is only space to thank a few. First, I would like to thank Rodrigo Palma, who was my colleague throughout my undergraduate and PhD studies and in the robot soccer team. Rodrigo was a very good friend and an interesting person with whom to discuss ideas. I learned a lot from our conversations and experiences together. I would also like to thank Paul Vallejos, who was my colleague in the

robot soccer team and in the Phd program. Paul was a very sharp partner and it was always very stimulating to discuss our research together. I have also learned a lot from him. I thank them both for their friendship. I would also like to thank Javier Testart, Ricardo Dodds, Pablo Recabal, Felipe Tobar, Matías Arenas, Matías Valenzuela, Román Marchant, José Miguel Yáñez and Sylvain Blunier for being such good laboratory partners. Their friendship made the PhD program a lot more interesting and enjoyable. I would also like to thank Javier Testart, Ricardo Dodds, Luis Alberto Herrera, Pablo Recabal, Miguel Romero, Gonzalo Díaz, Josué Fredes, Matías Arenas, Matías Valenzuela, Román Marchant, José Miguel Yáñez and Sylvain Blunier for their valuable work related to the robot soccer team, their contribution to the development of the UChileLib robotics library and their work on the publications we have written during this years.

I would also like to thank my parents, Laura and Juan, for their unconditional love and support that made me the person I am today. They always stimulated and encouraged me to make me be the best person I could be in all senses. I must also thank my sister, Naty. Her love and company were a crucial part of my support and she always taught and helped me.

There is no way I can sufficiently thank my wife, Alejandra, who has being my companion during these years. I am sincerely grateful for her love, support, patience, and encouragement throughout my PhD studies. Alejandra helped me to make the decision to undertake the PhD and I really thank her for that. I would also like to thank my children, Martin, Daniela and Francisca. They bring happiness to my life and teach me new things every day.

Finally, I would like to thank the Doctoral Scholarship Program of CONICYT and the Advanced Mining Technology Center (AMTC, CONICYT Project FBO09) for financing my Doctorate studies, and the Doctoral Internship Program of Becas Chile for financing my internship in Australia.

This research was partially funded by FONDECYT project 1090250, Chile.

Contents

List of Figures

List of Tables

List of Acronyms

AMDP	Augmented Markov Decision Process
AP	Additional Parameters
Angle-GP	Angle Gaussian Process
CDF	Central Difference Filter
CDP	Continuous Decision Process
DD1	Divided Difference Filter 1
DD2	Divided Difference Filter 2
DDP	Discrete Decision Process
EAV	Expected Action Task Value
EKF	Extended Kalman Filter
EKPF	Extended Kalman Particle Filter
GP	Gaussian Process
GP-EKF	Gaussian-Process Extended Kalman Filter
GT	Ground Truth
HGP	Heteroscedastic Gaussian Process
HGP-EKF	Heteroscedastic-Gaussian-Process Extended Kalman Filter
IEKF	Iterative Extended Kalman Filter
ISI	Institute for Scientific Information
KF	Kalman Filter
LNCS	Lecture Notes in Computer Science
LRKF	Linear Regression Kalman Filter

MAP	Maximum a Posteriori
MCMC	Markov Chain Monte Carlo
MC-POMDP	Monte-Carlo Partially-Observable Markov Decision Process
MDP	Markov Decision Process
MI	Mutual Information
ML-HGP	Most-Likely Heteroscedastic Gaussian Process
MMSE	Minimum Mean Square Error
MOGP	Multiple-Output Gaussian Process
MSE	Mean Square Error
pdf	Probability Density Function
PF	Particle Filter
POMDP	Partially-Observable Markov Decision Process
RANSAC	Random Sample Consensus
RBF	Rao-Blackwellized Filter
ROI	Region of Interest
RS	Reference System
SinCos-GP	Sine-Cosine Gaussian Process
SERV-HGP	Simultaneous Estimation and Regression of the Variance Heteroscedastic Gaussian Process
SLAM	Simultaneous Localization and Mapping
SPKF	Sigma-Point Kalman Filter
SR-UKF	Square Root Unscented Kalman Filter
SRUPF	Square-Root Unscented Particle Filter
STD	Standard Deviation
SVD	Singular Value Decomposition

TOOC	Task-Oriented Optimality Criteria
UKF	Unscented Kalman Filter
UPF	Unscented Particle Filter
UT	Unscented Transform
VV	Value Variance

Chapter 1. Introduction

1.1 General Motivation

There has been a great deal of research in the field of mobile robotics in recent decades and today it continues to be the subject of increasing interest. The reason for this intense research activity is clear: robots are expected to become a very important part of human life in the next decades. Analogously to what happened with computers in the eighties and nineties, the expectation is that robots change from being a very expensive piece of laboratory equipment with few practical applications to having a more mainstream presence in homes, offices, and industries, with thousands of applications. Of course, there is still a gap that must be filled before this hypothetic situation becomes a reality. This gap has several components, including many hardware and software features that will need to be improved.

1.2 Definition of the Problem

This work addresses the problem of handling uncertainty in a mobile robot that is performing a non-trivial task in a complex and dynamic world. Most real robots' controllers are based on digital computer and thus they must make decisions in discrete time.

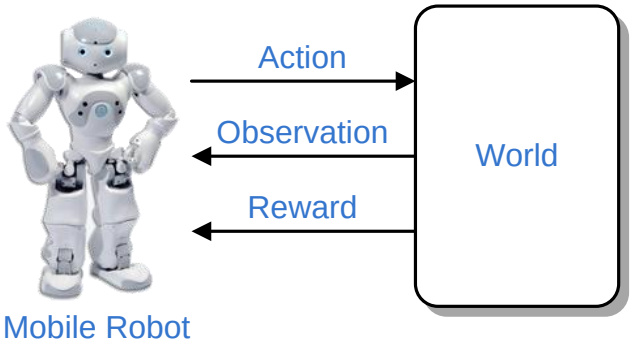


Figure 1. Interaction between a mobile robot and its environment.

Therefore, a natural approach for executing a task is the sequential decision-making approach. In the sequential decision-making approach, time is discretized and in every instant, the robot: (i) executes an action which influences the world, then (ii) senses an observation from the world, and finally (iii) receives an immediate reward depending on how successfully the robot is performing the task. This interaction is illustrated in Figure 1.

While the world is infinite dimensional, there is often a finite set of variables that can define the most relevant part of the world from the viewpoint of the task being executed. This set of variables is known as the *state*. The state evolves through time in a manner that might be partially or totally unknown to the robot. The evolution of the state is influenced by the robot's actions, while at each instant the observation measured and reward received by the robot are influenced by the state.

In order to understand the sources of uncertainty, we will first conduct a brief review of the information that the robot uses to model the world and make decisions and the way this information is used.

The most obvious sources of information for a robot are its sensors. Sensors gather information from the world and make it available for the robot. If the sensors were perfect, no additional information or uncertainty treatment would be necessary. Unfortunately, sensors are far from being perfect and the sensorial information is:

- Limited: The information collected by sensors is limited for several reasons. From a theoretical point of view, the world is infinite dimensional while sensorial information is always finite dimensional. In most mobile robotics applications, the sensors are embedded in the robot itself. Thus, severe weight, space, and energy consumption constraints may be imposed on the sensors. Additionally, most sensors have a limited range in terms of distance and their resolution decays with distance. Several families of sensors also have a limited angle range (for example laser, sonar and visual sensors). Moreover, most sensors are affected by occlusions. Of course, there are always cost considerations in the design of a mobile robotic system that also restrict the sensing capabilities of the robot. There are also computational limitations that restrain the quantity of sensorial information the robot can process in real time.

- Indirect: The sensorial information does not necessarily contain the state and there is not necessarily a map from the sensorial information to the state. Therefore, many times the sensorial information only gives incomplete clues about the state and the latter must be inferred from the former.
- Noisy: The sensorial information is naturally noisy. Each kind of sensors has its own sources of noise.

Other key information sources are the models that the robot has related to its interaction with the world. These models usually include: (i) a *process model* that describes the dynamics of the evolution of the state and, therefore, the influence of the action on the state, (ii) an *observational model* that depicts the influence of the state on the observations, and (iii) a *reward model* that quantifies the immediate gain that the robot will get from a given state. Most of the time, these models are only partially known and there is a great amount of uncertainty stemming from this lack of knowledge. Furthermore, generally the observational and process models do not only depend on known variables, such as the previous action, or partially known variables, such as the state, but also on totally unknown variables that cannot be measured or inferred in any way. These sets of variables are respectively called the *observational noise* and *process noise*.

While handling uncertainty in a mobile robot is a very complex topic with several aspects to be considered, this thesis addresses the problem from two viewpoints.

The first viewpoint relates to model learning and characterization of uncertainty. As it was already stated, most of the time, the observational and process models are unavailable and/or their theoretical versions are too inaccurate. Furthermore, the noise statistics are often assumed to have arbitrary values or learned from data but with stationarity assumptions. From this viewpoint, this thesis work presents a method for simultaneously learning the models and the noise statistics conditioned on the state, the action and possibly other variables.

The second viewpoint is that of task-oriented uncertainty reduction using active vision. The main idea behind this approach is to modify the optimality criterion in an active-vision system so that the system minimizes the most relevant components of the uncertainty in the state for the task being performed.

1.3 Notation and Basic definitions

In time step *k*, the *world state*, $\mathbf{x}_k$, is the set of the relevant variables that describe the current condition of the world in order for the robot to make decisions. Let $\mathbf{u}_k$ denote the robot's action, and $\mathbf{z}_k$ the robot's observation. Let us define $U_k = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ and $Z_k = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ as the respective sets of actions and observations up to instant *k*. Then, we can define the *state belief*, or simply the *belief*, $b_k(\mathbf{x}_k) = p(\mathbf{x}_k | U_k, Z_k)$, as the pdf of $\mathbf{x}_k$ given all the observations and actions up to time step *k*.

The state is typically expected to fulfill the *Markov property*, which means that it contains all the information provided by the past observations and actions. In other words, given the state $\mathbf{x}_k$ at time *k*, the actions' and observations' sets, U_k and Z_k, do not provide information about the next state, $\mathbf{x}_{k+1}$. Put in equations,

$$p(\mathbf{x}_{k+1} | \mathbf{x}_k, U_{k+1}, Z_k) = p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_{k+1}). \quad (1)$$

In order to model the dynamics of the state and the observations, we define two probabilistic models: the *process model*, $p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_{k+1})$, which models the effect of the robots' actions in the state transitions, and the *observation model*, $p(\mathbf{z}_k | \mathbf{x}_k, \mathbf{u}_k)$, which models the relationship between the current state and the current observation. If the next state and the observation do not depend on additional known variables, the process and observation models become functions, respectively: $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_{k+1}, \mathbf{w}_k)$ and $\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k)$ with $\mathbf{w}_k$ and $\mathbf{v}_k$ respectively the process and observational noises. In the Gaussian case, $\mathbf{w}_k$ and $\mathbf{v}_k$ are zero-mean Gaussian noises with their respective covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$. The Jacobians, with respect to the state, of f and h functions are noted as $\mathbf{F}_k$ and $\mathbf{H}_k$, and we can refer to them as the process and observation Jacobians.

For any task that the robot is performing, we should be able to define a *task value function*, $V_k(\mathbf{x}_k)$, which tells the robot how convenient the state $\mathbf{x}_k$ is for the accomplishment of this task. Depending on the complexity of the task, $V_k(\mathbf{x}_k)$ might be obtained from simple ad-hoc heuristics or calculated from popular methods like *Markov Decision Processes* or

Reinforcement Learning [39]. Note that the function $V_k(\mathbf{x}_k)$ is, in general, dependent on k[1]. In the case when $V_k(\mathbf{x}_k)$ does not depend on k, the subscript can be omitted.

1.4 Objectives

1.4.1 General Objective

The general objective of this thesis is to study the relevance of handling uncertainty in the performance of a mobile robot while it is executing a determined task. This performance will be measured as the mean in time of a reward function that is adequate to the task being executed.

1.4.2 Specific Objectives

The specific objectives of this thesis are the following:

- To evaluate the utility of making a characterization of the uncertainty in the observations and actions as a means to achieve a better performance of the robot's state estimation and decision making.
- To develop an active-vision system for mobile robots that can reduce the robot uncertainty in order to maximize the robot's performance on an arbitrary task.
- To study the feasibility of developing a robot control system that handles uncertainty in real time.

1.5 Hypotheses

The hypotheses of this thesis are the following:

- It is possible and convenient for an active vision system to explicitly consider the task that the robot is performing in order to direct the uncertainty reduction to the most relevant components.

---

[1] An example of tasks in which V_k depends on k is when the task has a finite time horizon, because V_k might depend on how much time is still available for the task execution.

- By making the process and observational noises, instead of fixed statistics, functions of the current estimated state, action and observation, it may be possible to improve the performance of the state estimation process.
- It is convenient for the state estimation process to collect and consider some additional parameters in the perception and actuation processes. These additional parameters can be collected from additional sensors in the robot, and from intermediate variables in the perception process that are not considered in the observation itself.

1.6 Background

1.6.1 Two-Dimensional Reference-System Transformations

Every position or pose[2] is referred to a determined reference system (RS). It is of great help for the state estimation methods in mobile robotics to define operations of RS transformation. First, let us note that a RS itself can be defined by a pose with respect to an implicitly defined absolute RS, where the position corresponds to the origin of the RS and the orientation corresponds to the rotation of the RS. Then, given any arbitrary pose, $\mathbf{p} = \left(p_x, p_y, p_\theta \right)^T$, we can define transformations between the absolute RS and any arbitrary RS, $\mathbf{s} = \left(s_x, s_y, s_\theta \right)^T$. The transformations go in both directions:

$$T_{rel \to abs}\left(\mathbf{s}, \mathbf{p}\right) = \begin{bmatrix} \mathbf{s}_{pos} + Rot\left(s_\theta\right)\mathbf{p}_{pos} \\ s_\theta + p_\theta \end{bmatrix}, \tag{2}$$

$$T_{abs \to rel}\left(\mathbf{s}, \mathbf{p}\right) = \begin{bmatrix} Rot\left(-s_\theta\right)\left(\mathbf{p}_{pos} - \mathbf{s}_{pos}\right) \\ p_\theta - s_\theta \end{bmatrix}. \tag{3}$$

[2] A pose is defined as the position and orientation of an object.

Where $\mathbf{p}_{pos} = (p_x, p_y)$ and $\mathbf{s}_{pos} = (s_x, s_y)$ are the positions of respectively $\mathbf{p}$ and $\mathbf{s}$, and $Rot(\alpha)$ is the rotation matrix for angle α. It is also interesting to calculate the Jacobians of these functions with respect to the RS and the parameter pose:

$$\frac{\partial T_{rel\to abs}}{\partial \mathbf{s}}(\mathbf{s},\mathbf{p}) = \begin{bmatrix} I_{2\times 2} & \dot{Rot}(s_\theta)\mathbf{p}_{pos} \\ 0_{1\times 2} & 1 \end{bmatrix}, \tag{4}$$

$$\frac{\partial T_{rel\to abs}}{\partial \mathbf{p}}(\mathbf{s},\mathbf{p}) = \begin{bmatrix} Rot(s_\theta) & 0 \\ 0_{1\times 2} & 1 \end{bmatrix}, \tag{5}$$

$$\frac{\partial T_{abs\to rel}}{\partial \mathbf{s}}(\mathbf{s},\mathbf{p}) = \begin{bmatrix} -Rot(-s_\theta) & \dot{Rot}(-s_\theta)(\mathbf{s}_{pos} - \mathbf{p}_{pos}) \\ 0_{1\times 2} & -1 \end{bmatrix}, \tag{6}$$

$$\frac{\partial T_{abs\to rel}}{\partial \mathbf{p}}(\mathbf{s},\mathbf{p}) = \begin{bmatrix} Rot(-s_\theta) & 0 \\ 0_{1\times 2} & 1 \end{bmatrix}, \tag{7}$$

where $I_{N\times N}$ is the $N\times N$ identity matrix, $0_{N\times M}$ is the $N\times M$ zero-valued matrix, and $\dot{Rot}(\alpha)$ is the element-wise derivative of $Rot(\alpha)$ with respect to α.

1.6.2 Approximated Integration

In general, an integral of the form $\int p(a)q(a)da$, where $p(a)$ is a continuous pdf and $q(a)$ is any function of the D_a-dimensional random variable a, is not computable. One obvious approximation consists of discretizing the space of a and summing the integrand over all the discretization points, $\{a_i\}$, yielding $\sum_i p(a_i)q(a_i)$. This approximation is, in general, inefficient since most of the points $\{a_i\}$ may be in regions of the space where p is zero or too small. A more efficient technique is to use sampling techniques for approximating these integrals. If it is possible to sample N_a weighted samples from $p(a)$: $\{(\chi_i^a, \omega_i^a)\}$, with $\sum_i \omega_i^a = 1$. Then, the former integral can be approximated to:

$$\int p(a)q(a)da \approx \sum_i \omega_i^a q\left(\boldsymbol{\chi}_i^a\right). \tag{8}$$

In many practical applications the pdf's are considered to be Gaussians. Furthermore, as is argued in [28], almost any continuous pdf can be approximated by a sum of Gaussians, and then it is possible to obtain samples from them using Gaussian sampling. We consider two alternatives for sampling from a Gaussian's pdf $N(\mu_a, \Sigma_a)$: random sampling and deterministic sampling. In the following subsections we will detail these procedures.

1.6.2.1 Random sampling

This method is also known as Monte Carlo, and it has been widely used over the last decades for several applications, including active vision [11], [28], [29]. There are several methods for obtaining a random sample from a Gaussian (see for example [42]). If the sampling procedure is repeated N_a times, then $\left\{\boldsymbol{\chi}_i^a\right\}$ is obtained. The weights are set all equal: $\omega_i^a = 1/N_a$

1.6.2.2 Deterministic Sampling

1.6.2.2.1 Mean

This is a trivial and widely used procedure, most of the times without making it explicit. It consists of considering only one sample, in the mean of the pdf, i.e., $N_a = 1$, and $\left\{\left(\boldsymbol{\chi}_i^a, \omega_i^a\right)\right\} = \left\{\left(\mu_a, 1\right)\right\}$. This great simplicity comes at the price of a poor representation of the pdf.

1.6.2.2.2 Sigma Points

In a sigma point scheme, $\left\{\left(\boldsymbol{\chi}_i^a, \omega_i^a\right)\right\}$ are sampled following some deterministic pattern that conserves the pdf's two first moments. A common procedure for achieving this property is to obtain a *square root*, S_a, of the covariance matrix Σ_a, i.e., a matrix S_a that fulfills the property: $\Sigma_a = S_a S_a^T$. If $\mathbf{s}_a^j$ is the j^{th} column of S_a, then the following selection of $\left\{\left(\boldsymbol{\chi}_i^a, \omega_i^a\right)\right\}$ ensures the two first moments conservation:

$$\boldsymbol{\chi}_i^a = \begin{cases} \mu_a & i = 0 \\ \mu_a + \sqrt{(D_a + \eta)}\mathbf{s}_a^i & i \in [1, D_a] \\ \mu_a - \sqrt{(D_a + \eta)}\mathbf{s}_a^{i-D_a} & i \in [D_a + 1, 2D_a] \end{cases} . \quad (9)$$

The corresponding weights are:

$$\omega_i^a = \begin{cases} \omega_0 = \dfrac{\eta}{(D_a + \eta)} & i = 0 \\ \dfrac{1}{2(D_a + \eta)} & \sim \end{cases} . \quad (10)$$

Where η is a scaling's parameter that determines the spread of the samples. In our particular implementation, we have set the scaling parameter η to 1. Note that in this case, $N_a = 2D_a + 1$.

Two methods for obtaining S_a from Σ_a, which derive in two different sigma-point methods, are mentioned: The first one is called the *Unscented Transform* (UT) [43], and in this case, S_a is selected as the *Cholesky decomposition* [42] of Σ_a. In this case, S_a is a triangular square root of Σ_a. The second one is based on the *Singular Value Decomposition* (SVD) [42], which for a covariance matrix is equivalent to the *Eigenvalue Decomposition*[3]. Then, from the SVD, we get a decomposition of the form $\Sigma_a = V_a \Lambda_a V_a^T$ where Λ_a is a diagonal matrix with the eigenvalues of Σ_a in its diagonal. There is a trivial square root L_a of Λ_a which is a diagonal matrix where each diagonal element is the square root of the corresponding eigenvalue in Λ_a. Thus, S_a may be selected as $S_a = V_a L_a$ which is obviously a square root of Σ_a.

[3] Since the covariance matrix is symmetric and positive semi-definite, SVD and the eigenvalue decomposition are equivalent.

1.6.3 Bayesian Estimation

As previously stated, one of the main uncertainties a mobile robot must deal with when solving non-trivial tasks in a complex and dynamic world is the state uncertainty. This uncertainty comes from the fact that the main variables that influence the robot performance may be only partially observable.

Several state-estimation methods have been used to solve different problems in robotic applications. The most successful known methods for solving the state-estimation problem are the so-called Bayesian state-estimation methods. This subsection introduces Bayesian state-estimation methods while the related literature is reviewed in subsection 3.1.1. An excellent survey for Bayesian state estimation can be found in [48].

Bayesian filters are probabilistic state estimators that recursively maintain an estimate of the state belief. Bayesian filters rely heavily on the Markov assumption. Given this fact, Bayesian filters execute two stages in every instant: prediction and correction. These stages are depicted in figure 2.

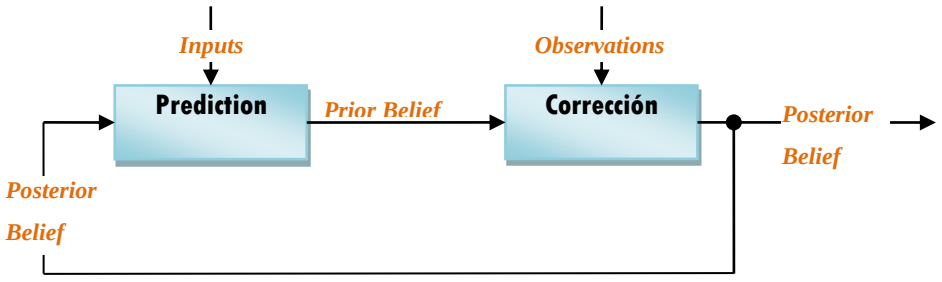


Figure 2. Bayesian filters for state estimation: predictive and corrective stages.

In the predictive stage, the filter calculates the prior belief, which is the result of modifying the posterior belief of the previous instant by adding the effect of the inputs. The prediction stage is performed in accordance with the process model. In the corrective stage, the filter calculates the posterior belief by conditioning the prior belief on the observations. The correction stage is performed in accordance with the observational model. There is an exact analytic solution for the Bayesian filtering problem with arbitrary models and pdf's.

Unfortunately, this solution is computationally intractable since it contains integrals over the state and observational spaces.

There are specific situations in which the state estimator can be implemented computationally. For example, if both the process and the observational models are linear, the system is called linear. The state estimator for linear systems is much simpler than the general case. The state estimators also become significantly simpler when the spaces of the belief and of the pdf's of the noises are restricted. For instance, if the state and observational spaces are finite, then the integrals over the state space become computable sums. Another restriction of the pdf's that might lead to simplifications is when they belong to any family of parametric functions from which Gaussian pdf's are a notable case.

Sometimes these restrictions over the models and/or the pdf's are assumed to be held by the system even when they are not. These assumptions lead to approximate solutions that, depending on how far the real system pdf's are from the assumed ones, could result in the poor performance of the state estimator. The so-called *Kalman Filter* (KF) estimates the mean and covariance matrix of the state when the models are linear. There are at least two relevant KF-based approximations for the non-linear case: the *Extended Kalman Filter* (EKF), based on the linearization of the models, and the *Unscented Kalman Filter* (UKF), which is based on the Unscented Transform (see subsection 1.6.2.2.2). A different approach is that of the *Particle Filters* (PF), where the pdfs are approximated by using random sampling. Due to its relevance to this Thesis, only the KF and EKF will be further described in this subsection.

1.6.3.1 The Kalman Filter

The Kalman Filter is suitable for systems with linear models with additive noise, i.e., $\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k^{\mathbf{x}} + \mathbf{w}_k$ and $\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$. If the models are linear and the two first moments of the noises are known, the KF is the *minimum mean squared error* (MMSE) estimator for the mean and the variance of the state. The KF has the following predictive stage:

$$\mu_k^{\mathbf{x}-} = \mathbf{F}_k \mu_{k-1}^{\mathbf{x}} + \mathbf{B}_k \mathbf{u}_k^{\mathbf{x}}, \tag{11}$$

$$\Sigma_k^{\mathbf{x}^-} = \mathbf{F}_k \Sigma_k^{\mathbf{x}} \mathbf{F}_k^T + \mathbf{Q}_k . \tag{12}$$

While corrective stage is the following:

$$\mu_k^{\mathbf{x}} = \mu_k^{\mathbf{x}^-} + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H} \mu_k^{\mathbf{x}^-} \right), \tag{13}$$

$$\Sigma_k^{\mathbf{x}} = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \Sigma_k^{\mathbf{x}^-}, \tag{14}$$

where $\mathbf{K}_k$ is known as the *Kalman gain*:

$$\mathbf{K}_k = \Sigma_k^{\mathbf{x}^-} \mathbf{H}_k^T \left(\mathbf{H}_k \Sigma_k^{\mathbf{x}^-} \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1}. \tag{15}$$

1.6.3.2 The Extended Kalman Filter

For the non-linear and non-additive case, the Jacobians of the models, $\mathbf{F}_k = \partial f / \partial \mathbf{x}_k$, $\mathbf{W}_k = \partial f / \partial \mathbf{w}_k$, $\mathbf{H}_k = \partial h / \partial \mathbf{x}_k$, and $\mathbf{V}_k = \partial h / \partial \mathbf{v}_k$, can be used to make a first-order Taylor approximation of the solution. The EKF has the following predictive stage:

$$\mu_k^{\mathbf{x}^-} = f \left(\mu_{k-1}^{\mathbf{x}}, \mathbf{u}_k^{\mathbf{x}}, 0 \right), \tag{16}$$

$$\Sigma_k^{\mathbf{x}^-} = \mathbf{F}_k \Sigma_k^{\mathbf{x}} \mathbf{F}_k^T + \mathbf{W}_k \mathbf{Q}_k \mathbf{W}_k^T . \tag{17}$$

While corrective stage is the following:

$$\mu_k^{\mathbf{x}} = \mu_k^{\mathbf{x}^-} + \mathbf{K}_k \left(\mathbf{z}_k - h\left(\mu_k^{\mathbf{x}^-} \right) \right), \tag{18}$$

$$\Sigma_k^{\mathbf{x}} = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \Sigma_k^{\mathbf{x}^-}, \tag{19}$$

where the Kalman gain, $\mathbf{K}_k$, is now defined as:

$$\mathbf{K}_k = \Sigma_k^{\mathbf{x}^-} \mathbf{H}_k^T \left(\mathbf{H}_k \Sigma_k^{\mathbf{x}^-} \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T \right)^{-1}. \tag{20}$$

See a description of the Jacobian and covariance matrices in section 1.3.

1.6.4 Gaussian Processes for Regression

Gaussian Processes (GPs) are a non-parametric tool for non-linear regression and classification. In this subsection, we will provide a brief summary of GPs from a practical

viewpoint. An excellent summary that includes both theoretical and practical aspects and references to deeper theoretical insights can be found in [32]. Although we are only interested in the regression capabilities of GPs, they can also solve classification problems as is also shown in [32].

The regression problem can be defined as the problem of learning a function f from finite data samples $\{(\mathbf{x}_i, y_i)\}$.[4,5]. We are interested in the case when these samples have an output noise, i.e., $y_i = f(\mathbf{x}_i) + \varepsilon_i$. GPs are able to solve this kind of regression problems at least when the noise ε_i is assumed to be Gaussian with zero mean and arbitrary variance σ_n^2. Furthermore, for any arbitrary input test, GPs are able to give a predictive mean and variance of the output.

1.6.4.1 Covariance Functions

Covariance functions are a key component of GPs for regression and classification. Covariance functions encode the information of the kind of functions that a GP can learn. They also restrict the possible measures of proximity that are necessary for the regression mechanism to operate. Usually covariance functions have parameters, called *hyperparameters* and denoted $\boldsymbol{\theta}$, but the GPs theory brings a method to calculate them from data and that is why GPs are said to be non-parametric (even when the selection of the covariance function itself is a parameter).

There are several kinds of covariance functions examined in the literature and some of them are described in [32]. The covariance function is defined as the covariance of the outputs of f for two input points:

$$k(\mathbf{x}, \mathbf{x}') = \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')). \tag{21}$$

---

[4] Even though the state (contextualized in the state estimation problem) is noted by $\mathbf{x}$ in the rest of the document, in subsections 1.6.4 and 1.6.6, $\mathbf{x}$ will stand for a regression input.

[5] Note that in this definition, the output y_i is one-dimensional. This restriction will remain valid up to subsection 1.6.4.3. The multiple-output problem is mentioned in subsection 1.6.4.4.

The selection of an adequate covariance function is crucial for the correct resolution of a determined problem. Since the focus of this thesis work is not in the theoretical aspects of GPs, we will only make use of the most common covariance function, which is the *squared exponential covariance function*:

$$k(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T W (\mathbf{x}-\mathbf{x}')\right). \quad (22)$$

Where W is a diagonal matrix with *scaling factors* in its diagonal. Then, for the squared exponential covariance function, the hyperparameters are the so-called *signal variance,* σ_f^2, the scaling factors in the diagonal of W, and the noise variance σ_n^2.

1.6.4.2 Prediction

Given a covariance function and a set of values for its hyperparameters, a GP can predict the output mean and variance for any arbitrary input point set.

For convenience, given a set of input vectors, $\{\mathbf{x}_i'\}$, we will define its *aggregated matrix* as:

$$Agg(\{\mathbf{x}_i'\}) = [\mathbf{x}_1' \quad \dots \quad \mathbf{x}_n'] . \quad (23)$$

Then, let X denote the aggregated matrix of the training input set, $\{\mathbf{x}_i\}$, and $\mathbf{y}$ denote the transposed of the aggregated matrix of the training output set, $\{y_i\}$. Note that, since the output is one-dimensional, $\mathbf{y}$ is actually a vector. If we have a set of test inputs, $\{\mathbf{x}_{*i}\}$, then let X_* denote the aggregated matrix of $\{\mathbf{x}_{*i}\}$. Additionally, let us define $f_{*i} = f(\mathbf{x}_{*i}) \sim N(\bar{f}_{*i}, \mathrm{var}(f_{*i}))$, which of course is a random variable. Finally, let $\mathbf{f}_*$ denote the aggregated matrix of $\{f_{*i}\}$, $\bar{\mathbf{f}}_*$ its mean and $\mathrm{cov}(\mathbf{f}_*)$ its covariance matrix.

Given two aggregated matrices X' and X'' of respectively the input sets $\{\mathbf{x}_i'\}$ and $\{\mathbf{x}_i''\}$, we can define the *covariance matrix between* X' *and* X'', $K(X', X'')$, as the matrix whose components are defined by:

$$K(X', X'')_{i,j} = k(\mathbf{x}_i', \mathbf{x}_j'') . \quad (24)$$

Let us define $K_X = K(X, X)$ and call it simply *the covariance matrix*, and let us define $K_* = K(X, X_*)$. Then, the predictive mean and covariance of $\mathbf{f}_*$, given the input vector set $\{\mathbf{x}_{*i}\}$, are [32]:

$$\bar{\mathbf{f}}_* = K_*^T \left(K_X + \sigma_n^2 I\right)^{-1} \mathbf{y}, \tag{25}$$

$$\operatorname{cov}(\mathbf{f}_*) = K(X_*, X_*) - K_*^T \left(K_X + \sigma_n^2 I\right)^{-1} K_*. \tag{26}$$

If there is only one test point $\mathbf{x}_*$, we can write $\mathbf{k}_* = K(X, \mathbf{x}_*)$ and then the predictive mean and variance of the output f_*, given the input $\mathbf{x}_*$, are:

$$\bar{f}_* = \mathbf{k}_*^T \left(K_X + \sigma_n^2 I\right)^{-1} \mathbf{y}, \tag{27}$$

$$\operatorname{var}(f_*) = K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T \left(K_X + \sigma_n^2 I\right)^{-1} \mathbf{k}_*. \tag{28}$$

Finally, let us note $\{y_{*i}\}$ the output set for the test input set $\{\mathbf{x}_{*i}\}$ and $\mathbf{y}_*$ its aggregated matrix. Then, given the assumptions on ε_i, $\mathbf{y}_*$ has mean $\bar{\mathbf{f}}$ and covariance matrix $\operatorname{cov}(\mathbf{f}_*) + \sigma_n^2 I$. For convenience, we will define the functions $GP_\mu(\mathbf{x}_*)$ and $GP_\Sigma(\mathbf{x}_*)$ as respectively the predictive mean and predictive variance of $\mathbf{y}_*$ at input point $\mathbf{x}_*$. Put in equations,

$$GP_\mu(\mathbf{x}_*) = K_*^T \left(K_X + \sigma_n^2 I\right)^{-1} \mathbf{y}, \tag{29}$$

$$GP_\Sigma(\mathbf{x}_*) = K(\mathbf{x}_*, \mathbf{x}_*) - K_*^T \left(K_X + \sigma_n^2 I\right)^{-1} K_* + \sigma_n^2. \tag{30}$$

1.6.4.3 Learning

As it was stated before, there are automatic methodologies to learn the hyperparameters from the training data. The main idea behind this process is the maximization of the *log marginal likelihood*, which corresponds to (see [32] for derivation):

$$\log p(y|X) = -\frac{1}{2}\mathbf{y}^T \left(K_X + \sigma_n^2 I\right)^{-1} \mathbf{y} - \frac{1}{2}\log\left|K_X + \sigma_n^2 I\right| - \frac{n}{2}\log 2\pi. \tag{31}$$

Then, the learning process consists of finding the hyperparameter vector that maximizes $\log p\left(y\middle|X\right)$. Note that the hyperparameters vector contains a set of parameters for the covariance function, that will influence $\log p\left(y\middle|X\right)$ through K_X, and σ_n^2 that influences $\log p\left(y\middle|X\right)$ directly. A convex optimization algorithm may be employed for finding the maximum. A useful expression for the maximization algorithm is the marginal likelihood gradient:

$$\frac{\partial \log p\left(y\middle|X,\boldsymbol{\theta}\right)}{\partial \theta_j} = \frac{1}{2}\mathrm{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \left(K_X + \sigma_n^2 I\right)^{-1}\right)\frac{\partial\left(K_X + \sigma_n^2 I\right)}{\partial \theta_j}\right), \tag{32}$$

where θ_j is the j^{th} element of $\boldsymbol{\theta}$, i.e., the j^{th} hyperparameter, and $\boldsymbol{\alpha} = \left(K_X + \sigma_n^2 I\right)^{-1}\mathbf{y}$.

The calculus of the matrix derivatives $\partial\left(K_X + \sigma_n^2 I\right)\big/\partial\theta_j$ is detail ed in the following element-wise equations:

$$\left[\frac{\partial\left(K_X + \sigma_n^2 I\right)}{\partial W_{i,i}}\right]_{j,k} = -\frac{1}{2}\sigma_f^2\left(\left[\mathbf{x}_j - \mathbf{x}_k\right]_i\right)^2 \exp\left(-\frac{1}{2}\left(\mathbf{x}_j - \mathbf{x}_k\right)^T W\left(\mathbf{x}_j - \mathbf{x}_k\right)\right), \tag{33}$$

$$\frac{\partial\left(K_X + \sigma_n^2 I\right)}{\partial \sigma_f^2} = \exp\left(-\frac{1}{2}\left(\mathbf{x} - \mathbf{x}'\right)^T W\left(\mathbf{x} - \mathbf{x}'\right)\right), \tag{34}$$

$$\frac{\partial\left(K_X + \sigma_n^2 I\right)}{\partial \sigma_n^2} = I, \tag{35}$$

1.6.4.4 Multiple Outputs

When the output is multidimensional, a *multiple-output GP* (MOGP) may be defined. Although there are formal methods for treating GPs with correlated multiple outputs (e.g. [33]), the most common practice is to assume them as independent one-dimensional GPs. To do so, given a finite set of data samples $\left\{\left(\mathbf{x}_i, \mathbf{y}_i\right)\right\}$, a set of one-dimensional GPs $\left\{GP^j\right\}$ is defined, where GP^j will perform the regression of the j^{th} dimension of $\mathbf{y}_i$. Then, GP^j

will be a one-dimensional GP that will use as training data the set $\left\{\left(\mathbf{x}_i, y_{i,j}\right)\right\}$ where $y_{i,j}$ is the j^{th} component of vector $\mathbf{y}_i$. Then, the predictive mean of the MOGP will be a vector formed by the predictive means of each of the one-dimensional GPs, while the predictive covariance matrix of the MOGP will be a diagonal matrix containing in each diagonal element the predictive variance of the respective one-dimensional GP. Put in element-wise equations:

$$\left[\mathbf{MOGP}_{\mu}\left(\mathbf{x}_*\right)\right]_j = GP_{\mu}^{j}\left(\mathbf{x}_*\right), \tag{36}$$

$$\left[\mathbf{MOGP}_{\Sigma}\left(\mathbf{x}_*\right)\right]_{i,j} = \begin{cases} GP_{\Sigma}^{j}\left(\mathbf{x}_*\right) & i = j \\ 0 & \sim \end{cases}. \tag{37}$$

It is also possible to write matrix equations:

$$\mathbf{MOGP}_{\mu}\left(\mathbf{x}_*\right) = \begin{bmatrix} Agg\left(\left\{y_{1,j}\right\}\right)\left(K_X^1 + \sigma_n^2 I\right)^{-1} K_*^1 \\ \vdots \\ Agg\left(\left\{y_{N_{\mathbf{y}},j}\right\}\right)\left(K_X^{N_{\mathbf{y}}} + \sigma_n^2 I\right)^{-1} K_*^{N_{\mathbf{y}}} \end{bmatrix}, \tag{38}$$

$$\left[\mathbf{MOGP}_{\Sigma}\left(\mathbf{x}_*\right)\right]_{i,j} = \begin{bmatrix} GP_{\Sigma'}^{1}\left(\mathbf{x}_*\right) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & GP_{\Sigma'}^{N_{\mathbf{y}}}\left(\mathbf{x}_*\right) \end{bmatrix} + \sigma_n^2 I_{N_{\mathbf{y}} \times N_{\mathbf{y}}}, \tag{39}$$

where $GP_{\Sigma'}^{j}\left(\mathbf{x}_*\right) = K^{j}\left(\mathbf{x}_*, \mathbf{x}_*\right) - \left(K_*^{j}\right)^T \left(K_X^{j} + \sigma_n^2 I\right)^{-1} K_*^{j}$, $N_{\mathbf{y}}$ is the dimensionality of the output and the superscripts in the covariance matrices stand for the fact that each one-dimensional GP has its own hyperparameters and consequently different covariance function parameters.

1.6.5 Gaussian-Process Bayesian Estimation

It is a very common assumption in the Bayesian estimation literature that the observational and process models are given. Most of the time, this assumption does not hold and often theoretical models do not accurately approximate the behavior of the system in practice. Recently, a set of methods that combine GPs and Bayesian estimation has been developed

[89][90]. The so-called GP-BayesFilters use the regression techniques provided by GPs to build the observational and process model from real data. Furthermore, they make the noises' variances functions of the state. GP-BayesFilters were tested on the estimation of a robotic blimp's 3D kinematic state, which has twelve dimensions [89]. While algorithms for instantiating this idea to EKF, UKF, and PF have been developed and published, the EKF instantiation will be the only one explained in this document. The reason for this selection is that the EKF instantiation, called *Gaussian Process EKF* (GP-EKF), will serve as a base to one of the methods proposed in section 3.1.

In GP-EKF, training samples are used to predict the process and observational models. The observational model GP, noted $GP^{\mathbf{h}}$, can be learned from a training set $\left\{\left(\mathbf{x}_i^{\mathbf{h}}, \mathbf{y}_i^{\mathbf{h}}\right)\right\}$ where each pair consists of the state and the observation in a determined instant. Analogously, the process model GP, noted $GP^{\mathbf{f}}$, can be learned from a training set $\left\{\left(\mathbf{x}_i^{\mathbf{f}}, \mathbf{y}_i^{\mathbf{f}}\right)\right\}$ consisting in triplets state-action-resulting state. In this case, the input $\mathbf{x}_i^{\mathbf{f}}$ is a vector that concatenates the state and the action, while the output $\mathbf{y}_i^{\mathbf{f}}$ corresponds to the resulting state. Additionally, the Jacobians of these models must be calculated in order to use them in the EKF linear operations. The Jacobian of the predictive mean of a GP can be directly derived from equation (29):

$$\frac{\partial GP_{\mu}\left(\mathbf{x}_*\right)}{\partial \mathbf{x}_*} = \frac{\partial K_*{}^{T}}{\partial \mathbf{x}_*}\left(K_X + \sigma_n^2 I\right)^{-1}\mathbf{y}\,. \tag{40}$$

The resulting Bayesian estimation method is summarized in Algorithm 1. Steps 1 to 4 correspond to the predictive stage, where the predicted mean $\mathbf{x}_k^-$ and variance P_k^- of the state are calculated from the estimations obtained in the previous iteration, $\mathbf{x}_{k-1}$, P_{k-1}, and the action $\mathbf{u}_k$. Steps 1 and 4 are the standard EKF prediction steps, while in steps 2 and 3 the process noise covariance, Q_k, and the process Jacobian, A_k, are calculated. Steps 5 to 10 correspond to the correction stage, where the corrected mean $\mathbf{x}_k$ and covariance matrix P_k are calculated from the predicted ones and the new observation, $\mathbf{z}_k$. In steps 5 to 7, the expected observation, $\mathbf{h}_k$, the observation covariance matrix, R_k, and the observation

process Jacobian, H_k, are calculated. Steps 8 to 10 are the standard EKF correction equations, including the calculation of the Kalman gain matrix, K_k, in step 8.

Algorithm 1. GP-EKF

1. $\mathbf{x}_k^- = GP_\mu^{\mathbf{f}}\left([\mathbf{x}_{k-1}, \mathbf{u}_k]\right)$
2. $Q_k = GP_\Sigma^{\mathbf{f}}\left([\mathbf{x}_{k-1}, \mathbf{u}_k]\right)$
3. $A_k = \frac{\partial GP_\mu^{\mathbf{f}}}{\partial \mathbf{x}_{k-1}}\left([\mathbf{x}_{k-1}, \mathbf{u}_k]\right)$
4. $P_k^- = A_k P_{k-1} A_k^T + Q_k$
5. $\mathbf{h}_k = GP_\mu^{\mathbf{h}}\left(\mathbf{x}_k^-\right)$
6. $R_k = GP_\Sigma^{\mathbf{h}}\left(\mathbf{x}_k^-\right)$
7. $H_k = \frac{\partial GP_\mu^{\mathbf{h}}}{\partial \mathbf{x}_k}\left(\mathbf{x}_k^-\right)$
8. $K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$
9. $\mathbf{x}_k = \mathbf{x}_k^- + K_k\left(\mathbf{z}_k - \mathbf{h}_k\right)$
10. $P_k = \left(I - K_k H_k\right) P_k^-$

GP-EKF uses the independent MOGP algorithm described in subsection 1.6.4.4. Consequently, the covariance matrices Q_k and R_k are diagonal, i.e., no correlation between the noise components are considered. Furthermore, no information from the training data is used to infer the values of Q_k and R_k over the different regions of the state and state-action spaces. These covariance matrices (Q_k and R_k) will increase in some regions only due to the lack of training data.

1.6.6 Heteroscedastic Gaussian Processes for Regression

The GPs that consider an input-dependent noise are called *Heteroscedastic Gaussian Processes* (HGPs). Because of its relevance for the Thesis, a HGP method that estimates the noise variance using a different GP is described in the next subsection, while the HGP literature is reviewed in subsection 2.1.

1.6.6.1 Most-Likely Heteroscedastic Gaussian Processes

In this subsection, we will briefly describe the HGP method proposed in [36], called *Most-Likely HGP* (abbreviated ML-HGP in the present document). The basic idea behind ML-

HGPs is having two independent GPs, one for the mean and other for the variance. We will call them respectively *mean-GP* and *variance-GP* and note them respectively GP^m and GP^v. To guarantee the positivity of the variance predictions, variance-GP actually performs a regression not on the variance itself but on its logarithm. Then, for any input test, the variance is obtained from the exponential of the log-variance predicted by variance-GP. Let $\left\{\left(\mathbf{x}_i^m, y_i^m\right)\right\}$ and $\left\{\left(\mathbf{x}_i^v, y_i^v\right)\right\}$ denote the training sample sets of respectively mean-GP and variance-GP. Naturally, the training samples for mean-GP are the same as the HGP training samples, i.e., $\left\{\left(\mathbf{x}_i^m, y_i^m\right)\right\} = \left\{\left(\mathbf{x}_i, y_i\right)\right\}$. For convenience, the training inputs of variance-GP are selected to be equal as those of mean-GP, i. e. $\left\{\mathbf{x}_i^v\right\} = \left\{\mathbf{x}_i^m\right\}$.

1.6.6.1.1 Prediction

In ML-HGPs the predictive mean of the variance-GP is used as the noise variance by the mean-GP. In other words, for mean-GP, the term $\sigma_n^2 I$ in equations (27) and (28) is replaced by $\Sigma_n = diagMat\left(\left\{\exp\left(GP_\mu^v\left(x_i^v\right)\right)\right\}\right)$, where $diagMat\left(\mathbf{v}\right)$ is a diagonal matrix with the elements of the vector $\mathbf{v}$ in its diagonal. Additionally, the last term in equation (28), σ_n^2, which is related to the observational noise (which in the case of regular GPs is assumed stationary) is replaced by an input-dependent value, $\exp\left(GP_\mu^v\left(\mathbf{x}_*\right)\right)$, whose estimation is the objective of the regression performed by variance-GP.

Consequently, the predictive equations are the following:

$$ML\_HGP_\mu\left(\mathbf{x}_*\right) = GP_\mu^m\left(\mathbf{x}_*\right) = \left(\mathbf{k}_*^m\right)^T\left(K_X^m + \Sigma_n\right)^{-1} y_i^m, \tag{41}$$

$$\begin{aligned} ML\_HGP_\Sigma\left(\mathbf{x}_*\right) &= K^m\left(X_*, X_*\right) - \left(\mathbf{k}_*^m\right)^T\left(K_X^m + \Sigma_n\right)^{-1}\mathbf{k}_*^m + \exp\left(GP_\mu^v\left(\mathbf{x}_*\right)\right) \\ &= K^m\left(X_*, X_*\right) - \left(\mathbf{k}_*^m\right)^T\left(K_X^m + \Sigma_n\right)^{-1}\mathbf{k}_*^m + \exp\left(\left(\mathbf{k}_*^v\right)^T\left(K_X^v + \Sigma_n\right)^{-1} y_i^v\right) \end{aligned}. \tag{42}$$

Note the superscripts m and v in the covariance function and covariance matrices. They specify that the hyperparameters and the training data are selected from respectively mean-GP and variance-GP. [6]

1.6.6.1.2 Learning

The predictive distribution of the mean-GP is used to generate the training samples of the variance-GP. Therefore, both GPs are interdependent. This suggests that a possible training procedure alternates between learning the variance-GP hyperparameters with fixed mean-GP noise variances and vice versa. This is the main idea behind Most-Likely HGPs. The hyperparameters learning procedure used in Most-Likely HGPs is summarized in Algorithm 2.

In step 1 of Algorithm 2, the standard GP learning procedure, described in subsection 1.6.4.3, is used to learn the hyperparameters of mean-GP. Steps from 2.a to 2.d are repeated until convergence.

Algorithm 2. Most-Likely HGP hyperparameters learning.

1. Learn mean-GP from data using the standard GP learning procedure.
2. While not converged
    a. Estimate the training samples $\{y_i^v\}$ for variance-GP.
    b. Learn new hyperparameters for variance-GP.
    c. Use variance-GP to predict the noise variance values of mean-GP.
    d. Learn new hyperparameters for mean-GP

In step 2.a, the output training samples for variance-GP are generated using an estimation of the process variance. This variance is estimated by sampling from the predictive distribution of mean-GP. For each variance-GP training input, $\mathbf{x}_i^v$, the corresponding training output is calculated as:

[6] mean-GP and variance-GP have different hyperparameters and training sets and, thus, we must specify which sets of hyperparameters and training samples we are considering even when we are using the same covariance function.

$$y_i^v = \log\left(\frac{1}{N_s}\sum_{j=1}^{N_s}\frac{1}{2}\left(y_i - s_j^i\right)^2\right), \tag{43}$$

where $\left\{s_j^i\right\}$ is a set of N_s random values sampled from the predictive distribution, $N\left(ML\_HGP_\mu\left(\mathbf{x}_i^v\right), ML\_HGP_\Sigma\left(\mathbf{x}_i^v\right)\right)$, at $\mathbf{x}_i^v$.

In step 2.b, new hyperparameters for variance-GP are learned by means of the standard GP learning procedure, described in subsection 1.6.4.3.

In step 2.c, the noise variances for the training points of mean-GP are replaced by the variance that variance-GP predicts using the standard GP prediction procedure described in subsection 1.6.4.2. In other words, in the next step, the term $\sigma_n^2 I$ in equations (27) and (28) will be replaced by Σ_n.

In step 2.d, new hyperparameters are learned for mean-GP. The hyperparameter σ_n^2 is not learned because it is not used anymore. Consequently, the hyperparameter-learning equations (31) and (32) are replaced by the following:

$$\log p\left(y|X\right) = -\frac{1}{2}\mathbf{y}^T\left(K_X + \Sigma_n\right)^{-1}\mathbf{y} - \frac{1}{2}\log\left|K_X + \Sigma_n\right| - \frac{n}{2}\log 2\pi, \tag{44}$$

$$\frac{\partial \log p\left(y|X,\boldsymbol{\theta}\right)}{\partial \theta_j} = \frac{1}{2}\mathrm{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \left(K_X + \Sigma_n\right)\frac{\partial\left(K_X + \Sigma_n\right)}{\partial \theta_j}\right)\right). \tag{45}$$

ML-HGPs are only defined for one-dimensional outputs and there is no suggested procedure for expanding them to correlated multidimensional outputs.

1.7 Contributions

The present section outlines the original contributions made by this thesis work, as well as the publications on which I worked while being a doctorate student. The contributions of this work can be divided into those related to the correct estimation of the uncertainty in the models and those related to the convenient reduction of uncertainty in the state.

1.7.1 Estimation of the Uncertainty

This thesis work presents several original contributions regarding the correct estimation of the uncertainty in a mobile robot and, in addition, some of them are applicable to more

general scenarios. Given that the contributions in this area make it possible to better characterize the uncertainty in the actions and observations of a state estimator, they also improve the estimations of both the mean and the variance of the state.

The first contribution is an original HGP learning method. In contrast with the existent HGP learning methods, the one presented in this thesis simultaneously performs the estimation of the variance and its regression. This innovative method, called *Simultaneous Estimation and Regression of the Variance HGP* (SERV-HGP) not only extends ML-HGP to multiple dimensions but also presents a faster training procedure in the one-dimensional case with a comparable performance.

The second, and most important, contribution is the development of an innovative method for state estimation called *Heteroscedastic Gaussian Process Extended Kalman Filter* (HGP-EKF), which is an improvement as compared with the existent GP-EKF method. The main advantages of HGP-EKF over GP-EKF are:

- While GP-EKF is only able to estimate the variance of the noises from the uncertainty in the models coming from the lack of training data, HGP-EKF also conducts a variance regression using the variance expressed in the training data as a function of the state.
- In contrast with GP-EKF, HGP-EKF is able to learn the crossed components of covariance matrices and, therefore, HGP-EKF does not assume diagonal covariance matrices.
- HGP-EKF allows the use of additional parameters that can offer clues regarding how to improve both the process and observational model.

Finally, a specialized GP learning procedure is developed for angle outputs.

1.7.2 Convenient Reduction of the Uncertainty

The problem of reducing uncertainty is addressed by developing an active-vision method. This active-vision method is called *Task-Oriented Active Vision*. The main contribution of the task-oriented active-vision paradigm over other probabilistic methods discussed in the literature is that it explicitly intends to reduce the uncertainty components that are most

relevant to the task being executed. This focused reduction of uncertainty is achieved by taking into account a value function that is related to the task being executed.

1.7.3 Related Publications

There are two publications related to this thesis work (1 ISI and 1 LNCS). Both of them are related to the active vision system proposed in Chapter 3. In the following paragraphs, these publications are briefly presented.

In [15] an original active vision system called task-oriented active vision was presented for the first time. In this publication, an intuition-based optimality criterion (the minimum value variance) was discussed as a means to solve the problem of active vision.

In [16] task-oriented active vision was more formally presented. Additionally, an original optimality criterion (the maximum expected value) was introduced. Finally, in that work task-oriented active vision methods were compared to those based on information-theory, leading to the conclusion that the former methods outperform the latter ones when the task is not uncertainty reduction itself.

1.7.4 Other Publications

I contributed to several projects during my doctorate studies. In particular, I was a part of the Robotic Soccer Team of the Universidad de Chile. Several publications emerged from this work, and they are summarized in the following paragraphs, separated by topic.

1.7.4.1 Color Segmentation

In [17], a real time system to segment images using color segmentation is presented. The main innovation of this system is the use of the pixel's context to determine its color class from a set of a priori classes that were determined from the pixel color value.

An automatic color-calibration system is presented in [18]. This system has the ability to learn the mapping from color values to color classes with no a priori information using the spatial relationships among color classes in the color space.

1.7.4.2 Use of Context in Robot Vision

In [19] an original vision system that probabilistically integrates information from the spatiotemporal context to improve the detection and tracking of objects was presented. The

system is able to integrate information from the current image and previous images and commands. For each object, the past information is summarized in a Bayesian state estimator. These ideas were further developed in [20][21].

1.7.4.3 Localization and Tracking of Objects

A method for estimating the odometric error of a mobile robot was outlined in [22]. The system has many similarities to some Simultaneous Localization and Mapping (SLAM) systems. For instance, the system has independent Bayesian estimators for the odometry error and each object. The article proposes an egocentric viewpoint of the state estimation problem in a mobile robot.

1.7.4.4 Robot-Soccer Decision Making

A probabilistic decision-making method for robot soccer was proposed in [23]. The article presented a formal derivation of a decision-making system for a robot that must kick the ball. The system is able to work in real time and has the natural ability to select between kicking to the goal, making direct passes, passes in advance or clearing the ball. These decisions are made without the need of explicitly defining these actions. Additionally, the proposed system makes it possible to smoothly between offensive and defensive concerns.

An MDP-based integrated robot-soccer decision-making system was presented in [24]. This work proposed several improvements over [23], such as the following: (i) by using an MDP model this system is able to take into account several steps in advance instead of just one or two, and (ii) this system integrated the whole decision-making problem (kicking and not-kicking robots) instead of just considering the kicking-robot problem.

1.7.4.5 Evaluation of Methods

With the purpose of bringing tools for the evaluation and comparison of vision and state-estimation methods, the following two works were carried out. First, a benchmark for robot-soccer vision systems was introduced in [26]. Then, a ground-truth system for mobile robots was presented in [27]. This ground-truth system was used in this thesis to generate the ground-truth data used for evaluation of methods in real experiments.

1.8 Structure of the Document

The thesis is organized as follows. First, the present Chapter introduces the reader to the topic being addressed and to the thesis work itself. In particular, the objectives and hypotheses of this thesis are presented. Additionally, the background necessary for the reader to understand the methods discussed in this thesis is introduced. At the end of this Chapter, both the main contributions of the thesis, as well as the publications that helped to build the body of work in this thesis are discussed. Then, Chapter 2 presents an innovative method for estimating Heteroscedastic Gaussian Processes and the related literature in which this method is contextualized. Chapter 3 presents the methods that were developed in this thesis to estimate the partially-observable state of a system by learning the models and noise statistics from training data sets. In Chapter 4, an innovative task-oriented active-vision system is outlined. Chapter 5 presents the experimental results for the proposed methods and explains the discussion that arises from these results. Finally, Chapter 6 offers conclusions based on the methodologies developed and results obtained.

Chapter 2. Heteroscedastic Gaussian Processes

Some GPs cannot be assumed to have stationary noise. As such, it is convenient to have a regression method with input-dependent noise. The input-dependent-noise version of the GPs is the Heteroscedastic Gaussian Processes (HGPs). Section 2.1 reviews the existent HGP literature.

A novel procedure for the estimation of an HGP is presented in section 2.2. This procedure is based on ML-HGPs and with the purpose of making it suitable to multidimensional outputs and non-diagonal covariance matrices, several modifications to the original method are introduced.

2.1 Literature Review

There are a large number of publications regarding different theoretical and practical aspects of GPs. However, we will only mention some articles related to the problem of regression with input-dependent noises. The input-dependent noise regression problem involves the estimation of the noise mean and variance. In the literature, there are solutions in which this estimation is made separately for the mean and the variance and other solutions in which both are estimated jointly.

One idea belonging to the first group is that of having an independent GP for the noise variance. This idea was introduced in [34]. In that work, Markov-chain Monte Carlo methods are used to estimate the posterior distribution of the noise variance. Most recently, an iterative procedure to estimate alternately the mean and variance GPs was proposed in [36]. In this thesis, we propose an HGP method that is based on [36].

Examples of joint estimation of the mean and variance can be found in [35] and [37].

It is also worth mentioning the *Gaussian Copula Process Volatility* model [38], which is used to predict the variance in the future of a timed signal from past observations of that signal.

2.2 Simultaneous Estimation and Regression of the Variance HGP

HGPs are able to produce an estimation of a non-stationary stochastic process and, for each input, give an estimation of the output's mean and variance. In this subsection, an innovative HGP estimation procedure called *Simultaneous Estimation and Regression of the Variance HGP* (SERV-HGP) is presented. This procedure is based on ML-HGPs [36], where, as it was already proposed in [34], an independent GP is used for the variance. SERV-HGP includes several modifications to ML-HGP with the purpose of enabling the method to handle multidimensional and correlated outputs. In order to allow the multidimensionality of the outputs, the main modification is that in SERV-HGP both mean-GP and variance-GP are MOGPs (see subsection 1.6.4.4). Thus, we will denote mean-GP and variance-GP as $\mathbf{MOGP}^{m}\left(\mathbf{x}_{*}\right)$ and $\mathbf{MOGP}^{v}\left(\mathbf{x}_{*}\right)$ respectively, and their one-dimensional component GPs as $\left\{\mathrm{GP}^{m,j}\right\}$ and $\left\{\mathrm{GP}^{v,j}\right\}$ respectively.

SERV-HGPs are able to estimate the crossed components of the predictive covariance matrix. However, the correlation between output variables is not considered in the mean-GP, which is a potential improvement in relation to the method presented.

As described above, the procedure followed by ML-HGP and other related methods in order to solve the dependent-noise regression problem is to separately: (i) calculate the variance estimate for each training point, and (ii) solve the regression problem for the log of the variance. One of the main features of SERV-HGP, from which it derives its name, is its ability to simultaneously perform the estimation and regression of the process variance. The proposed procedure for accomplishing this task is based on the observation that the variance (or the covariance matrix, in the case of multidimensional outputs) itself can be defined as an expectation:

$$\mathrm{cov}\left(\mathbf{y}\right)=E\left(\left(\mathbf{y}-E\left(\mathbf{y}\right)\right)\left(\mathbf{y}-E\left(\mathbf{y}\right)\right)^{T}\right). \tag{46}$$

The idea behind SERV-HGP is to use the GP inference capabilities over the training data to simultaneously estimate this variance at each point and perform the regression process. As

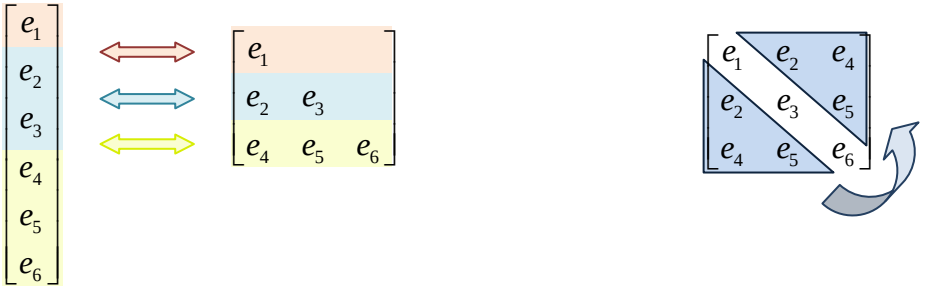
such, in SERV-HGP, the random variable whose mean is being estimated by variance-GP is the squared deviation of the output $\mathbf{y}$ from its mean:

$$\left(\mathbf{y}-E\left(\mathbf{y}\right)\right)\left(\mathbf{y}-E\left(\mathbf{y}\right)\right)^{T}. \tag{47}$$

Additionally, SERV-HGP extends ML-HGPs to multidimensional correlated outputs by defining an ML-HGP for each dimension and additionally estimating the crossed components of the covariance matrix. For any test point, $\mathbf{x}_*$, we can define a function with matrix output, $\bar{\Sigma}'^{\mathbf{y}}\left(\mathbf{x}_*\right)$, as a by-entry regression of the covariance matrix of the correspondent output $\mathbf{y}$:

$$\bar{\Sigma}'^{\mathbf{y}}\left(\mathbf{x}_*\right)=R_{\mathbf{vec}\to Sym}\left(\mathbf{MOGP}_{\mu}^{v}\left(\mathbf{x}_*\right)\right), \tag{48}$$

where $M=R_{\mathbf{vec}\to Sym}\left(\mathbf{v}\right)$ is the $N_M\times N_M$ symmetric matrix resulting from reshaping the vector $\mathbf{v}$ into a triangular matrix and then copying the non-diagonal elements to the empty triangle to ensure M being symmetric. Note that the dimension of $\mathbf{v}$ must be $N_{\mathbf{v}}\left(N_{\mathbf{v}}+1\right)/2$. An inverse function, $\mathbf{v}=R_{Sym\to\mathbf{vec}}\left(M\right)$, can be also defined. Figure 3 illustrates the reshaping pattern used by these functions.



(a) (b)

Figure 3. Example of the reshaping pattern of functions $R_{\mathbf{vec}\to Sym}\left(\mathbf{v}\right)$ and $R_{Sym\to\mathbf{vec}}$ for a 3x3 matrix. (a) The elements of the vector are copied from/to the upper triangle in the symmetric matrix. (b) In the case, of $R_{\mathbf{vec}\to Sym}\left(\mathbf{v}\right)$, the elements of the lower triangle are copied to the upper one maintaining the symmetry.

Note that $\bar{\Sigma}'^{\mathbf{y}}\left(\mathbf{x}_*\right)$ is not guaranteed to be positive definite and, consequently, it is not a well-defined covariance matrix. With the purpose of having a well-defined covariance

matrix that is close to $\bar{\Sigma}'^{\mathbf{y}}(\mathbf{x}_*)$ (or equal if possible) we define the *predicted-from-data covariance matrix*, $\bar{\Sigma}^{\mathbf{y}}(\mathbf{x}_*)$, as:

$$\bar{\Sigma}^{\mathbf{y}}(\mathbf{x}_*) = \max^{\lambda}\left(\bar{\Sigma}'^{\mathbf{y}}(\mathbf{x}_*), \lambda_{\min}\right), \tag{49}$$

with $\lambda_{\min}$ an arbitrary small positive value and the function $\max^{\lambda}$ defined as:

$$\max^{\lambda}(A, \lambda_{\min}) = V\Lambda' V^{-1}, \tag{50}$$

where $A = V\Lambda V^{-1}$ is the eigenvector decomposition of A and

$$\Lambda' = diagMat\left(\mathbf{max}\left(diag(\Lambda), \lambda_{\min}\right)\right), \tag{51}$$

with $diag(M)$ a vector containing the diagonal of the squared matrix M, and $\mathbf{max}(\mathbf{v}, \alpha)$ a function that returns a vector with the same dimensionality as $\mathbf{v}$, with coordinates:

$$\left[\mathbf{max}(\mathbf{v}, \alpha)\right]_i = \max(\mathbf{v}_i, \alpha). \tag{52}$$

Note that $\bar{\Sigma}^{\mathbf{y}}(\mathbf{x}_*)$ is ensured to be positive definite and, provided $\bar{\Sigma}'^{\mathbf{y}}(\mathbf{x}_*)$ is positive definite and $\lambda_{\min}$ is small enough, $\bar{\Sigma}^{\mathbf{y}}(\mathbf{x}_*) = \bar{\Sigma}'^{\mathbf{y}}(\mathbf{x}_*)$.

2.2.1.1 Prediction

Analogously to ML-HGP, in SERV-HGP the predictive mean of the variance-GP is used as the noise variance by the mean-GP. In other words, for $\mathrm{GP}^{m,j}$, the term $\sigma_n^2 I$ in equations (38) and (39) is replaced by $\bar{\Sigma}^{\mathbf{y}}_{diag,j}\left(\{\mathbf{x}_i^v\}\right)$, where $\bar{\Sigma}^{\mathbf{y}}_{diag,j}\left(\{\mathbf{x}_i\}\right)$ is a diagonal matrix whose diagonal elements are defined by:

$$\left[\bar{\Sigma}^{\mathbf{y}}_{diag,j}\left(\{\mathbf{x}_i\}\right)\right]_{k,k} = \left[\bar{\Sigma}^{\mathbf{y}}(\mathbf{x}_k)\right]_{j,j}. \tag{53}$$

Additionally, in equation (39), the observational noise term, $\sigma_n^2 I_{N_{\mathbf{y}} \times N_{\mathbf{y}}}$, (which is assumed to be stationary and does not consider the crossed components of the covariance matrix) is replaced for the input-dependent covariance term, $\bar{\Sigma}^{\mathbf{y}}(\mathbf{x}_*)$, which does consider the crossed components of the covariance matrix.

Consequently, the predictive equations are the following:

$$SERV\_HGP_{\mu}\left(\mathbf{x}_{*}\right)=\begin{bmatrix} Agg\left(\left\{y_{1,j}\right\}\right)\left(K_X^1+\bar{\Sigma}^{\mathbf{y}}_{diag,j}\left(\left\{\mathbf{x}_i^v\right\}\right)\right)^{-1}K_*^1 \\ \vdots \\ Agg\left(\left\{y_{N_{\mathbf{y}},j}\right\}\right)\left(K_X^{N_{\mathbf{y}}}+\bar{\Sigma}^{\mathbf{y}}_{diag,j}\left(\left\{\mathbf{x}_i^v\right\}\right)\right)^{-1}K_*^{N_{\mathbf{y}}} \end{bmatrix}, \tag{54}$$

$$SERV\_HGP_{\Sigma}\left(\mathbf{x}_{*}\right)=\begin{bmatrix} GP_{\Sigma'}^{m,1}\left(\mathbf{x}_{*}\right) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & GP_{\Sigma'}^{m,N_{\mathbf{y}}}\left(\mathbf{x}_{*}\right) \end{bmatrix}+\bar{\Sigma}^{\mathbf{y}}\left(\mathbf{x}_{*}\right). \tag{55}$$

The term $GP_{\Sigma'}^{m,j}\left(\mathbf{x}_{*}\right)=K^{m,j}\left(X_{*},X_{*}\right)-\left(K_{*}^{m,j}\right)^{T}\left(K_X^{m,j}+\bar{\Sigma}^{\mathbf{y}}_{diag,j}\left(\left\{\mathbf{x}_i^v\right\}\right)\right)^{-1}K_{*}^{m,j}$ accounts for the component of the covariance matrix provided by the lack of training data in the vicinity of $\mathbf{x}_{*}$.

2.2.1.2 Learning

Algorithm 3 summarizes the resulting learning method. Step 1 of Algorithm 3 is analogous to that of Algorithm 2, i.e., the standard GP learning procedure, described in subsection 1.6.4.3, is used to learn the hyperparameters of each of the one-dimensional component GPs of mean-GP. Steps from 2.a to 2.d are repeated until convergence.

Algorithm 3. SERV-HGP hyperparameters learning.

1. Learn mean-GP from data using the standard GP learning procedure.
2. While not converged
    a. Calculate the training samples $\left\{\mathbf{y}_i^v\right\}$ for variance-GP.
    b. Learn new hyperparameters for variance-GP.
    c. Use variance-GP to predict the noise variance values of mean-GP.
    d. Learn the hyperparameters of mean-GP

In step 2.a, the output training samples for variance-GP are generated. Differing from ML-HGP, the training samples are set to:

$$\mathbf{y}_i^v=R_{Sym\rightarrow\mathbf{vec}}\left(\left(\mathbf{y}_i-SERV\_HGP_{\mu}\left(\mathbf{x}_i^v\right)\right)\left(\mathbf{y}_i-SERV\_HGP_{\mu}\left(\mathbf{x}_i^v\right)\right)^{T}\right). \tag{56}$$

Where $SERV\_HGP_{\mu}\left(\mathbf{x}_i^v\right)$, the predictive mean at $\mathbf{x}_i^v$, is used to approximate $E\left(\mathbf{y}_i\right)$.

Analogously to ML-HGP, in step 2.b, new hyperparameters for each of the one-dimensional component GPs of variance-GP are learned by means of the MOGP learning procedure, described in subsection 1.6.4.4.

In step 2.c, the noise variances for the training points of mean-GP are replaced by the variance that variance-GP predicts using the standard GP prediction procedure described in subsection 1.6.4.2. If necessary, these values are fixed to be positive. In other words, the term $\sigma_n^2 I$ in equations (25) and (26) is replaced by $\bar{\Sigma}^{\mathbf{y}}_{diag,j}\left(\left\{\mathbf{x}_i^v\right\}\right)$.

In step 2.d, the hyperparameters of mean-GP are relearned without varying the noise variance.

Chapter 3. HGP-Extended Kalman Filter

As Ko et al. already suggested in [89], one of the main flaws of GP-BayesFilters is that they do not learn the variances of the noises from the data. Instead, they just use the prediction variance from GP's to build the variances of the noises. Furthermore, they do not take into consideration the crossed components of the noises' covariance matrices, i.e., they assume them to be diagonal.

Section 3.2 describes the core of the proposed methodology, which is a Bayesian filtering methodology that is based on GP-EKF. The main improvement over GP- EKF proposed in this work is the use of an HGP to estimate the means and covariance matrices of the observational and process noises. By using an HGP instead of a GP, the system has the ability to learn not only the component of the variance that comes from the lack of data, but also the component that is expressed in the data. Furthermore, the system is able to learn the crossed components of the covariance matrix. Additionally, section 3.3 proposes an innovative methodology that takes into consideration additional information in order to reduce the inherent uncertainty in the process and observational models. Finally, a novel GP-based regression methodology for the learning functions with angle outputs is presented in section 3.4.

3.1 Literature Review

3.1.1 Bayesian State Estimation

For the general case, all the Bayesian filters reviewed in this section constitute approximations that lead to suboptimal estimations that are practically implementable in a computer. There are two main categories that have dominated the literature on Bayesian state estimation in the last decades: *Kalman filters* and *particle filters*. There are some other methods that are based on combinations of filters from these two categories. They can be called *hybrid filters*. Finally, the Markov grids do not fit in any of these categories. Therefore, they will be described separately.

3.1.1.1 Markov Grids

The *Markov Grids* [58] method consists of discretizing the state space in a grid. The selected grid must be fine enough to represent the state space. The grid is fixed and usually equispaced; thus, some regions of the space are not able to be explored and the method remains with a fixed resolution. For high dimensional state spaces, this method has significant requirements regarding computational memory and processing in order to achieve a reasonable resolution. This undesirable feature is known as the *dimensionality curse* [59]. A better state space discretization is provided by random sampling methods since they can adjust the representation of the state space to have more resolution in the most probable regions.

3.1.1.2 Kalman Filters

Kalman filters are state estimators based on linear operations of the state mean and covariance estimators. By construction, the original *Kalman filter* (KF) [49] minimizes the mean squared error (MSE) in the state estimation for linear systems. For this reason, the Kalman filter is theoretically optimal using the criterion of minimal MSE (MMSE). One of the strongest criticisms that the KF has received is that it assumes gaussianity of the noises, which is often not the case. Actually, the lack of gaussianity of the noises is not a problem for the KF, but rather, for interpretations of its estimates. The KF is able to estimate the mean and variance of the belief without bias even when the noises are not gaussian. The problem appears when the belief is multimodal. In such cases, the mean offers poor information about the belief, especially if one is interested in the most probable state or *maximum a posteriori* (MAP). Some methods described in the next subsections deal better with multimodal beliefs. Given the assumed linearity of the models, the Kalman filter is inapplicable for non-linear systems. To apply the linear operatory of the Kalman filter to non-linear systems, it is possible to make a linearization of the non-linear models around the state estimate. This technique is known as the *Extended Kalman Filter* (EKF) [50] and has been successfully applied to diverse state estimation problems over the last decades. The linearization of the models made by the EKF derives into a linear propagation of the mean and covariance of the random variables. This linear propagation may cause divergences on non-linear systems. For that reason, some methods that try to propagate in

different ways these statistics have been developed. One of these methods is the *Iterative Extended Kalman Filter* (IEKF) [51], which make several iterations of the corrective stage updating in every iteration the point the linearization is made around. There are also methods that propagate the covariances using a discrete representation of the pdf's obtained by deterministic sampling. These methods have been called *Linear Regression Kalman Filters* (LRKF) [52] or *Sigma-Point Kalman Filters* (SPKF) [53]. Currently, the most popular SPKF are: the *Unscented Kalman Filter* (UKF) [54], which is based on the Unscented Transform (UT), the *Central Difference Filter* (CDF) [55], and the *Divided Difference Filters* (DD1 y DD2) [56]. A SPKF based on *Singular Value Decomposition* (SVD) [48] has also been proposed. *Square Root Unscented Kalman Filter* (SR-UKF) [57] constitutes an efficient implementation of UKF. In an effort to reduce computational complexity, SR-UKF maintains the square root of the covariance matrix of the estimator.

3.1.1.3 Particle Filters

Particle filters are state estimators that approximate the pdf's through random sampling. By means of maintaining pdf samples instead of a parametric representation, the task of propagating a random variable through arbitrary functions, which can be hard for some parametric representations, becomes trivial. It is possible to obtain a representation of the propagated pdf by applying the desired function to each sample. The particle filters have been developed independently in various fields with different names, such as *Bootstrap Filter* [60], *CONDENSATION* [61][62] and *Monte Carlo Filter* [62]-[66]. There are different techniques for updating the posterior distribution from an observation. Of these, it is worth mentioning *Rejection Sampling* [67], *Importance Sampling* [68], *Sampling/Importance Resampling* [69] and *MCMC* [70][71]. In contrast with Markov grids, particle filters do not suffer from the curse of dimensionality, but they can converge very slowly in high dimensional spaces. Thus, they offer a tradeoff between convergence speed and computational complexity that can be easily balanced through the number of particles.

3.1.1.4 Hybrid Filters

Hybrid Filters intend to obtain the advantages of the Kalman and particle filters by combining them in different ways. Examples of hybrid filters are: *Rao-Blackwellized*

Filters (RBF), the particle filters that use Kalman to obtain the proposed distribution and the *Gaussian Sum Kalman Filters*.

The Rao-Blackwellized filters [72] exploit the model structure with the purpose of improving the efficiency and reducing the variance of the estimator. Some Rao-Blackwellized filters decompose the state space dimensions into two parts. One of them, which is assumed to be linear, is calculated using the Kalman Filter, while the other is inferred using particle filters.

Some particle filters use Kalman filters to estimate the proposed distribution. In particular, an EKF or UKF can be used to generate and propagate the proposed distribution [73]. In the first case, the resulting filter is known as *Extended Kalman Particle Filter* (EKPF) and, in the second case, as *Unscented Particle Filter* (UPF). There is an efficient variant of the UPF, which uses an SR-UKF to generate the proposed distribution. This method is known as *Square-Root Unscented Particle Filter* (SRUPF) [74].

The Gaussian Sum Kalman Filters intend to overcome the poor performance of Kalman filters when facing multimodal pdf's. The *Gaussian Sum Filter* [75][76] uses several independent EKF's to face multimodality by means of letting each mode of the belief be tracked by a different EKF. As an output, the Gaussian sum filter offers a linear combination of the outputs of these EKF's.

3.1.1.5 Applications in Mobile Robotics

There are diverse applications of state estimators in mobile robotics. Among these applications, those that have been investigated the most are *Self-Localization* and *Simultaneous Localization And Mapping* (SLAM). There are also works in which the static and/or kinematic state of other objects are estimated.

In mobile robotics, self-localization is the process of determining the pose of the robot, which is defined as its position and orientation relative to a fixed reference system in the environment. There is a huge body of publications regarding mobile localization using Bayesian filters. Some of these methods are compared in [77]. There are examples of self-localization systems based on Markov grids [58], EKF [78][80], particle filters [81]-[83] and combinations of the two [84].

SLAM is a very important problem for mobile robotics and it is the extension of the self-localization problem in cases where there is no previously known map of the environment. In this case, the robot must simultaneously estimate two variables: the robot pose and the environment map. There is also a large number of publications that outline solutions to the SLAM problems that are based on Bayesian filters. The approach presented in [85], which uses EKF to solve the SLAM problem, has been followed by several works. It is worth mentioning the methods *FastSLAM* [86] and *FastSLAM 2.0* [87], which use Rao-Blackwellized particle filters to solve the SLAM problem.

The tracking of moving objects is a very important problem in mobile robotics because it greatly influences the ability of a mobile robot to navigate and interact with the environment. An example of tracking system for moving objects is discussed in [88]. In that work, a Rao-Blackwellized particle filter is applied by dividing the state into a linear and a non-linear part.

3.1.2 Gaussian Process Bayesian Filtering

Ko et al. proposed *Gaussian-Process Bayes Filters* (GP-BayesFilters) [89][90] for state estimation. The main idea behind GP-BayesFilters is the use of the regression abilities of GPs to learn the models used in a Bayesian filter. The main limitation of GP-BayesFilters, which was already pointed out by Ko et al., is that they do not learn the covariance matrices of the noises from the dispersion of the training data. Instead, they only take into consideration the prediction variance provided by the GPs, which correspond to the uncertainty in the models. This prediction variance depends only the lack of training data in the vicinity of the tested point and not on the variability of the training data. Furthermore, GP-BayesFilters assume that the covariance matrices of the noises are always diagonal. This assumption may lead to a poorer performance.

In this thesis, a novel set of methods called HGP-BayesFilters is proposed. HGP-BayesFilters overcome the previously mentioned flaws of regular GP-BayesFilters by using HGPs for both the observational and process models instead of regular GPs.

Recently, *Analytic Moment-based Gaussian Process Filtering* (GP-ADF) [91] was proposed, where exact expressions for the mean and the covariance matrix are provided for both the prediction and the correction steps.

3.2 HGP-EKF

This thesis proposes a method that uses the non-linear regression capabilities of GPs for learning the models used in Bayes filters. This method is called HGP-EKF and it is an improvement in comparison with the existent GP-EKF method. In contrast with GP-EKF, HGP-EKF uses an HGP to represent the observational and process models. By doing so, it is able to consider not only the component of the noises' variance coming from the lack of training data, but also the component that is expressed in the training data.

The SERV-HGP method is selected for handling the required HGPs. The resulting method is summarized in Algorithm 4.

4. HGP-EKF

1. $\mathbf{x}_k^- = SERV\_HGP_\mu^{\mathbf{f}}\left([\mathbf{x}_{k-1}, \mathbf{u}_k]\right)$
2. $Q_k = SERV\_HGP_\Sigma^{\mathbf{f}}\left([\mathbf{x}_{k-1}, \mathbf{u}_k]\right)$
3. $A_k = \frac{\partial SERV\_HGP_\mu^{\mathbf{f}}}{\partial \mathbf{x}_{k-1}}\left([\mathbf{x}_{k-1}, \mathbf{u}_k]\right)$
4. $P_k^- = A_k P_{k-1} A_k^T + Q_k$
5. $\mathbf{h}_k = SERV\_HGP_\mu^{\mathbf{h}}\left(\mathbf{x}_k^-\right)$
6. $R_k = SERV\_HGP_\Sigma^{\mathbf{h}}\left(\mathbf{x}_k^-\right)$
7. $H_k = \frac{\partial SERV\_HGP_\mu^{\mathbf{h}}}{\partial \mathbf{x}_k}\left(\mathbf{x}_k^-\right)$
8. $K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$
9. $\mathbf{x}_k = \mathbf{x}_k^- + K_k\left(\mathbf{z}_k - \mathbf{h}_k\right)$
10. $P_k = \left(I - K_k H_k\right) P_k^-$

The calculation of the Jacobians is trivial from equation 54:

$$\frac{\partial SERV\_HGP_\mu}{\partial \mathbf{x}_*}\left(\mathbf{x}_*\right) = \left(\frac{\partial K_*^m}{\partial \mathbf{x}_*}\right)^T \left(K_X^m + \Sigma_n\right)^{-1} \mathbf{y}_i^m . \tag{57}$$

3.3 Additional Model Parameters

Sometimes, the robot can sense variables that are not directly part of the observation of the action, but can offer some information regarding the statistics of the observational and/or

process noises. We will call these variables *additional model parameters* or simply *additional parameters*. Additional parameters for the process model could include several measurements from sensors in the actuators such as current, temperature, pressure or position sensors. Additional parameters for the observational model could include information from the sensors that could be relevant to estimate the observational noise parameters. For instance, the velocity of a robot's neck could help to estimate the accuracy of a measurement made from the robot's head.

In order to better characterize the inherent uncertainty in the models, this thesis proposes a methodology for using the additional model parameters in the process and observational models.

Then, we can define $\mathbf{a}_k^f$ and $\mathbf{a}_k^h$ as respectively the process and the observational additional vectors at instant k. Consequently, the process and observational models may be redefined as:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k^f, \mathbf{w}_k\right), \tag{58}$$

$$\mathbf{h}\left(\mathbf{x}_k, \mathbf{a}_k^h, \mathbf{v}_k\right). \tag{59}$$

It is expected that the relationship between the additional parameters and the noise parameters will be learned by an HGP. In the HGP-EKF algorithm, this can be easily achieved by augmenting the input of the GPs by adding the additional parameters to the training and testing inputs.

In Chapter 5, examples of additional parameters for both the process and the observational models are tested on simulated and real robots.

3.4 Angle Gaussian Processes

In this section, a modification to regular GPs is introduced that leads to an improvement of the performance of the regression when the output is an angle. Let us call a GP modified in such a way an *Angle Gaussian Process* (Angle-GP). Although the case of an output variable being an angle is very particular when contextualized on the general regression problem, in the context of mobile robotics this case is relevant. This relevance stems from

the fact that the orientation of a robot and other objects is a key variable in the processes of estimating the robot's localization and the poses of other objects.

The motivation for this procedure is that regular GPs perform poorly when solving a regression problem with an output variable that corresponds to an angle. The reason for this poor performance appears to be the fact that GPs always bring as an output a linear combination of the training outputs. However, the linear combination of angles is not a good way to combine them. The methodology proposed here intends to overcome this problem by emulating a common strategy for combining angles: (i) getting the sine and cosine of each angle, (ii) linearly combining the obtained sine and cosine values independently, and, (iii) use the arctangent function to get the combined angle form the combined sine and cosine.

Before introducing the Angle-GP methodology, we will introduce an intermediate step, which we will call *SinCos-GP*. SinCos-GP tries to solve the aforementioned angle regression problem by separating it into two independent regression problems: one for the sine and the other for the cosine of the angle. Two independent regular GPs are trained using the standard GP training procedure: one, $GP^{\sin}$, for the sine of the angle and other, $GP^{\cos}$, for the cosine. Then, for each test input SinCos-GP will predict independently the sine and the cosine of the output angle and will bring as a predictive output for the angle the result of the atan2 function applied to the predicted sine and cosine.

Angle-GP is an improvement over SinCos-GP in the sense that it contemplates the correlation between the sine and the cosine of the angle. For this purpose, the same hyperparameters are learned for both $GP^{\sin}$ and $GP^{\cos}$ for considering the influence of both the sine and the cosine in the learning process, a modification in the log marginal likelihood definition in equation (31) is introduced. The new definition of the log marginal likelihood is the following:

$$\log p\left(y\middle|X\right) = \\ -\frac{1}{2}\mathbf{y}_s^T\left(K_X+\sigma_n^2 I\right)^{-1}\mathbf{y}_s-\frac{1}{2}\mathbf{y}_c^T\left(K_X+\sigma_n^2 I\right)^{-1}\mathbf{y}_c-\frac{1}{2}\log\left|K_X+\sigma_n^2 I\right|-\frac{n}{2}\log 2\pi \quad (60)$$

Where $\mathbf{y}_s$ and $\mathbf{y}_c$ are vectors with the respective sines and cosines of the elements of $\mathbf{y}$, i.e.:

$$\left(\mathbf{y}_c\right)_{(i)} = \cos\left(\mathbf{y}_{(i)}\right), \tag{61}$$

$$\left(\mathbf{y}_s\right)_{(i)} = \sin\left(\mathbf{y}_{(i)}\right). \tag{62}$$

And $\left(\mathbf{y}_c\right)_{(i)}$, $\left(\mathbf{y}_s\right)_{(i)}$ and $\mathbf{y}_{(i)}$ are respectively the i^{th} element of $\mathbf{y}_c$, $\mathbf{y}_s$ and $\mathbf{y}$.

Just like SinCos-GP, when a test input $\mathbf{x}$ is presented, Angle-GP uses $GP^{\sin}$ and $GP^{\cos}$ to independently predict outputs for the sine and cosine of the angle and then the predicted output is calculated applying the atan2 function to them.

3.5 Composite Models

Sometimes it can be useful to learn not the models themselves, but rather some related functions. In particular, we will explore the situation in which it is convenient to define the model as composite functions:

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right) = f_1\left(f_2\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right), \mathbf{x}_k, \mathbf{u}_k\right), \tag{63}$$

$$\mathbf{z}_k = h\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k\right) = h_1\left(h_2\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k\right), \mathbf{x}_k, \mathbf{u}_k\right). \tag{64}$$

In this case, the relevant Jacobians of the models can be calculated as:

$$\begin{aligned} A_k = \frac{\partial f}{\partial \mathbf{x}_k}\left(\mathbf{x}_k, \mathbf{u}_{k+1}, 0\right) = \frac{\partial f_1\left(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\right)}{\partial \mathbf{p}_1}\left(f_2\left(\mathbf{x}_k, \mathbf{u}_k, 0\right), \mathbf{x}_k, \mathbf{u}_{k+1}\right)\frac{\partial f_2}{\partial \mathbf{x}_k}\left(\mathbf{x}_k, \mathbf{u}_k, 0\right) \\ + \frac{\partial f_1\left(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\right)}{\partial \mathbf{p}_2}\left(f_2\left(\mathbf{x}_k, \mathbf{u}_k, 0\right), \mathbf{x}_k, \mathbf{u}_{k+1}\right) \end{aligned}, \tag{65}$$

$$W_k = \frac{\partial f}{\partial \mathbf{w}_k}\left(\mathbf{x}_k, \mathbf{u}_{k+1}, 0\right) = \frac{\partial f_1\left(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\right)}{\partial \mathbf{p}_1}\left(f_2\left(\mathbf{x}_k, \mathbf{u}_k, 0\right), \mathbf{x}_k, \mathbf{u}_{k+1}\right)\frac{\partial f_2}{\partial \mathbf{w}_k}\left(\mathbf{x}_k, \mathbf{u}_k, 0\right), \tag{66}$$

$$\begin{aligned} H_k = \frac{\partial h}{\partial \mathbf{x}_k}\left(\mathbf{x}_k, \mathbf{u}_k, 0\right) = \frac{\partial h_1\left(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\right)}{\partial \mathbf{p}_1}\left(h_2\left(\mathbf{x}_k, \mathbf{u}_k, 0\right), \mathbf{x}_k, \mathbf{u}_k\right)\frac{\partial h_2}{\partial \mathbf{x}_k}\left(\mathbf{x}_k, \mathbf{u}_k, 0\right) \\ + \frac{\partial h_1\left(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\right)}{\partial \mathbf{p}_2}\left(h_2\left(\mathbf{x}_k, \mathbf{u}_k, 0\right), \mathbf{x}_k, \mathbf{u}_k\right) \end{aligned}, \tag{67}$$

$$V_k = \frac{\partial h}{\partial \mathbf{v}_k}\left(\mathbf{x}_k, \mathbf{u}_k, 0\right) = \frac{\partial h_1\left(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\right)}{\partial \mathbf{p}_1}\left(h_2\left(\mathbf{x}_k, \mathbf{u}_k, 0\right), \mathbf{x}_k, \mathbf{u}_k\right)\frac{\partial h_2}{\partial \mathbf{v}_k}\left(\mathbf{x}_k, \mathbf{u}_k, 0\right). \quad (68)$$

Given these definitions, it is possible to define modified versions of GP-EKF and HGP-EKF, where the process model and/or the observational model are defined as composite functions. In this case the GP or HGP functions take the place of the f_2 and h_2 functions in equations (63) to (66).

3.5.1 Special Cases

There are some special cases in which the use of composite functions is interesting. One of them, already mentioned in [89], is when a theoretical model exists that can be integrated into the GP-EKF method. This integration is realized by permitting the GPs to learn the error in the theoretical models instead of learning the models themselves. In this case, the models take the form:

$$f\left(\mathbf{x}, \mathbf{u}, \mathbf{w}\right) = f_{th}\left(\mathbf{x}, \mathbf{u}, \mathbf{w}\right) + GP_{\mu}^{\Delta \mathbf{f}}\left(\left[\mathbf{x}, \mathbf{u}\right]\right), \quad (69)$$

$$h\left(\mathbf{x}, \mathbf{u}, \mathbf{v}\right) = h_{th}\left(\mathbf{x}, \mathbf{u}, \mathbf{v}\right) + GP_{\mu}^{\Delta \mathbf{h}}\left(\left[\mathbf{x}, \mathbf{u}\right]\right). \quad (70)$$

Where f_{th} and h_{th} are respectively the process and observational theoretical models whose respective errors are learned by the GPs, $GP_{\mu}^{\Delta \mathbf{f}}$ and $GP_{\mu}^{\Delta \mathbf{h}}$, which are trained with samples of the error of the theoretical models. The Jacobians can be calculated using equations (65) and (66).

For the problems of robot localization and/or local tracking, it is interesting to use the GPs to learn the robot dynamics in a relative form instead of directly making them learn the process model. This simplification makes sense when the environment is homogeneous enough to allow that the behavior of the robot dynamics be reasonably independent of the robot position. The robot dynamics can be defined as a function (see subsection 1.6.1 for the definition of $T_{\cdot \to \cdot}\left(\cdot\right)$ and the derivation of its Jacobians):

$$f_{dyn}\left(\mathbf{u}_{k+1}, \mathbf{w}_k\right) = T_{abs \to rel}\left(\mathbf{x}_k, \mathbf{x}_{k+1}\right). \quad (71)$$

Then, the process model for the localization, f_{Loc}, and the process model for the object tracking, f_{OT}, can be defined as:

$$f_{Loc}\left(\mathbf{x},\mathbf{u},\mathbf{w}\right)=T_{rel\to abs}\left(\mathbf{x},\, f_{dyn}\left(\mathbf{u},\mathbf{w}\right)\right), \tag{72}$$

$$f_{OT}\left(\mathbf{x},\mathbf{u},\mathbf{w}\right)=T_{abs\to rel}\left(\, f_{dyn}\left(\mathbf{u},\mathbf{w}\right),\mathbf{x}\right). \tag{73}$$

Consequently, the process Jacobians have the form:

$$A_{Loc,k}=\frac{\partial f_{Loc}}{\partial \mathbf{x}}\left(\mathbf{x}_k,\mathbf{u}_{k+1},0\right)=\frac{\partial T_{rel\to abs}\left(\mathbf{s},\mathbf{p}\right)}{\partial \mathbf{s}}\left(\mathbf{x}_k,\, f_{dyn}\left(\mathbf{u}_{k+1},0\right)\right), \tag{74}$$

$$W_{Loc,k}=\frac{\partial f_{Loc}}{\partial \mathbf{w}}\left(\mathbf{x}_k,\mathbf{u}_{k+1},0\right)=\frac{\partial T_{rel\to abs}\left(\mathbf{s},\mathbf{p}\right)}{\partial \mathbf{p}}\left(\mathbf{x}_k,\, f_{dyn}\left(\mathbf{u}_{k+1},0\right)\right)\frac{\partial f_{dyn}}{\partial \mathbf{w}}\left(\mathbf{u}_{k+1},0\right), \tag{75}$$

$$A_{OT,k}=\frac{\partial f_{OT}}{\partial \mathbf{x}}\left(\mathbf{x}_k,\mathbf{u}_{k+1},0\right)=\frac{\partial T_{abs\to rel}\left(\mathbf{s},\mathbf{p}\right)}{\partial \mathbf{p}}\left(\, f_{dyn}\left(\mathbf{u}_{k+1},\mathbf{w}_k\right),\mathbf{x}_k\right), \tag{76}$$

$$W_{OT,k}=\frac{\partial f_{OT}}{\partial \mathbf{w}}\left(\mathbf{x}_k,\mathbf{u}_{k+1},0\right)=\frac{\partial T_{abs\to rel}\left(\mathbf{s},\mathbf{p}\right)}{\partial \mathbf{s}}\left(\, f_{dyn}\left(\mathbf{u}_{k+1},0\right),\mathbf{x}_k\right)\frac{\partial f_{dyn}}{\partial \mathbf{w}}\left(\mathbf{u}_{k+1},0\right). \tag{77}$$

Chapter 4. Active Vision

In this chapter, an explicitly task-oriented approach to the active vision problem is presented. The system tries to reduce the most relevant components of the uncertainty in the world model for the task the robot is currently performing. It is task oriented in the sense that it explicitly considers a task-specific value function. As test-bed for the presented active vision approach, we selected a robot soccer attention problem: goal-covering by a goalie player. The proposed system is compared with information-based approaches. Experimental results show that it surpasses them in the tested application. We conclude that the minimization of the belief entropy is not a useful optimality criterion when the goal is not the reduction of uncertainty itself, and that for such cases, task-oriented optimality criteria are better suited.

Since this uncertainty may have a tremendous negative impact on the performance of the task that the robot is executing, one important issue is how the robot can select actions to reduce it. Active vision basically consists of executing control strategies with the purpose of improving the perception performance by focusing on the most relevant parts of the sensor data. An active vision system may control physical variables, such as gaze direction, camera parameters, etc., and/or the way data is processed such as the considered vision methodology, *region of interest* (ROI), resolution, and parameters, etc. The relationship between perception and action has been studied in several fields apart from robotics, such as neurosciences, psychology and cognitive science. For instance, experiments have shown that the task that a human is executing has a strong influence on his/her gaze selection [1]. In robotics, this suggests that the selection of the most relevant information must be based on the task the robot is performing.

An autonomous mobile robot must estimate the state of a complex and dynamic world. It must pay attention to landmarks in order to self-localize, and to other relevant objects it must interact with, like obstacles to avoid, objects to manipulate, targets to pursuit, etc. The process of estimating the absolute *pose* of the robot from *landmark* observations is known as *self-localization*, while the process of estimating a mobile object position from

observations of the object is known as *object tracking*. Often, a mobile robot must perform self-localization and object tracking simultaneously, which is a problem because these two processes require the observation of different objects. Self-localization requires the observation of landmarks, while object tracking requires the observation of other objects defined by the current task.

In this work, we present a task-oriented approach to the active vision problem, which tries to reduce the most relevant components of the uncertainty in the world model for the execution of the current task the robot is performing. The proposed approach is task oriented because it intends to minimize some statistics that depend on a task-specific *value function* or simply a *task value function*. A task value function is defined as a function of the world state that increases with the degree of convenience of the world state associated with the execution of the current task. The output of the system is a selected sensing control action, which will determine the most convenient part of the data to focus on. The proposed approach is generally applicable for problems where the following assumptions hold: (i) the robot has some degrees of freedom that can directly influence the observations but not the state (e.g. neck motors), (ii) there is a world-modeling stage which uses a Bayesian filter to estimate the world state, and (iii) the task being performed has a task value function defined. The existence of a task value function is held by a wide variety of applications since it is difficult to even make decisions without a measurement of the convenience of each state.

The general approach being described was tested on a specific application taken from robot soccer: goal covering by a goalie player. Robot soccer is a very interesting platform for testing robotic methodologies and algorithms since it presents a challenging environment and complex tasks [2]. This problem makes the selected application interesting for testing the proposed active vision approach.

Section 4.1 reviews the relevant literature regarding sequential decision-making and active vision. While in section 4.2, the problem and its main variables are defined. Section 4.3, describes the main assumptions the proposed system relies on. The sensing action space and different alternatives for its definition are presented in Section 4.4. Then, section 4.5 presents the optimality criteria used for the selection of the sensing action. Approximated versions of the optimality criteria are described in section 4.6. Following this description,

the procedure for updating the belief is explained in section 4.7. Section 4.8 shows the connections of the systems with POMDPs. Finally, a robot soccer case study is examined in section 4.9.

4.1 Literature Review

4.1.1 Sequential Decision-Making

An agent faces the sequential decision-making problem when it interacts with its environment in a discrete-time fashion. As it was explained in section 1.2, in every instant the robot is executing an action, which influences the state evolution. As a result, it obtains an observation and a reward, both of which are influenced by the world state. *Dynamic Programming* [59] is a common modeling scheme for the sequential decision-making problem. Dynamic programming models the state transitions, and the observations in some cases, in a probabilistic fashion. If the observations are assumed to be a deterministic function, then the resulting model is called *Markov Decision Process* (MDP) [92]. On the other hand, if the observations are modeled probabilistically, then we are in presence of a *Partially Observable Markov Decision Process* (POMDP) [93][96]. There are different approaches to solve the sequential decision-making problem.

When there is no model available of the process dynamics, the agent must learn to behave optimally using only the information provided by the obtained rewards and observations. In that case, the agent is facing the *Reinforcement Learning* [39][97] problem. This thesis does not intend to solve the reinforcement learning problem. Thus, a further bibliographical revision is not included.

4.1.1.1 Markov Decision Processes

An MDP is an elegant mathematical model of the stochastic interaction between a decision maker agent and an observable environment. MDPs take into account the predictive uncertainty. With this purpose, the state transition is modeled as a pdf conditioned on the initial state and the executed action. The reward is a deterministic function of the state. The MDPs have their origin in the study of dynamic programming [92]. In [92] the recursive equation for the *value function*, known as the *Bellman equation*, is identified. MDPs can be classified as continuous (CDP) or discrete (DDP), depending on whether the state space is

continuous or discrete [98]. For DDP's, there is an exact solution, while for CDP's there are only computable approximations in the general case. The approximate solutions for CDP can be divided in two groups: discrete and parametric (also called smooth). The discrete solutions make a discretization of the state space and then solve the resulting DDP. The parametric solutions approximate the value function to a parametric function with a finite number of parameters. In [99], different alternatives for solving CPD's are compared. MDPs do not take into account the observational uncertainty and they can have a poor performance when that uncertainty is relevant.

4.1.1.2 Partially-Observable Markov Decision Processes

A POMDP is a model based on an MDP that also considers the observational uncertainty. Therefore, at every instant, instead of knowing the state, the agent knows the state belief, given the past observations. Some of the first approaches to the POMDP idea can be found in [95] and [96]. The *value iteration* algorithm for POMDPs was introduced in [94] and [100]. It solves the POMDP problem for a finite number of states but it consumes a huge amount of computational resources, even for a very small number of states (less than 100). For this reason, several approximate techniques for the solution of POMDPs have been developed. Some of them are reviewed in [101].

One technique for approximately solving a POMDP assumes that the uncertainty in the state exists at the current instant, but will disappear after executing the next action. This approximation, known as QMDP [102][103], simply solves the inherent MDP using value iteration, and then, in every instant, the value of any resulting belief is approximated by the expected value of the value function given the current belief. The main drawback of QMDP is that the agent will never execute actions with the purpose of gaining information.

Another known technique for solving POMDPs is that of *Augmented MDP* (AMDP) [104], which represents the belief using some subset of its statistics assumed to be statistically sufficient: the most probable state and the belief entropy. AMDP solves an MDP with a so-called augmented state consisting of the already mentioned statistics of the belief. By adding the belief entropy to the state, an agent using AMDP is able to make decisions that will aim to gain information. There is another approximate technique for solving POMDPs

known as *MonteCarlo POMDP* (MC-POMDP) [105], which represents the belief using a state sample set. MC-POMDP is able to work with continuous state spaces.

4.1.2 Active Vision

The basic ideas of the active vision paradigm were introduced with different names, such as: *active vision* [3], *active perception* [4], and *animate vision* [5]. The contribution of this paradigm is that it does not consider perception to be a passive process, but as one that could conveniently influence the control actions. Since the creation of the active vision paradigm, a wide variety of active vision approaches have been proposed. The applications of these approaches include the following: object recognition [6][8], self-localization [9][11], robot navigation [9], [12], SLAM [13], among others. The controlled variables in active vision approaches include, for example, robot movements [9], sensor direction [6]-[10], [14][29], camera parameters [6], [28], and the order in which the pixels are analyzed in the vision module [30].

Currently existing active vision approaches may be classified from different points of view. Firstly, they can be classified as probabilistic or non-probabilistic approaches. Probabilistic approaches estimate the *probability density function* (pdf) of the state, often called *belief*, based on the past observations and actions, and try to reduce the belief's uncertainty. Most probabilistic approaches reduce the belief uncertainty regardless of the task being performed. As a result, they are well suited for applications where the task is the reduction of the uncertainty itself, but they are not generally optimal from the task performance point of view. These approaches are often based on information theory (e.g. [6], [7], [10], [13], [28]). On the other hand, some active vision systems are focused on the performance of the task execution. They have been called *behavioral* [31]. Typically, they learn a mapping between the last observation and the next sensing control action, which intends to maximize some measurement of the task execution performance [14]. By doing so, they neglect the information contained in the past observations and actions.

There are works that combine both probabilistic and behavioral features. For example, in [9], [12], both a belief is estimated and a task-specific cost function is taken into account. In both works, the actions have influence on the task being executed and an observational cost function is defined. Finally, the cost function is balanced with the information gain using a heuristic combination. In [29] the entropy of the self-localization estimate is linearly

combined with the entropy of the estimation of a task-relevant object (a soccer ball), which does not influence the localization.

4.2 Basic Definitions

Without loss of generality, we assume that for each time step *k*, the events occur in the following chronological order (see figure 4):

- Based on b_{k-1}, the decision-making module decides the best action $\mathbf{u}_k^{act}$, which should maximize the expectation of the task value function: $E\left(V_k\left(\mathbf{x}_k\right)\middle|\mathbf{u}_k^{act}\right)$. $\mathbf{u}_k^{act}$ is executed and according to the process dynamics it moves the state from $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$.
- The world-modeling module makes a prediction about $\mathbf{x}_k$, $b_k^-\left(\mathbf{x}_k\right) = p\left(\mathbf{x}_k \middle| U_{k-1}, Z_{k-1}, \mathbf{u}_k^{act}\right)$.
- From the knowledge of b_k^-, the robot decides the optimal sensing action $\mathbf{u}_k^{sen}$. This decision should be made considering some optimality criterion regarding the resulting belief b_k.
- The robot executes the sensing action $\mathbf{u}_k^{sen}$ and gets the observation $\mathbf{z}_k$, which is influenced by $\mathbf{u}_k^{sen}$.
- The world-modeling module uses the observation $\mathbf{z}_k$ to update the belief and thus obtain b_k.

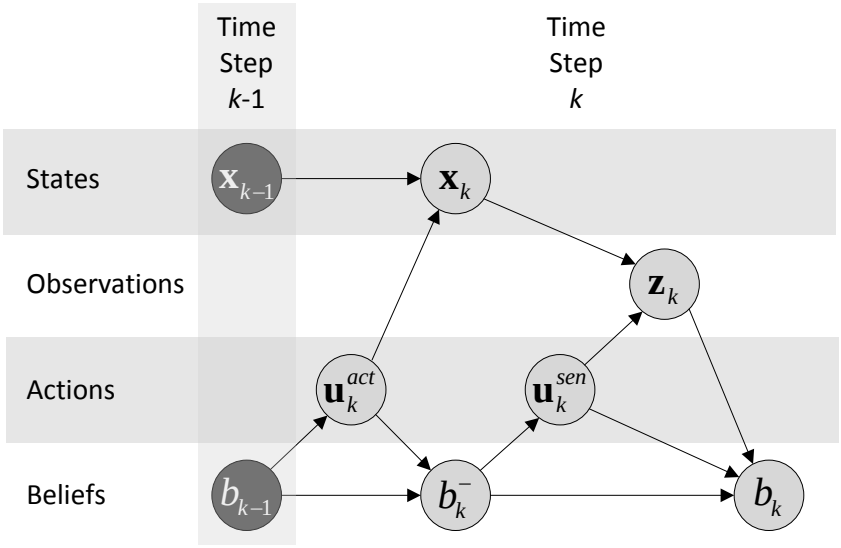


Figure 4. Chronological order of events and information flow for instant *k*, given the state and the belief for instant *k*-1.

In practice, these events are generally asynchronous. For example, for a legged robot with a neck, $\mathbf{u}_k^{act}$ is selected before every robot step, while $\mathbf{u}_k^{sen}$ could be modified for every

image. The former sequence of events is defined using the frequency of selection of $\mathbf{u}_k^{sen}$. If $\mathbf{u}_k^{sen}$ must be selected more often than $\mathbf{u}_k^{act}$, then, we must decompose $\mathbf{u}_k^{act}$ into different parts and consider that each part of $\mathbf{u}_k^{act}$ corresponds to one $\mathbf{u}_k^{sen}$. In addition, in order to work in real time, it is often convenient that the world-modeling stages of prediction and estimation are asynchronous to this decision-making process.

Since $\mathbf{u}_k^{sen}$ cannot directly influence the future states, its main goal is to reduce the uncertainty of the state. If the robot has a set U^{sen} of $N_{\mathbf{u}^{sen}}$ sensing action possibilities, we define the active vision problem as the selection of a sensing action $\mathbf{u}_k^{sen*} \in U^{sen}$, which is optimal according to some criterion. In a probabilistic framework, the optimality criteria should be a function of the predicted belief b_k^-. Note that in general b_k depends on $\mathbf{u}_k^{sen}$ because some parameters (for instance, the variance of $\mathbf{u}_k^{sen}$) used in the correction stage of the estimation of b_k could depend on $\mathbf{u}_k^{sen}$.

4.3 Main Assumptions

4.3.1 Sensing-actions existence

The system needs the existence of a subset of the robot's degrees of freedom that influences the observations but not the state. This assumption holds in many mobile robots, where the action at time *k*, $\mathbf{u}_k$, can be factorized into two parts: one, $\mathbf{u}_k^{act}$, that influences directly $\mathbf{x}_k$ and indirectly $\mathbf{z}_k$ through $\mathbf{x}_k$, and other, $\mathbf{u}_k^{sen}$, that influences directly $\mathbf{z}_k$ but not $\mathbf{x}_k$ (See figure 4). For example, in a legged robot with a camera mounted on an articulated neck, whose movements do not influence the world state, $\mathbf{u}_k^{sen}$ corresponds to the selection of vision parameters and the movements of the neck and $\mathbf{u}_k^{act}$ corresponds to the movements of the rest of the robot. The former condition is equivalent to:

$$\begin{cases} p\left(\mathbf{x}_{k+1} \middle| \mathbf{x}_k, \mathbf{u}_{k+1}\right) = p\left(\mathbf{x}_{k+1} \middle| \mathbf{x}_k, \mathbf{u}_{k+1}^{act}\right) \\ p\left(\mathbf{z}_k \middle| \mathbf{x}_k, \mathbf{u}_k\right) = p\left(\mathbf{z}_k \middle| \mathbf{x}_k, \mathbf{u}_k^{sen}\right) \end{cases}. \tag{78}$$

In the Gaussian case, the former condition is equivalent to:

$$\begin{cases} f\left(\mathbf{x}_k, \mathbf{u}_{k+1}, \mathbf{w}_k\right) = f\left(\mathbf{x}_k, \mathbf{u}_{k+1}^{act}, \mathbf{w}_k\right) \\ h\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k\right) = h\left(\mathbf{x}_k, \mathbf{u}_k^{sen}, \mathbf{v}_k\right) \end{cases}. \tag{79}$$

When this assumption holds, it makes sense to select $\mathbf{u}^{sen}$ in order to make more useful observations, and that $\mathbf{u}^{sen}$ is selected in terms of the last selection of $\mathbf{u}^{act}$. For this purpose, the selection of $\mathbf{u}_k^{act}$ may be performed before that of $\mathbf{u}_k^{sen}$, and then the selected $\mathbf{u}_k^{act}$ may be known in the $\mathbf{u}_k^{sen}$ selection process.

4.3.2 Task Value Function Existence

The proposed approach assumes that the robot is executing a task and it has defined a task value function for it. A task value function is defined as the expected accumulated *reward* that the robot will obtain given state $\mathbf{x}_k$ [39]:

$$V_k\left(\mathbf{x}_k\right) = E\left(\sum_k \gamma^k R_k \middle| \mathbf{x}_k\right). \tag{80}$$

With $\gamma \in (0,1]$ a *discount factor* that makes the rewards weight decay exponentially with time.

For several decision-making schemes in the literature, like *Markov Decision Processes* (MDP) and some kinds of *reinforcement learning*, the robot is able to estimate the task value function. Even when there is not any task value function theoretically defined, it can be learned using any appropriate reinforcement learning method such as Q-Learning [39].

4.3.3 Functional Modules

Apart from the active vision module, the proposed approach assumes that the robot control system has at least four additional functional modules: (i) *vision*, where the camera images are processed in order to get the observation $\mathbf{z}_k$, (ii) *world modeling*, where $\mathbf{u}_k^{act}$ and $\mathbf{z}_k$ are used to update the belief, (iii) *decision making*, where the belief is used to select the action $\mathbf{u}_k^{act}$ that the robot will execute, and (iv) *actuation*, where $\mathbf{u}_k^{act}$ is executed. See a block diagram of the complete system in figure 5.

The following paragraphs contain a short description of these functional modules and the particular implementations we have chosen for performing the experiments. The reader

should note that the presented active vision method is not restricted to these particular implementations of the modules.

4.3.3.1 Vision

The vision module processes camera images in order to detect interesting objects. The output of the vision module is a vector containing the relative positions of the detected objects. This vector is the observation, $\mathbf{z}_k$. Landmarks and other objects of interest are perceived using a color-based vision method. The distances and angles of the perceived objects in relation to the robot are estimated using a segmented image that is built using a look up table and the a priori knowledge of the field objects colors. Context-dependent color segmentation has been included in our library. Our color-based vision system is described in detail in [17], [40].

4.3.3.2 World Modeling

The world-modeling module processes the observation and odometry measures in order to filter the observation and odometric noise. Bayesian methods are currently the most used for this purpose. Among them, Kalman filters and particle filters are the most popular ones. Our particular implementation [17] of this module uses Extended Kalman Filters (EKF) for estimating both the robot self-localization and the relative position of the objects.

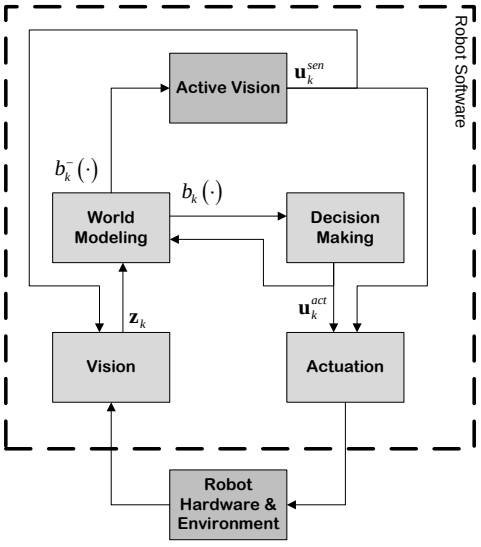


Figure 5. Block diagram of a robotic controller in which the proposed active vision system is immersed.

4.3.3.3 Decision Making

The decision-making module processes observations and the state belief in order to generate commands that the actuation module should execute. The decision-making system must fit with the task being executed. In applications such as robot soccer, where complex tasks must be solved, the behavior-based approach is often selected. Our particular implementation [41], [23] is based on behaviors of different levels of abstraction. Higher-level behaviors, such as "play soccer", select and parameterize lower level behaviors such as "go to ball", and so on. Finally, the lowest level behaviors, such as "walk", generate action commands that are carried out by the actuation module.

4.3.3.4 Actuation

This module is in charge of controlling the robot motors according to instructions provided by the decision-making module and it is also in charge of generating odometric estimations of the robot displacements. The module is able to move independently different subsets of the robot motor set (for example the head may be independent of the legs) and eventually coordinate different subsets for performing special movements. Of course, this module is very dependent on the robot platform. Our actuation [24] for humanoid robots (tested on Aldebaran Robotics NAO and Hajime HR-18) uses omnidirectional gaits provided by the robot manufacturers.

4.4 Sensing Action Space

The sensing action space U^{sen} can be selected from several options, from which we mention two:

4.4.1 Sensing Control Actions

The action space may be defined as the direct commands that are set to the actuators or the vision module. For example, in a gaze selection system, the action *space* could include tilt, pan and/or zoom of the camera. In a ROI selection system, the actions could be defined as the parameters of the ROI. If the resulting action space is continuous, it can be discretized in order to simplify the resolution of the active vision problem.

4.4.2 Object to Focus On

If there is a limited set of objects that the vision module can focus on, then the action space may be defined as the set of observable objects. This alternative is feasible only if the state contains enough information to select the control actions that will result in the system focusing on some object. In this case, the system relies also on an Object Focusing Behavior that must select an adequate control action.

4.5 Optimality Criteria

In order to select a sensing action $\mathbf{u}_k^{sen*}$ that is considered "optimal", it is necessary to define an optimality criterion. The selection of this criterion will strongly determine the behavior of the system, and here is where we believe that the task can have an explicit influence on the system. If we consider a discrete sensing action space, U^{sen}, then $\mathbf{u}_k^{sen*}$ can be found by iterating over U^{sen} and for each $\mathbf{u}_k^{sen}$ calculating the value of the selected optimality criterion.

With the aim of making explicit the influence of $\mathbf{z}_k$ in $b_k(\mathbf{x}_k)$, we will write the latter as $b_k(\mathbf{x}_k|\mathbf{z}_k)$. Additionally, we can define the *policy* $\pi(b_k)$ as a function that, given some belief b_k, returns the following action $\mathbf{u}_{k+1}^{act}$ that is optimal for a given criterion. Note that the functions $b_k(\cdot|\mathbf{z}_k)$ and $\pi(b_k)$ correspond respectively to the correction stage of the world-modeling module, and the decision-making process in the decision-making module. Thus, the active vision module has to simulate the operation of those modules in order to evaluate some of the optimality criteria (see examples in subsections 3.8 and 4.4).

The probability $p(\mathbf{z}_k|U_k, Z_{k-1})$ of observing $\mathbf{z}_k$ given U_k (which of course includes $\mathbf{u}_k^{sen}$) and Z_{k-1} may be calculated as:

$$p(\mathbf{z}_k|U_k, Z_{k-1}) = \int p(\mathbf{z}_k|U_k, Z_{k-1}, \mathbf{x}_k)\, p(\mathbf{x}_k|U_k, Z_{k-1})\, d\mathbf{x}_k \,. \tag{81}$$

Since $\mathbf{u}_k^{sen}$ does not influence $\mathbf{x}_k$, $p(\mathbf{x}_k|U_k, Z_{k-1}) = b_k^-(\mathbf{x}_k)$. Also, given $\mathbf{x}_k$, neither U_{k-1}, Z_{k-1}, nor $\mathbf{u}_k^{act}$ influence $\mathbf{z}_k$. Then,

$$p\left(\mathbf{z}_k \left| U_k, Z_{k-1}\right.\right) = \int p\left(\mathbf{z}_k \left| \mathbf{x}_k, \mathbf{u}_k^{sen}\right.\right) b_k^-\left(\mathbf{x}_k\right) d\mathbf{x}_k \,. \tag{82}$$

4.5.1 Information Theory Criteria

In the literature, two information-based optimality criteria for the active vision problem have been proposed. These criteria are the *minimum conditional entropy* [6] and the *maximum mutual information* (MI) [13][28].

The *continuous conditional entropy*[7], $En\left(\mathbf{x}_k \left| \mathbf{z}_k, \mathbf{u}_k^{sen}\right.\right)$, is defined as [6]:

$$En\left(\mathbf{x}_k \left| \mathbf{z}_k, \mathbf{u}_k^{sen}\right.\right) = -\int p\left(\mathbf{z}_k \left| U_k, Z_{k-1}\right.\right) \int b_k\left(\mathbf{x}_k \left| \mathbf{z}_k\right.\right) \log\left(b_k\left(\mathbf{x}_k \left| \mathbf{z}_k\right.\right)\right) d\mathbf{x}_k d\mathbf{z}_k \,. \tag{83}$$

The mutual information $I\left(\mathbf{x}_k; \mathbf{z}_k \left| \mathbf{u}_k^{sen}\right.\right)$ corresponds to:

$$I\left(\mathbf{x}_k; \mathbf{z}_k \left| \mathbf{u}_k^{sen}\right.\right) = En^-\left(\mathbf{x}_k\right) - En\left(\mathbf{x}_k \left| \mathbf{z}_k, \mathbf{u}_k^{sen}\right.\right), \tag{84}$$

where $En^-\left(\mathbf{x}_k\right)$ is the *differential entropy* of $\mathbf{x}_k$ before the observation $\mathbf{z}_k$:

$$En^-\left(\mathbf{x}_k\right) = -\int b_k^-\left(\mathbf{x}_k\right) \log\left(b_k^-\left(\mathbf{x}_k\right)\right) d\mathbf{x}_k \tag{85}$$

These criteria have a wide applicability since they do not depend on the task, but they intend to reduce the belief entropy. This generality comes at the cost of neglecting the important components of the state uncertainty.

In the following subsections, we will present two *task-oriented optimality criteria* (TOOC). These optimality criteria have the aim of reducing the uncertainty in the most relevant components from the point of view of the task being performed. This focused reduction of uncertainty is accomplished by means of considering the task value function as a way of weighting the uncertainty costs. The use of the task value function, by definition (see Eq. (4)), considers the long-term effects of the sensing action. The first TOOC was introduced in [15], while the second one was introduced in [16].

---

[7] Note that *H* is the standard notation for conditional entropy, but we have replaced it to avoid confusions with the observation Jacobian.

4.5.2 Minimum Expected Task Value Variance

This criterion is based on the intuition that the most important components of the uncertainty are those that make the task value function vary the most. Therefore, it intends to minimize the expected variance $V_k^{\text{var}}\left(\mathbf{u}_k^{sen}\right)$ of the task value function after the next observation:

$$V_k^{\text{var}}\left(\mathbf{u}_k^{sen}\right) = \int p\left(\mathbf{z}_k \middle| U_k, Z_{k-1}\right) \text{var}\left(V_k\left(\mathbf{x}_k\right)\middle|\mathbf{z}_k\right) d\mathbf{z}_k \,, \tag{86}$$

with

$$\text{var}\left(V_k\left(\mathbf{x}_k\right)\middle|\mathbf{z}_k\right) = E\left(V_k^2\left(\mathbf{x}_k\right)\middle|\mathbf{z}_k\right) - E\left(V_k\left(\mathbf{x}_k\right)\middle|\mathbf{z}_k\right)^2, \tag{87}$$

and

$$E\left(V_k\left(\mathbf{x}_k\right)\middle|\mathbf{z}_k\right) = \int b_k\left(\mathbf{x}_k\middle|\mathbf{z}_k\right) V_k\left(\mathbf{x}_k\right) d\mathbf{x}_k \,;\; E\left(V_k^2\left(\mathbf{x}_k\right)\middle|\mathbf{z}_k\right) = \int b_k\left(\mathbf{x}_k\middle|\mathbf{z}_k\right) V_k^2\left(\mathbf{x}_k\right) d\mathbf{x}_k \,. \tag{88}$$

Note that another possibility would be using the maximization of the expected mean, $V_k^E\left(\mathbf{u}_k^{sen}\right)$, of the task value function as an optimality criterion:

$$V_k^E\left(\mathbf{u}_k^{sen}\right) = \int p\left(\mathbf{z}_k \middle| U_k, Z_{k-1}\right) E\left(V_k\left(\mathbf{x}_k\right)\middle|\mathbf{z}_k\right) d\mathbf{z}_k \,. \tag{89}$$

This optimality criterion may be tempting at a first glance. However, it aims to be as optimistic as possible about the future task values but not to reduce the uncertainty.

4.5.3 Maximum Expected Action Task Value

This criterion intends to help the selection of the next action that influences the state, $\mathbf{u}_{k+1}^{act}$, by minimizing the negative effects that the uncertainty in b_k may have on the result of that decision. In other words, this criterion intends to maximize the expected value $V_{k+1}^E\left(\mathbf{u}_k^{sen}\right)$ of the task value function $V_{k+1}\left(\mathbf{x}_{k+1}\right)$ after the next action $\mathbf{u}_{k+1}^{act}$:

$$V_{k+1}^E\left(\mathbf{u}_k^{sen}\right) = \int b_k^-\left(\mathbf{x}_k\right) \int p\left(\mathbf{z}_k\middle|\mathbf{x}_k, \mathbf{u}_k^{sen}\right) \int V_{k+1}\left(\mathbf{x}_{k+1}\right) p\left(\mathbf{x}_{k+1}\middle|\mathbf{x}_k, \pi\left(b_k\left(\cdot\middle|\mathbf{z}_k\right)\right)\right) d\mathbf{x}_{k+1} d\mathbf{z}_k d\mathbf{x}_k \,. \tag{90}$$

See Appendix 1 for the proof.

4.6 Approximated Optimality Criteria

If the state and observation spaces are discrete, the integrals become sums and then they are computable. If some of these spaces are continuous, the integrals are not computable and then some approximation techniques must be used.

In the following subsections, approximated versions of the task-oriented optimality criteria are presented. These approximations are based on the deterministic sampling schemes called sigma points, which are described in subsection 1.6.2.2.2. An approximated version of the MI algorithm using the Monte Carlo sampling scheme is presented in [28].

4.6.1 Approximated Minimum Expected Task Value Variance

For approximating $V_k^{Var}\left(\mathbf{u}_k^{sen}\right)$ the following algorithm may be employed:

1. Sample $\left\{\left(\boldsymbol{\chi}_i^{\mathbf{x}_k^-}, \omega_i^{\mathbf{x}_k^-}\right)\right\}$ from $b_k^-\left(\mathbf{x}_k\right)$.
2. *For each* $\left(\boldsymbol{\chi}_i^{\mathbf{x}_k^-}, \omega_i^{\mathbf{x}_k^-}\right)$
   a. Sample $\left\{\left(\boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}, \omega_{j,i}^{\mathbf{z}_k}\right)\right\}$ from $p\left(\mathbf{z}_k \middle| \boldsymbol{\chi}_i^{\mathbf{x}_k^-}, \mathbf{u}_k^{sen}\right)$.
   b. *For each* $\left(\boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}, \omega_{j,i}^{\mathbf{z}_k}\right)$
      i. Calculate $b_k\left(\cdot \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right)$
      ii. Sample $\left\{\left(\boldsymbol{\chi}_{m,j,i}^{\mathbf{x}_k}, \omega_{m,j,i}^{\mathbf{x}_k}\right)\right\}$ from $b_k\left(\cdot \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right)$
      iii. $E\left(V_k\left(\mathbf{x}_k\right) \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right) \approx \sum_m \omega_{m,j,i}^{\mathbf{x}_k} V_k\left(\boldsymbol{\chi}_{m,j,i}^{\mathbf{x}_k}\right)$
      iv. $E\left(V_k^2\left(\mathbf{x}_k\right) \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right) \approx \sum_m \omega_{m,j,i}^{\mathbf{x}_k} V_k^2\left(\boldsymbol{\chi}_{m,j,i}^{\mathbf{x}_k}\right)$
      v. $\mathrm{var}\left(V_k\left(\mathbf{x}_k\right) \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right) = E\left(V_k^2\left(\mathbf{x}_k\right) \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right) - E\left(V_k\left(\mathbf{x}_k\right) \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right)^2$
3. $V_k^{\mathrm{var}}\left(\mathbf{u}_k^{sen}\right) \approx \sum_{i,j} \omega_i^{\mathbf{x}_k^-} \omega_{j,i}^{\mathbf{z}_k} \mathrm{var}\left(V_k\left(\mathbf{x}_k\right) \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right)$

4.6.2 Approximated Maximum Expected Action Task Value

For approximating $V_{k+1}^E\left(\mathbf{u}_k^{sen}\right)$ the following algorithm may be employed:

1. Sample $\left\{\left(\boldsymbol{\chi}_i^{\mathbf{x}_k^-}, \omega_i^{\mathbf{x}_k^-}\right)\right\}$ from $b_k^-(\mathbf{x}_k)$.
2. *For each* $\left(\boldsymbol{\chi}_i^{\mathbf{x}_k^-}, \omega_i^{\mathbf{x}_k^-}\right)$
    a. Sample $\left\{\left(\boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}, \omega_{j,i}^{\mathbf{z}_k}\right)\right\}$ from $p\left(\mathbf{z}_k \middle| \boldsymbol{\chi}_i^{\mathbf{x}_k^-}, \mathbf{u}_k^{sen}\right)$.
    b. *For each* $\left(\boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}, \omega_{j,i}^{\mathbf{z}_k}\right)$
        i. Calculate $b_k\left(\cdot \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right)$
        ii. $\mathbf{u}_{k+1,j,i}^{act} = \pi\left(b_k\left(\cdot \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right)\right)$
        iii. Sample $\left\{\left(\boldsymbol{\chi}_{m,j,i}^{\mathbf{x}_{k+1}}, \omega_{m,j,i}^{\mathbf{x}_{k+1}}\right)\right\}$ from $p\left(\mathbf{x}_{k+1} \middle| \boldsymbol{\chi}_i^{\mathbf{x}_k^-}, \mathbf{u}_{k+1,j,i}^{act}\right)$.
3. $V_{k+1}^E\left(\mathbf{u}_k^{sen}\right) \approx \sum_{i,j,m} \omega_i^{\mathbf{x}_k^-} \omega_{j,i}^{\mathbf{z}_k} \omega_{m,j,i}^{\mathbf{x}_{k+1}} V_{k+1}\left(\boldsymbol{\chi}_{m,j,i}^{\mathbf{x}_{k+1}}\right)$.

4.6.3 Computational Cost Considerations

Note that in both approximated TOOC, the algorithm steps up to 2.b.i. are identical. Since the optimality criterion must be calculated for all of the sensing actions in U^{sen}, the steps in 2.b of both approximated TOOC's must be executed $N_{\mathbf{u}^{sen}} N_{\mathbf{x}_k^-} N_{\mathbf{z}_k}$ times, which may make these procedures too computationally expensive. One alternative that would make them computationally cheaper is to make $N_{\mathbf{x}_k^-}$ and/or $N_{\mathbf{z}_k}$ equal to 1 by using the mean sampling scheme. For example, in [15], both $N_{\mathbf{x}_k^-}$ and $N_{\mathbf{z}_k}$ are set equal to one, and $\mathbf{u}_k^{sen}$ only influences in the covariance of $b_k\left(\cdot \middle| \boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}\right)$ but not its mean.

4.7 Belief Update

The belief update corresponds to the calculation of $b_k\left(\cdot \middle| \mathbf{z}_k\right)$ from b_k^- and $\mathbf{z}_k$. As previously stated, this step corresponds to the simulation of the correction stage in the world-modeling module. This step can be performed using different Bayesian methods such as Kalman Filters or Particle Filters. In the Gaussian case, if b_k / b_k^- are Gaussian pdf's with means $\mu_k^{\mathbf{x}}$ /

$\mu_k^{\mathbf{x}^-}$ and covariance matrices $\Sigma_k^{\mathbf{x}}/\Sigma_k^{\mathbf{x}^-}$. Then, using Extended Kalman filter (EKF), the update step becomes the correction stage of the EKF, shown in equations (18) and (19).

4.8 Connection to POMDPs

Partially Observable MDPs (POMDPs) [44] are the subject of much research. Differing from regular MDPs, in POMDPs the state is assumed to be only partially observable and it must be estimated through noisy observations. In the general POMDP solution, instead of a state-value function, $V_k(\mathbf{x}_k)$, a belief-value function $V_k(b_k(\mathbf{x}_k))$, whose parameter is the state belief, $b_k(\mathbf{x}_k)$, is estimated. If $V_k(b_k(\mathbf{x}_k))$ is known in advance, the active-vision problem in the terms we have presented it consists of selecting the $\mathbf{u}_k^{sen}$ that maximizes $V_k\left(b_k\left(\mathbf{x}_k|\mathbf{u}_k^{sen}\right)\right)$. However, the general solution for the estimation of $V_k(b_k(\mathbf{x}_k))$ is computationally intractable [45]. Several approximations that lead to suboptimal solutions of the POMDP problem have been presented in the literature.

The proposed task-oriented approach is closely related to an approximated solution to POMDPs known as QMDP [46]. QMDP approximates the POMDP by assuming that the uncertainty will disappear after the next action. So, the next action is selected considering only the uncertainties in the current state and in the result of the next action. If the QMDP assumption holds for $\mathbf{u}_{k+1}^{act}$ (i.e. the state uncertainty will disappear after executing $\mathbf{u}_{k+1}^{act}$), then the presented task-oriented approach for active vision appears as the natural solution for selecting $\mathbf{u}_k^{sen}$.

There are several other approximations for solving the POMDP problem. For example, approximated solutions with a reasonable performance and time consumption on problems with hundreds of states have been developed recently [47]. However, they are based on estimating $V_k(b_k(\mathbf{x}_k))$ only in a subset of the whole $b_k(\mathbf{x}_k)$ space. So, there is no guarantee that by using these methods the problem under study can be solved in practice. The applicability of those methods must be further studied.

4.9 Case Study: Goal-Covering by a Goalie Player

With the purpose of demonstrating the applicability of the proposed approach, we analyze a robot soccer related problem: *goal-covering* by a goalie player. Basically, this task consists of maintaining the robot's position between the ball and its own goal. For an illustration of a robot performing this task, see figure 6.

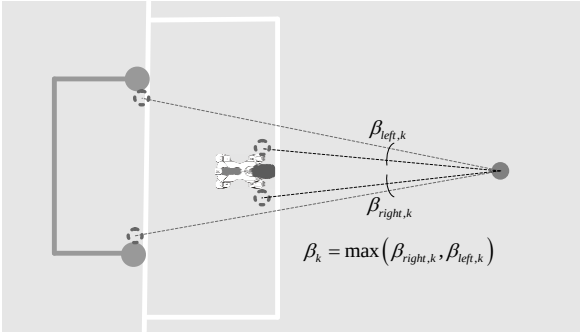


Figure 6. Geometry of the goal-covering task.

4.9.1 State Space

For the goal-covering task, the state of the world may be defined as $\mathbf{x}_k = \left(\mathbf{G}_k, \tilde{\mathbf{B}}_k\right)^T$, where $\mathbf{G}_k = \left(G_k^x, G_k^y, G_k^\theta\right)^T$ is the absolute[8] pose of the goalie and $\tilde{\mathbf{B}}_k = \left(\tilde{B}_k^x, \tilde{B}_k^y\right)^T$ is the relative[9] position of the ball. Note that the ball is mobile and its trajectory is unpredictable for the robot. This definition of $\mathbf{x}_k$ which uses $\tilde{\mathbf{B}}_k$ instead of the absolute position of the ball, $\mathbf{B}_k = \left(B_k^x, B_k^y\right)^T$, is convenient from the world-modeling point of view since $\mathbf{G}_k$ may be estimated using observations of the existent landmarks and $\tilde{\mathbf{B}}_k$ may be estimated using observations of the ball. From these definitions, the self-localization and ball-tracking processes are independent, which is not a requisite but is convenient for the simplicity of the problem formulation.

[8] By absolute we mean with respect to a coordinate system which is fixed to the field

[9] By relative we mean with respect to a coordinate system which is fixed to the robot

We consider the beliefs to have a Gaussian form, with $\mathbf{x}_k \sim N\left(\mu_x^k, \Sigma_x^k,\right)$, and we update the beliefs following the EKF equations as is shown in subsection 3.8.

4.9.2 Observation and Sensing Action Spaces

We selected the "*object to focus on*" sensing action space (See section 3.4.2). Then, $\mathbf{u}_k^{sen}$ corresponds to an intended object o_k to observe. For the sake of simplicity, we will assume that the observation $\mathbf{z}_k$ will always contain the object o_k and no other[10]. The zero-error observation of o_k may be defined as its relative position in polar coordinates $\tilde{\mathbf{O}}_k^{pol}$, $h\left(\mathbf{x}_k, \mathbf{u}_k^{sen}, 0\right) = h\left(\mathbf{x}_k, o_k\right) = \tilde{\mathbf{O}}_k^{pol} = \left(\tilde{O}_k^r, \tilde{O}_k^\theta\right)$. We also consider that the observational model is affected by additive Gaussian noise. Then, the observation model for o_k is:

$$h\left(\mathbf{x}_k, o_k, \mathbf{v}_k\right) = \left(\sqrt{\left(\tilde{O}_k^x\right)^2 + \left(\tilde{O}_k^y\right)^2}, \arctan\left(\tilde{O}_k^y / \tilde{O}_k^x\right)\right)^T + \mathbf{v}_k, \tag{91}$$

with

$$\tilde{\mathbf{O}}_k = \left(\tilde{O}_k^x, \tilde{O}_k^y\right) = \begin{cases} Rot\left(-G_{\theta,k}\right)\left(\mathbf{O}_k^i - \left(G_{x,k}, G_{y,k}\right)\right) & \text{if } o_k \text{ is a landmark} \\ \tilde{\mathbf{B}}_k & \text{if } o_k \text{ is the ball} \end{cases}, \tag{92}$$

where $\mathbf{O}_k^i$ is the fixed and known absolute position of o_k. Now, given a sample of the state, $\boldsymbol{\chi}_i^{\mathbf{x}_k^-}$, and a sensing action $\mathbf{u}_k^{sen}$, we can directly sample $\left\{\left(\boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}, \omega_{j,i}^{\mathbf{z}_k}\right)\right\}$ from $\mathbf{z}_k \sim N\left(h\left(\boldsymbol{\chi}_i^{\mathbf{x}_k^-}, o_k\right), \mathbf{R}_k\right)$.

We will make explicit the dependence between $\mathbf{H}_k$ and o_k:

[10] This assumption can easily be discarded if necessary, with slight modifications to the observation sampling procedure.

$$\mathbf{H}(o_k) = \frac{\partial h}{\partial \mathbf{x}_k}(\mathbf{x}_k, o_k) = \begin{cases} \begin{bmatrix} \left[\frac{\partial h_{o_k}}{\partial \mathbf{G}_k}(\mathbf{G}_k, o_k)\right] & 0 & 0 \\ & 0 & 0 \end{bmatrix} & \text{if } o_k \text{ is a landmark} \\ \begin{bmatrix} 0 & 0 & 0 & \left[\frac{\partial h_{o_k}}{\partial \tilde{\mathbf{B}}_k}(\tilde{\mathbf{B}}_k, o_k)\right] \\ 0 & 0 & 0 & \end{bmatrix} & \text{if } o_k \text{ is the ball} \end{cases} . \quad (93)$$

The reader should note that different selections of o_k will possibly result in different observation Jacobians $\mathbf{H}(o_k)$ and observation noise covariance matrices $\mathbf{R}_k$, and consequently, in different beliefs b_k.

4.9.3 Action Space

Our definition of $\mathbf{u}_{k+1}^{act}$ corresponds to the intended robot relative displacement. The robot relative displacement $\Delta\mathbf{G}_{k+1} = \left(\Delta G_{k+1}^x, \Delta G_{k+1}^y, \Delta G_{k+1}^\theta\right)^T$ is defined as the resulting pose of the robot $\mathbf{G}_{k+1}$ with respect to its previous pose $\mathbf{G}_k$. We also assert that the process model is affected by additive Gaussian noise. Put in equations, we define $\Delta\mathbf{G}_{k+1} = \mathbf{u}_{k+1}^{act} + \mathbf{w}_k$, and then $f\left(\mathbf{x}_k, \mathbf{u}_{k+1}^{act}, \mathbf{w}_k\right) = f\left(\mathbf{x}_k, \Delta\mathbf{G}_{k+1}\right)$, and,

$$f\left(\mathbf{x}_k, \Delta\mathbf{G}_{k+1}\right) = \left(f_G\left(\mathbf{G}_k, \Delta\mathbf{G}_{k+1}\right), f_B\left(\tilde{\mathbf{B}}_k, \Delta\mathbf{G}_{k+1}\right)\right)^T , \quad (94)$$

with,

$$f_G\left(\mathbf{G}_k, \Delta\mathbf{G}_{k+1}\right) = \left(\mathbf{G}_k^{pos} + Rot\left(G_k^\theta\right)\Delta\mathbf{G}_{k+1}^{pos}, G_k^\theta + \Delta G_{k+1}^\theta\right)^T , \quad (95)$$

and,

$$f_B\left(\tilde{\mathbf{B}}_k, \Delta\mathbf{G}_{k+1}\right) = Rot\left(-\Delta G_{k+1}^\theta\right)\left(\tilde{\mathbf{B}}_k - \Delta\mathbf{G}_{k+1}^{pos}\right), \quad (96)$$

where $\mathbf{G}_k^{pos} = \left(G_k^x, G_k^y\right)^T$ is the position of the robot, and $\Delta\mathbf{G}_k^{pos} = \left(\Delta G_k^x, \Delta G_k^y\right)^T$.

From these definitions, given samples of the state, $\boldsymbol{\chi}_i^{\mathbf{x}_k^-}$, and action, $\mathbf{u}_{k+1,j,i}^{act}$, we can sample $\left\{\left(\boldsymbol{\chi}_{m,j,i}^{\mathbf{x}_{k+1}}, \omega_{m,j,i}^{\mathbf{x}_{k+1}}\right)\right\}$ by first obtaining samples $\left\{\left(\boldsymbol{\chi}_{m,j,i}^{\Delta\mathbf{G}_{k+1}}, \omega_{m,j,i}^{\Delta\mathbf{G}_{k+1}}\right)\right\}$ from $\Delta\mathbf{G}_{k+1,j,i} \sim N\left(\mathbf{u}_{k+1,j,i}^{act}, \mathbf{Q}_k\right)$ and then directly evaluating $\boldsymbol{\chi}_{m,j,i}^{\mathbf{x}_{k+1}} = f\left(\boldsymbol{\chi}_i^{\mathbf{x}_k^-}, \boldsymbol{\chi}_{m,j,i}^{\Delta\mathbf{G}_{k+1}}\right)$ and making $\omega_{m,j,i}^{\mathbf{x}_{k+1}} = \omega_{m,j,i}^{\Delta\mathbf{G}_{k+1}}$.

4.9.4 Task Value Function and Policy

For the goal-covering task $V(\mathbf{x}_k)$ is independent of the instant *k* and is defined as:

$$V(\mathbf{x}_k) = \min\left(0, 1 - \frac{\beta_k}{\beta_0}\right), \tag{97}$$

where the *free goal angle* β_k corresponds to the maximum angle, defining an origin in the position of the ball, in which the own goal is not obstructed (see figure 3) and $\beta_0 = 0.514$ is a quantity related to the angle variance of the ball velocity after executing one of our available kicks. To calculate β_k, it is necessary to previously calculate $\mathbf{B}_k$:

$$\mathbf{B}_k = Rot\left(G_k^{\theta}\right)\tilde{\mathbf{B}}_k + \mathbf{G}_k^{pos}. \tag{98}$$

Our policy π for this task consists of positioning the robot over the bisector of the goal angle from the ball. With this, the potential attacker has two equal free goal angles to each side of the robot. The goalie must select then, at what distance d^* from the goal it should position itself. There is always a minimum distance $d_{\beta_k=0}^{\min}$ from the goal where the goalie can position itself to make $\beta_k = 0$. We will call $d_{zone}^{\max}$ the maximum distance to the goal where the goalie is still inside its penalty area. Then, we will select $d^* = \min\left(d_{\beta_k=0}^{\min}, d_{zone}^{\max}\right)$ to add the restriction that the goalie should stay inside its penalty area. The policy also intends to maintain the robot oriented towards the ball. Then, $\mathbf{u}_{k+1}^{act}$ corresponds to the displacement that moves the robot towards the desired pose as fast as possible considering the robot dynamics' restrictions. For the evaluation of the policy, we only take into account the mean of the belief[11].

In the next section, the results of simulated experiments for this application are presented.

---

[11] This policy π and the fact that it is only a function of the belief mean are not necessarily optimal from the decision-making point of view, but we have selected it because its simplicity helps us to show the applicability of the active vision system.

Chapter 5. Results and Discussion

This chapter presents the experiments that were performed to test the methods proposed in this thesis.

5.1 Experimental Setup

There are two types of experiments that were carried out in this thesis: simulated experiments and experiments using a real robot. Some of the simulated experiments were performed using synthetic data and others using data obtained from a robot simulator. All the experiments that use robot data (simulated or real) were performed using the robotics library UChileLib.

5.1.1 UChileLib

Our robotic soccer team has developed a robotics software library called UChileLib. The library is implemented in C++. UChileLib is multi-platform from the operative-system viewpoint (it has been successfully compiled and tested in Windows and Linux) and also from the hardware viewpoint (it has been successfully tested on the following robots: Aldebaran Nao and Hajime HR-18). UChileLib is not limited to robot soccer applications since it has a very flexible modular design. The part of the library that is in charge of handling modules, computer processes, threads, inter-process communications and sockets is based on boost libraries [107]. An important percentage of UChileLib was developed by the author of this thesis to allow the testing of the proposed methods.

5.1.2 Simulated Experiments

Simulated experiments with synthetic data are carried out using MATLAB.
Simulated robot experiments are performed using a simulator that allows the replicability of the results and a fair comparison of the tested methods. This simulator is called HLSim (from High Level Simulator) because it only simulates realistically the robot-controller's high level –world modeling and decision making– and it abstracts from the low level.

HLSim has a simplified 2D model of the robot and its environment, and the robot process and observations are modeled using configurable arbitrary noises over the perfect kinematic 2D model. In other words, HLSim does not model the system dynamics or the camera and vision issues. HLSim was developed by our robot soccer team and a significant proportion of its functionalities was developed by the author of this thesis.

5.1.3 Experiments on a Real Robot

5.1.3.1 Nao

Experiments using a real robot were performed on the Aldebaran Nao V3 robot. Nao is a humanoid robot with 21 degrees of freedom (5 in each leg, 1 in the hip, 3 in each arm and 2 in the neck). Its main sensors are two cameras in its head, with non-overlapping fields of view. Only one camera can be accessed at a time. Nao has also pressure, inertial, accelerometers, infrared, and sonar sensors. UChileLib accesses the Nao hardware using a firmware provided by the manufacturers called Naoqi [108].

5.1.3.2 Experiment Manager

Experiment Manager is a tool that facilitates experiments in which the robots that are performing the experiments must communicate and coordinate with a computer that is estimating the ground truth, and the human operator that is conducting the experiment. The application has embedded modules for the following: laser-based perception and pose estimation of robots and landmarks, ground-truth estimation, and experiment coordination. Figure 7 shows a block diagram of the functionality of the Experiment Manager. A laser sensor is used to obtain measurements that are then processed in a Ground-Truth module in order to get an estimation of the robot pose. The human operator may define desired trajectories of the robot and other events that are synchronized by the Experiment Coordination module. Finally, a Pose Controller module intends to control the robot pose by sending commands to the robot that should move it to the desired pose according to the currently set pose. Additionally, the Ground-Truth module logs in a file the whole robot trajectory measured. Finally, the robot itself stores all of the observations and odometries that occurred during the experiment. This system is described in detail in [27].

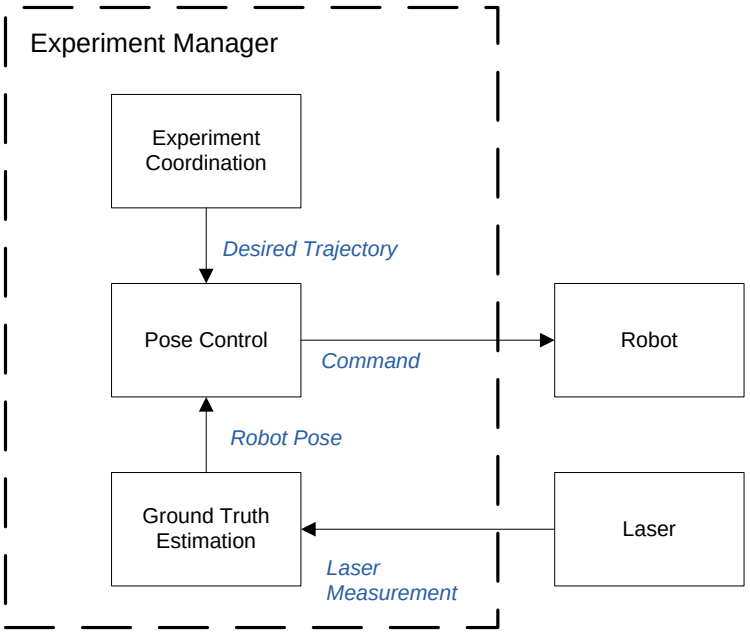


Figure 7. Experiment Manager Functionality Block Diagram.

5.2 Noise Parameters Regression

The objective of the first set of experiments is to validate the methodology presented for regression of the noise parameters.

5.2.1 SERV-HGP

The proposed methodology for HGP estimation was tested on simulated data as a means to measure its performance. The selected test is a one-dimensional regression problem. The training data is generated from the following functions:

$$X_i = U(0,10), \tag{99}$$

$$y_i = \sin(X_i) + e_i, \tag{100}$$

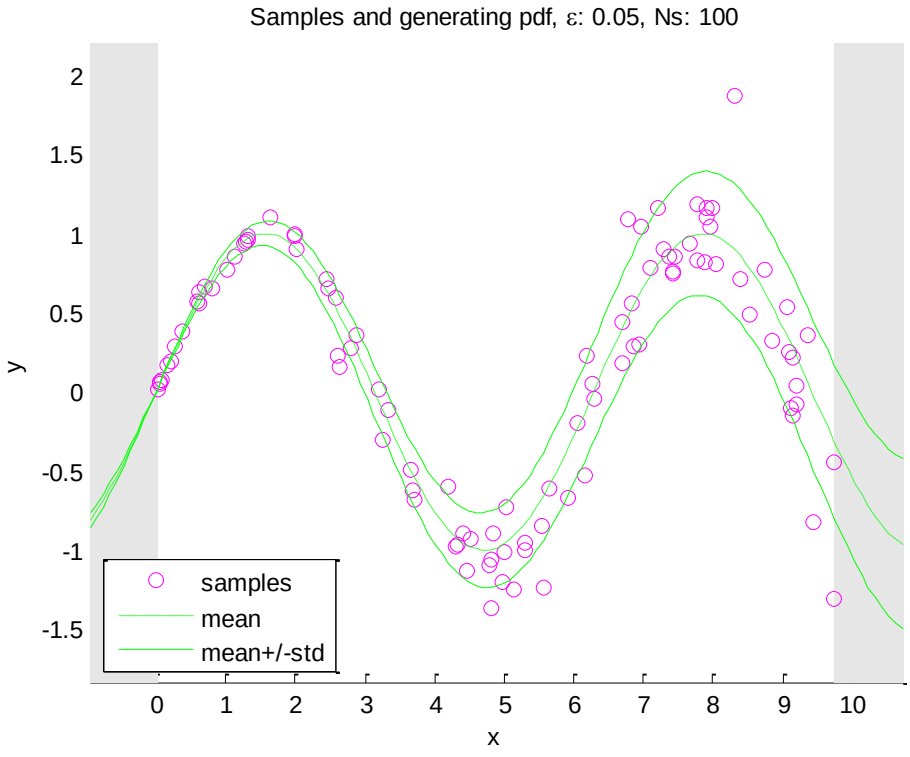
$$e_i = N\left(0,(\varepsilon X_i)^2\right). \tag{101}$$

To give a qualitative impression of the methodology's performance, figures 8 to 15 show examples of results obtained from the regression methodology. Figures 8 to 11 show these

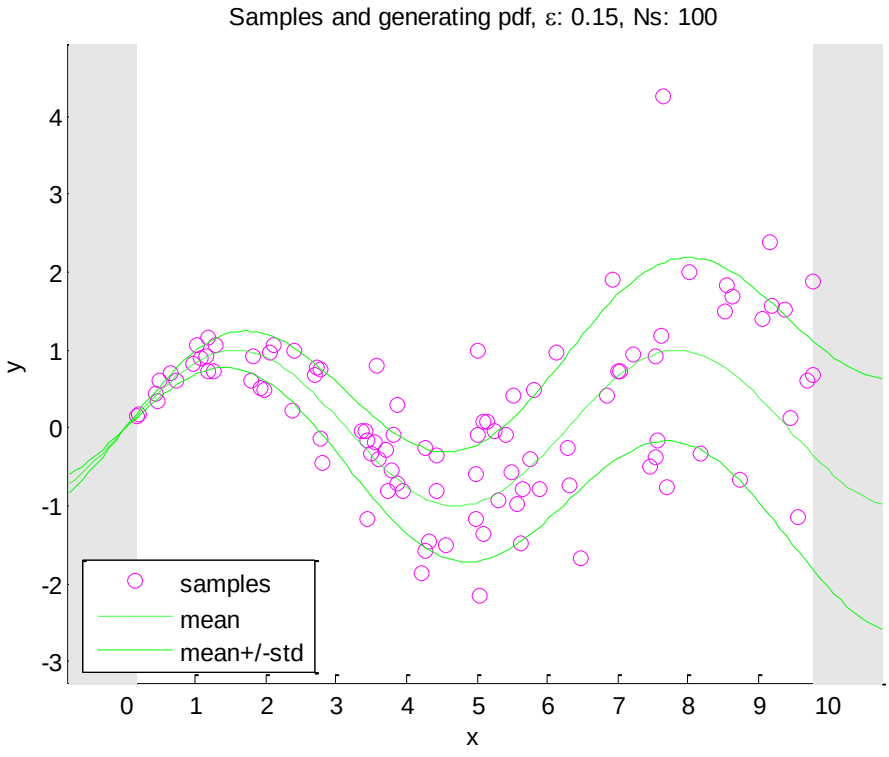
results using two sample sets (each set consists of 100 samples) with different values for ε (0.05 and 0.15). First, in figure 8, the sample points and the real mean and mean ± standard deviation (STD) of the generating pdf are shown. Then, figure 9 shows a comparison of the two regression methods in terms of determining the pdf mean and variance. The mean and variance regression appear separately in Figures 10 and 11 respectively. Figures 12 to 15 show the results of applying the regression methodology to two sample sets with different number of samples (100 and 300) and with the same value for ε (0.3). Again, in figure 12, the sample points and the real mean and mean ± STD of the generating pdf are shown. Then, figure 13 shows a comparison of the two regression methods in terms of determining the pdf mean and variance. The mean and variance regression appear separately in Figures 14 and 15 respectively. The results shown in figures 8 to 15 are dependent on the training sample set extracted from the pdf to estimate. Therefore, these figures are only illustrative of the qualitative differences between the results obtained by a GP regression and those obtained by the SERV-HGP regression, but they cannot be interpreted as a quantitative comparison.

A quantitative comparison among the performance of GP, ML-HGP and SERV-HGP regression on the simulated problem presented can be seen in tables 1 and 2. These tables show the MSE in the estimation of the mean and the standard deviation of the random variable for training sets with different numbers of samples and values of ε. In order to approximate the MSE, 20 simulations were run for each set of parameters. Table 1 compares the MSE in the estimation of the mean, while table 2 compares the MSE in the estimation of the standard deviation of the variable.

The first observation from the data in table 1 is that ML-HGP consistently outperforms GP in the estimation of the process mean. SERV-HGP has a performance close to that of ML-HGP for low noise values and a high number of training samples, but its performance on the estimation of the mean tends to decay faster than that of ML-HGP when the number of training samples decreases or the noise level increases, even to levels similar to that of GP.

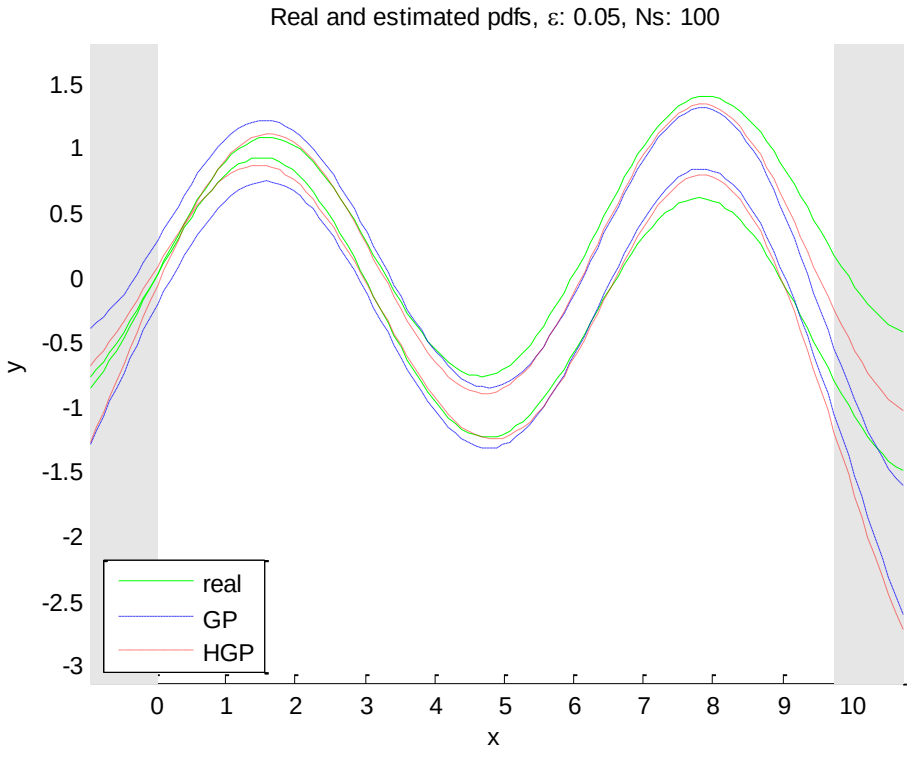


(a)

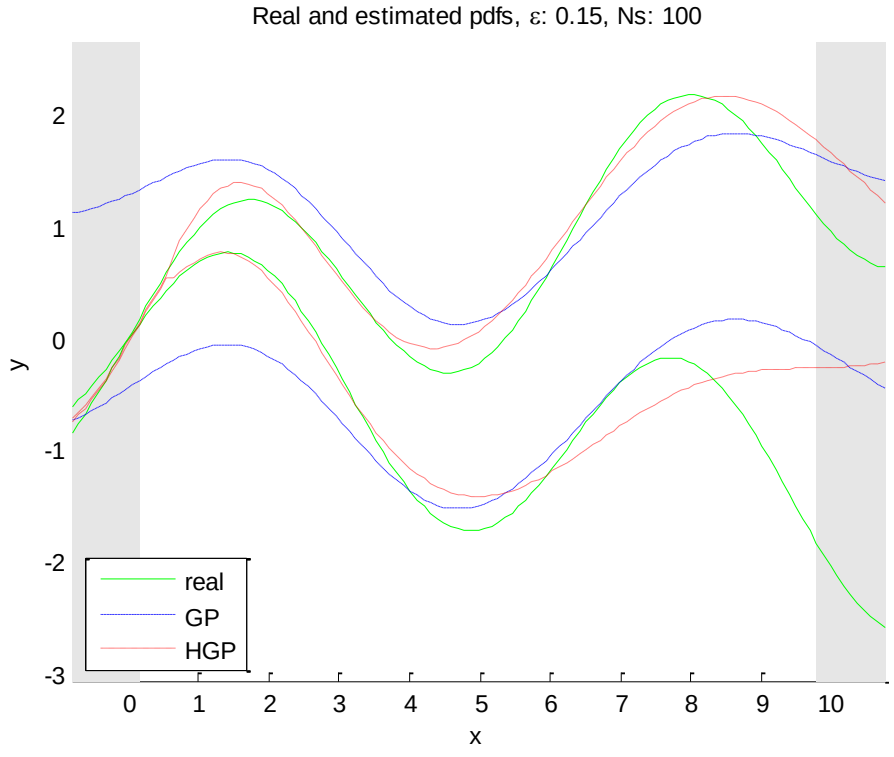


(b)

Figure 8. Samples (magenta circles) and the y mean (green dashed line) and mean ± variance (green solid lines) as a function of x, with (a) ε = 0.05 and (b) ε = 0.15.

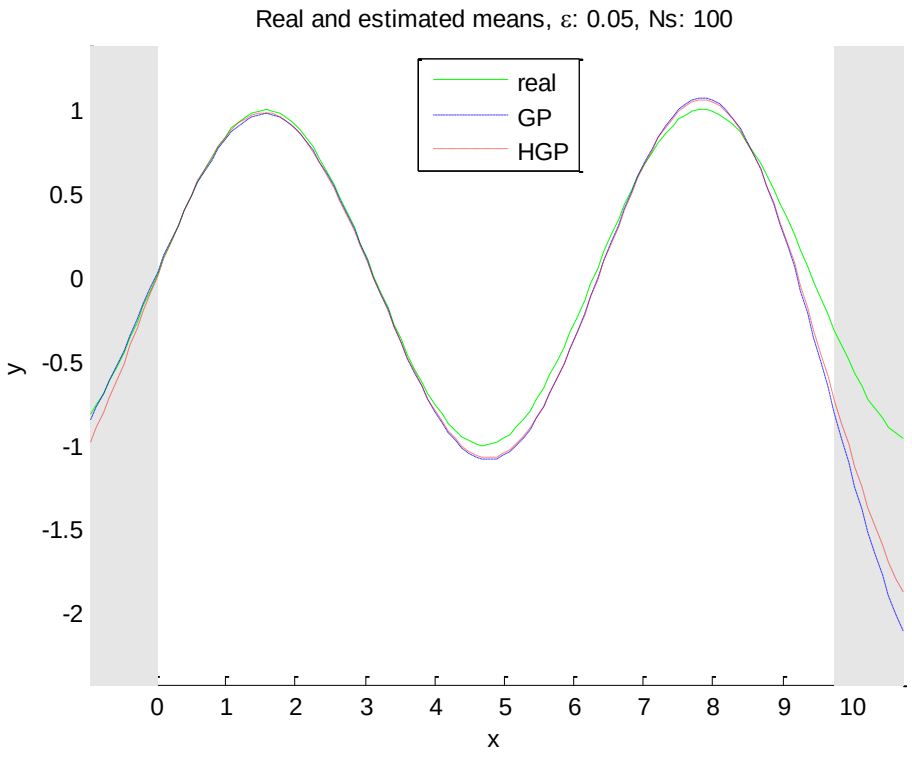


(a)

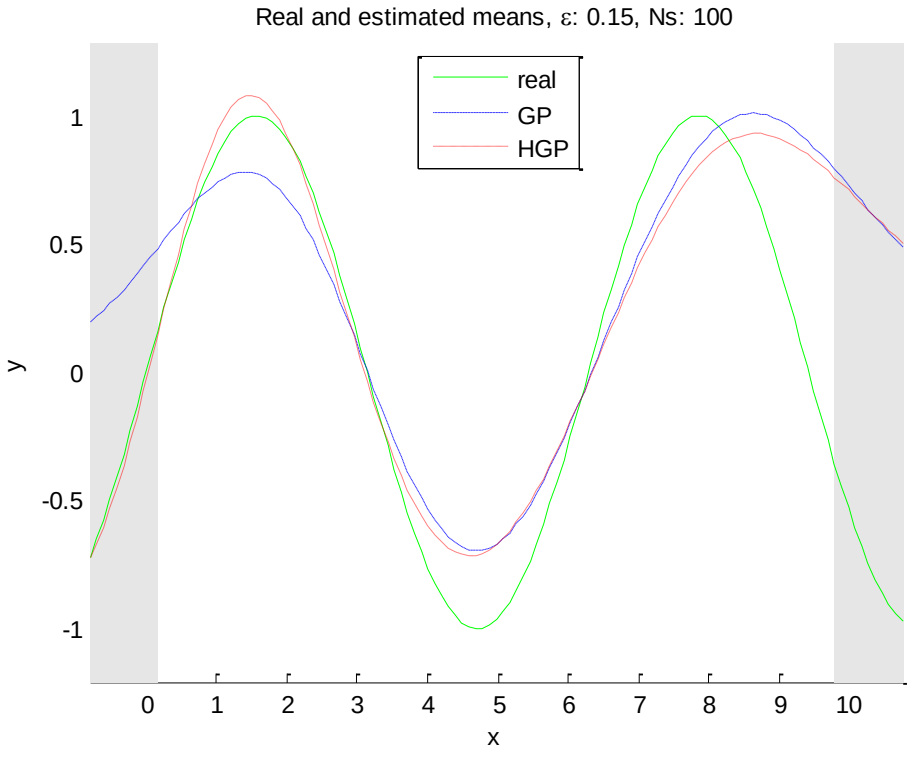


(b)

Figure 9. Samples (magenta circles) and the y mean (green dashed line) and mean ± variance (green solid lines) as a function of x, with (a) ε = 0.05 and (b) ε = 0.15. HGP stands for SERV-HGP.

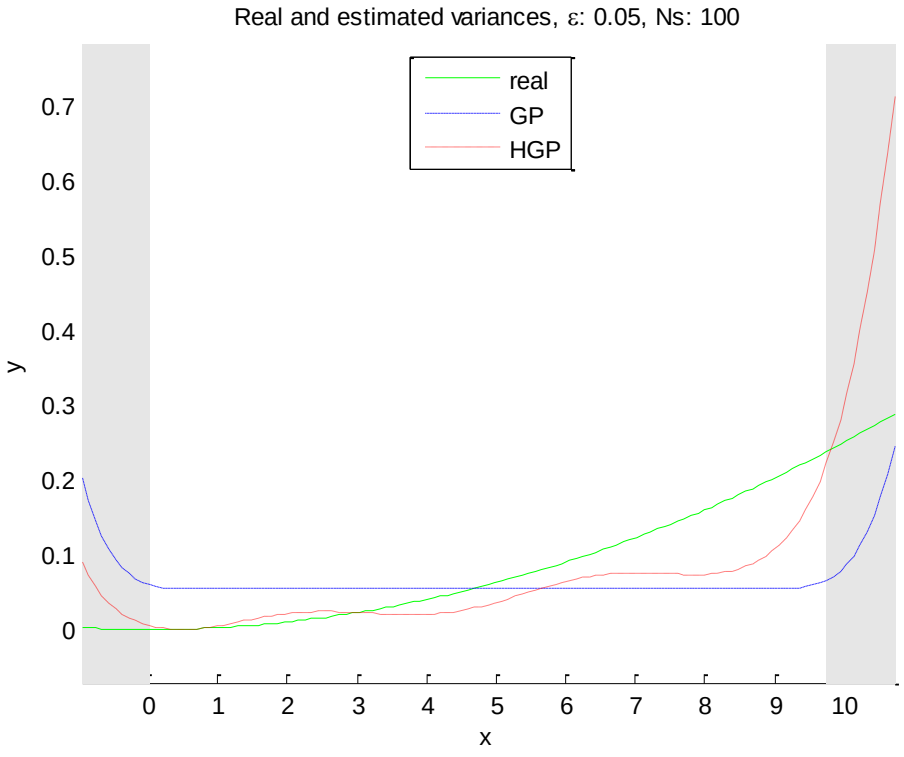


(a)

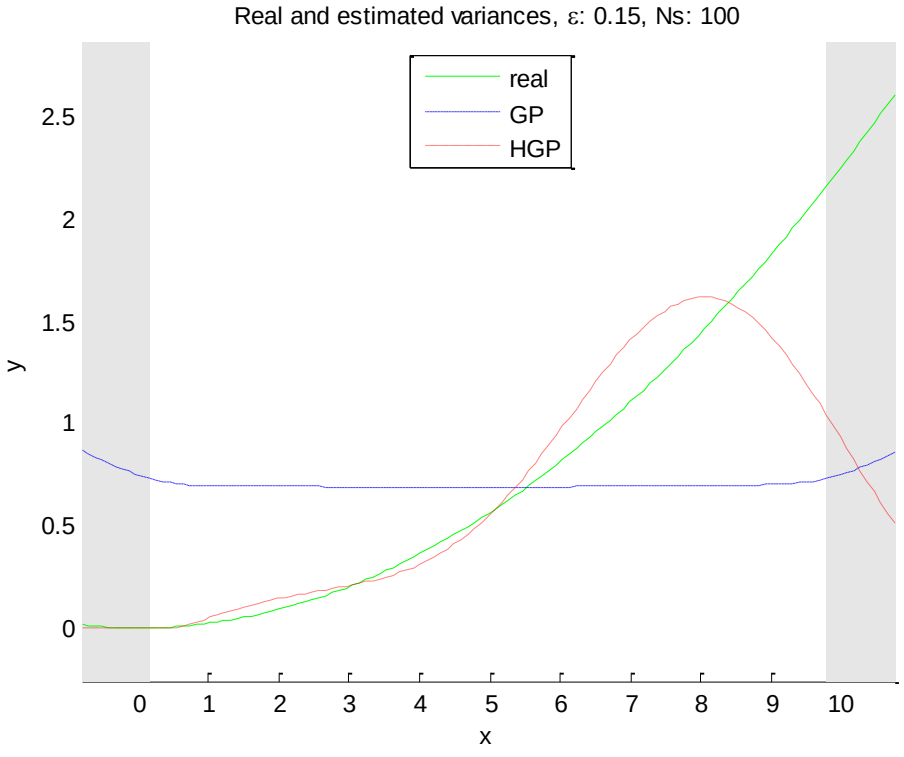


(b)

Figure 10. Samples (magenta circles) and the y mean (green dashed line) and mean ± variance (green solid lines) as a function of x, with (a) ε = 0.05 and (b) ε = 0.15. HGP stands for SERV-HGP.

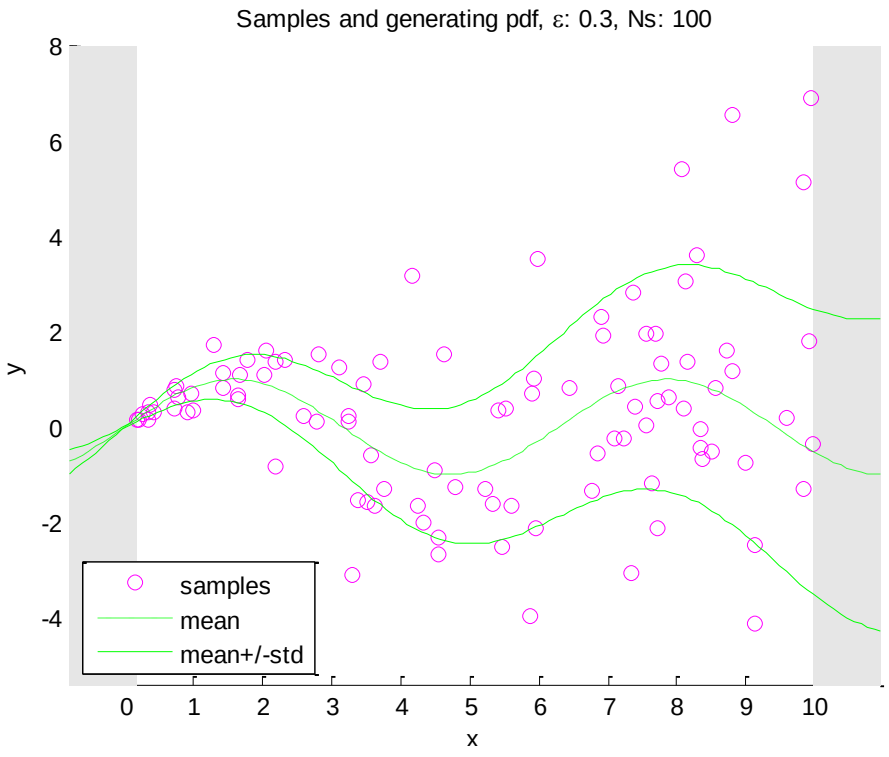


(a)

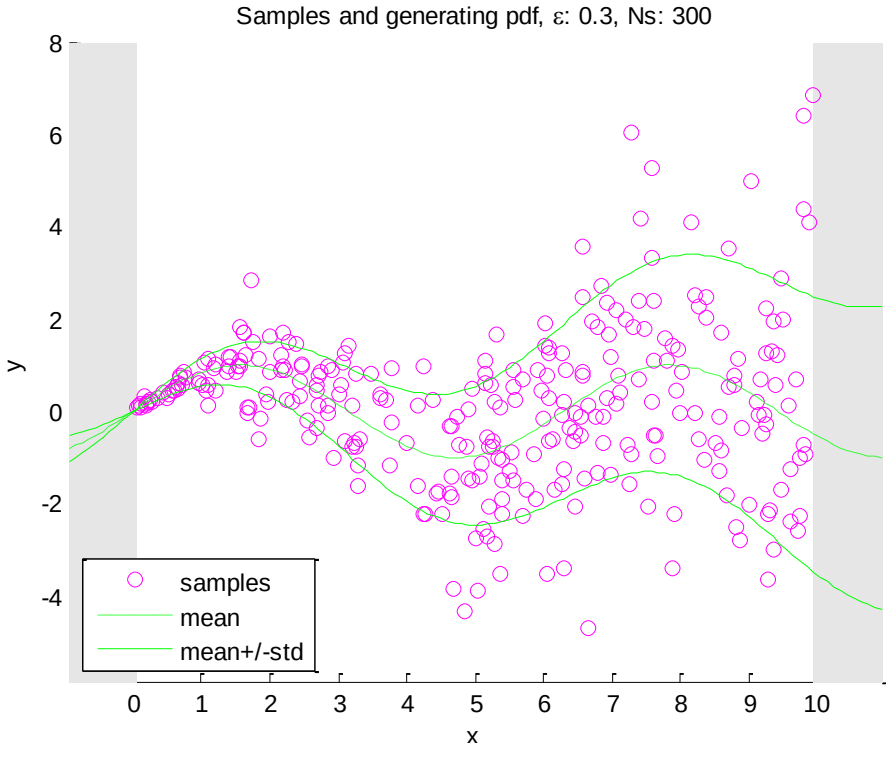


(b)

Figure 11. Samples (magenta circles) and the y mean (green dashed line) and mean ± variance (green solid lines) as a function of x, with (a) ε = 0.05 and (b) ε = 0.15. HGP stands for SERV-HGP.

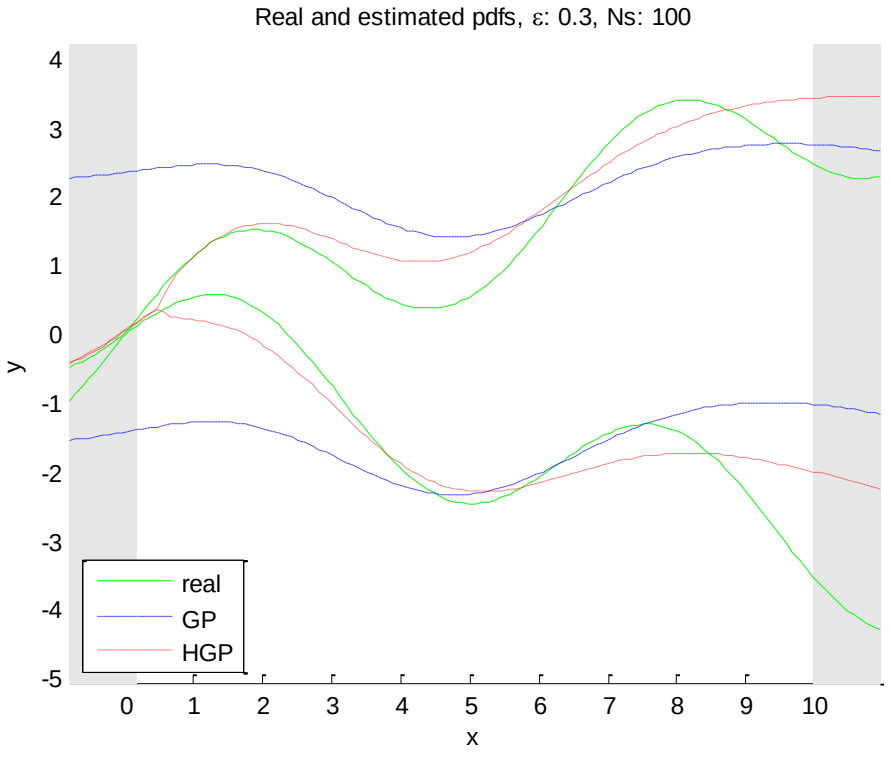


(a)

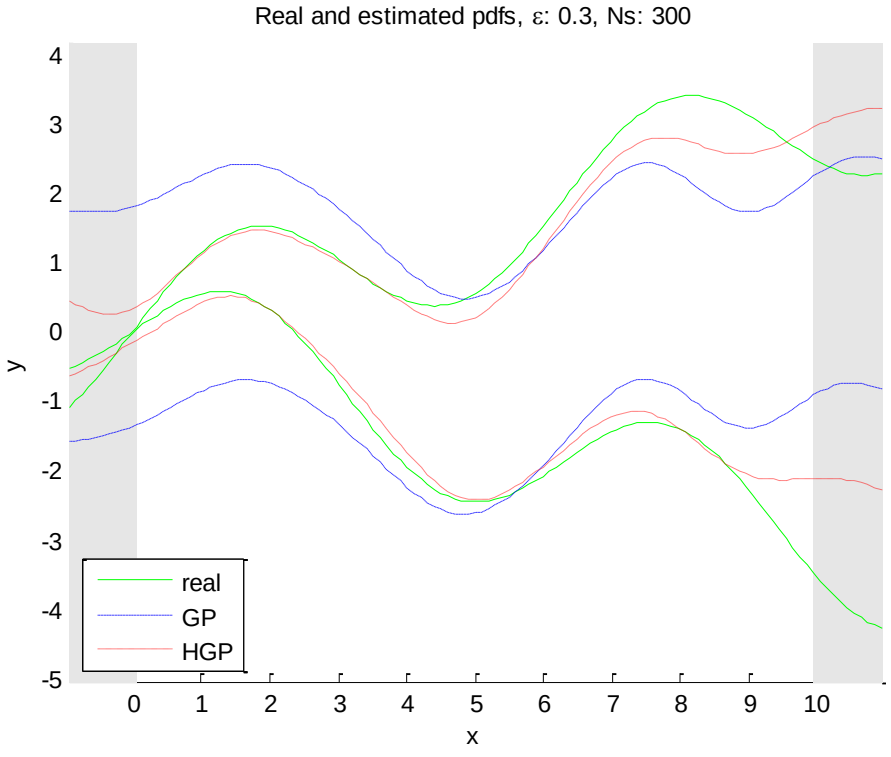


(b)

Figure 12. Samples (magenta circles) and the y mean (green dashed line) and mean ± variance (green solid lines) as a function of x, with (a) 100 training samples and (b) 300 training samples.

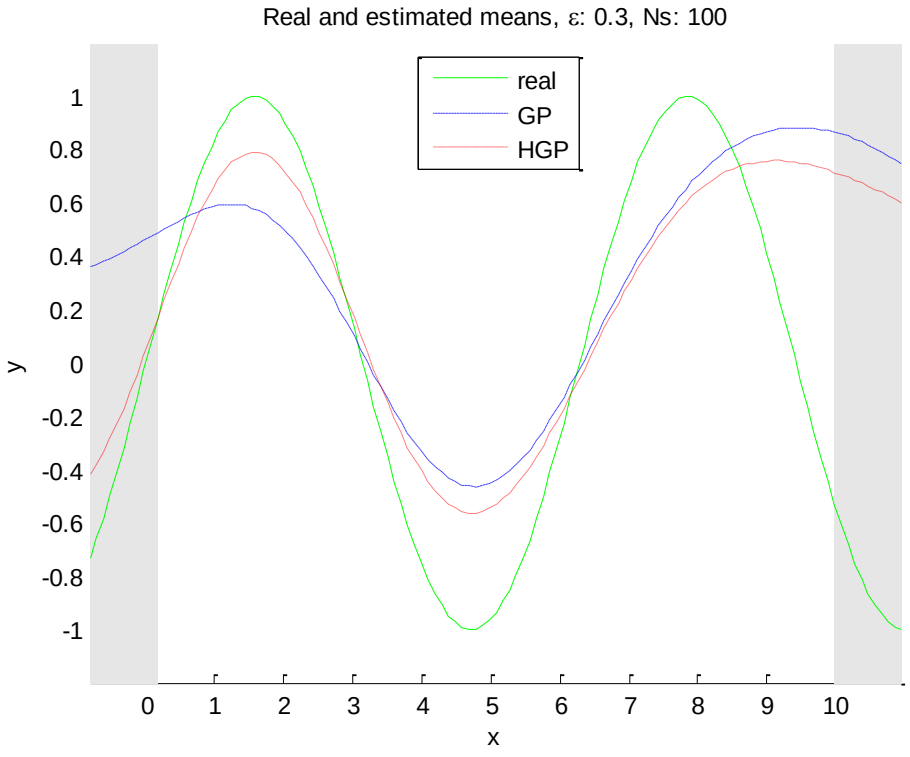


(a)

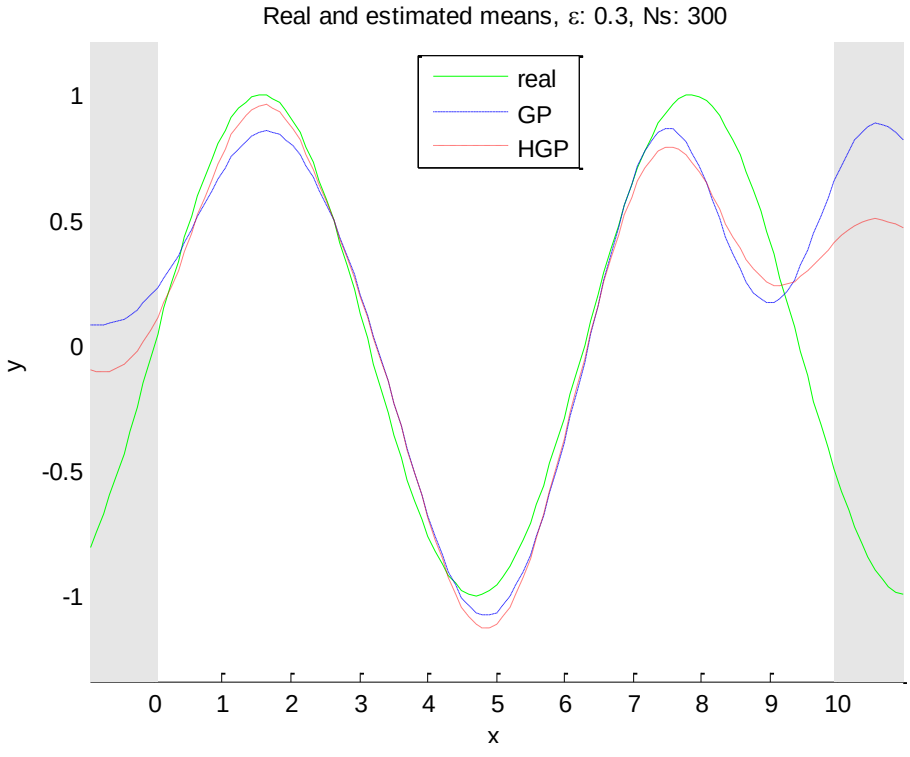


(b)

Figure 13. Samples (magenta circles) and the y mean (green dashed line) and mean ± variance (green solid lines) as a function of x, with (a) 100 and (b) 300 training samples. HGP stands for SERV-HGP.

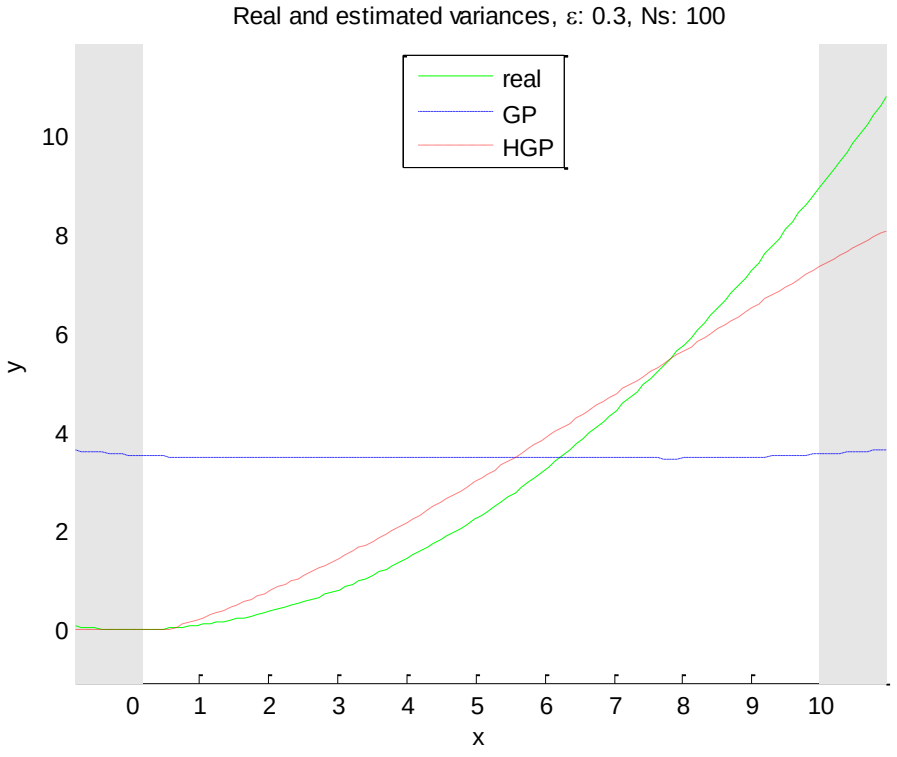


(a)

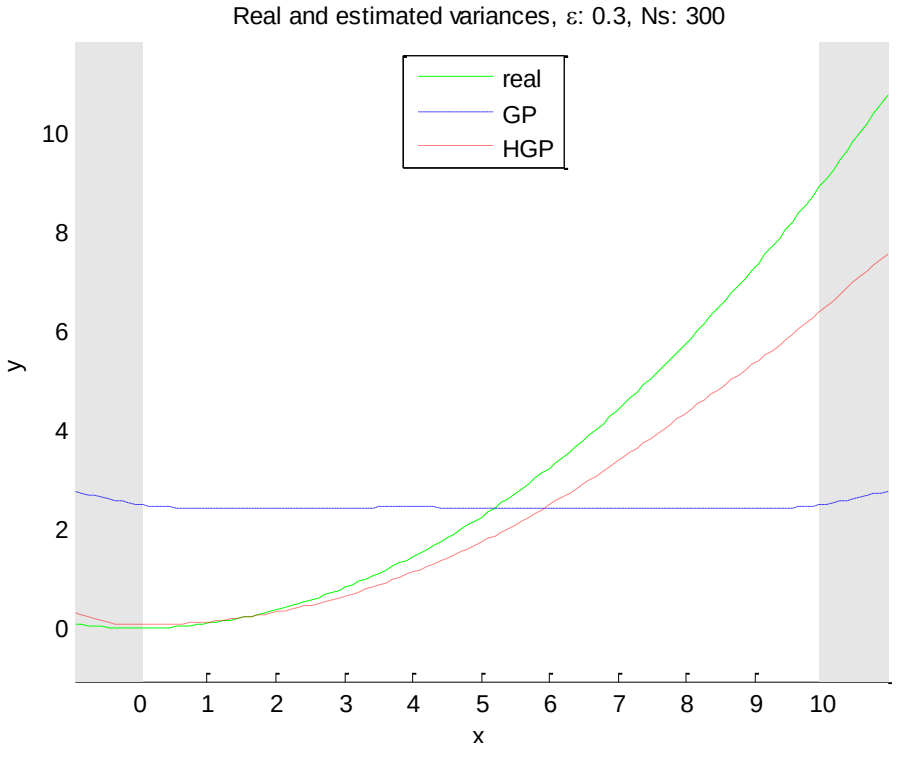


(b)

Figure 14. Samples (magenta circles) and the y mean (green dashed line) and mean ± variance (green solid lines) as a function of x, with (a) 100 and (b) 300 training samples. HGP stands for SERV-HGP.



(a)



(b)

Figure 15. Samples (magenta circles) and the y mean (green dashed line) and mean ± variance (green solid lines) as a function of x, with (a) 100 and (b) 300 training samples. HGP stands for SERV-HGP.

Table 1. Mean of the MSE in the estimation of the process mean.

ε	Method	Number of Training Samples				
		100	150	200	250	300
0.05	GP	0.0066	0.0054	0.0040	0.0028	0.0025
	ML-HGP	0.0058	0.0045	0.0035	0.0025	0.0020
	SERV-HGP	0.0056	0.0047	0.0034	0.0025	0.0020
0.1	GP	0.0404	0.0177	0.0158	0.0115	0.0146
	ML-HGP	0.0308	0.0150	0.0130	0.0082	0.0094
	SERV-HGP	0.0322	0.0152	0.0124	0.0085	0.0079
0.15	GP	0.0726	0.0591	0.0451	0.0379	0.0189
	ML-HGP	0.0486	0.0370	0.0256	0.0231	0.0186
	SERV-HGP	0.0506	0.0367	0.0250	0.0275	0.0159
0.2	GP	0.1109	0.0857	0.0490	0.0515	0.0555
	ML-HGP	0.0703	0.0645	0.0346	0.0430	0.0414
	SERV-HGP	0.0922	0.0828	0.0433	0.0526	0.0484
0.25	GP	0.1683	0.1315	0.0791	0.0678	0.0566
	ML-HGP	0.1116	0.0801	0.0604	0.0448	0.0384
	SERV-HGP	0.1507	0.1028	0.0821	0.0567	0.0390
0.3	GP	0.2992	0.1593	0.0992	0.1006	0.0799
	ML-HGP	0.1858	0.1148	0.0767	0.0554	0.0666
	SERV-HGP	0.3048	0.1698	0.0992	0.0674	0.0824

Table 2. Mean of the MSE in the estimation of the process standard deviation.

ε	Method	Number of Training Samples				
		100	150	200	250	300
0.05	GP	0.0251	0.0237	0.0239	0.0230	0.0231
	ML-HGP	0.0037	0.0048	0.0035	0.0042	0.0040
	SERV-HGP	0.0023	0.0024	0.0018	0.0013	0.0012
0.1	GP	0.1048	0.1092	0.1131	0.1384	0.0945
	ML-HGP	0.0163	0.0217	0.0166	0.0195	0.0199
	SERV-HGP	0.0334	0.0107	0.0071	0.0058	0.0051
0.15	GP	0.2289	0.2344	0.2220	0.2416	0.2518
	ML-HGP	0.0362	0.0441	0.0398	0.0392	0.0377
	SERV-HGP	0.0818	0.0189	0.0166	0.0116	0.0115
0.2	GP	0.3795	0.3715	0.3754	0.3873	0.3629
	ML-HGP	0.0608	0.0739	0.0795	0.0533	0.0861
	SERV-HGP	0.0415	0.0220	0.0192	0.0199	0.0199
0.25	GP	0.5719	0.5917	0.5821	0.5609	0.5609
	ML-HGP	0.1786	0.1178	0.1122	0.1267	0.1254
	SERV-HGP	0.0895	0.0405	0.0264	0.0256	0.0258
0.3	GP	0.7980	0.8207	0.8105	0.8106	0.8317
	ML-HGP	0.2199	0.1797	0.1860	0.1952	0.1420
	SERV-HGP	0.0848	0.0506	0.0322	0.0401	0.0247

From the data in table 2, it is possible to observe that both ML-HGP and SERV-HGP greatly outperform GP in the estimation of the process standard deviation. It is also possible to note, that with some exceptions, SERV-HGP outperforms ML-HGP in the estimation of the standard deviation of the process, in some cases by a noticeable ratio (up to approximately six times). These exceptions occur in the cases with few training samples which suggests that SERV-HGP could be less robust than ML-HGP to such cases.

Table 3 shows a comparison of the mean time consumption for the learning process of GP, ML-HGP and SERV-HGP regression on the simulated problem presented. The first observation from this table is that, as expected, GP is faster than both HGP methods. SERV-HGP is consistently faster in learning the HGP than ML-HGP. The former consumes approximately between 50% and 80% of the time consumed by the latter. For all the tested methods, the learning time grows with the number of training samples. GP and ML-HGP do not seem to increase significantly their learning time when the level of noise increases while SERV-HGP does. This fact suggests the convenience of further study the dependence of this time on the level of noise for SERV-HGP.

Table 3. Mean of the time consumption for the learning process (ms)

ε	Method	Number of Training Samples				
		100	150	200	250	300
0.05	GP	140	338	408	522	843
	ML-HGP	3,386	5,388	7,679	10,903	13,273
	SERV-HGP	1,795	2,626	3,6862	5,2078	7,895
0.1	GP	142	269	365	383	609
	ML-HGP	2,877	4,612	7,254	16,585	16,834
	SERV-HGP	1,803	2,549	3,607	6,133	8,831
0.15	GP	149	233	349	624	810
	ML-HGP	3,524	4,939	6,957	10,009	14,048
	SERV-HGP	1,904	2,703	3,990	5,307	7,209
0.2	GP	149	249	416	482	879
	ML-HGP	3,341	4,889	6,936	10,928	14,205
	SERV-HGP	1,776	2,533	3,327	5,238	9,241
0.25	GP	117	311	358	558	897
	ML-HGP	3,380	4,969	6,710	10,535	13,652
	SERV-HGP	1,827	2,985	4,098	4,583	7,959
0.3	GP	165	305	418	655	866
	ML-HGP	3,364	5,092	7,660	10,047	13,818
	SERV-HGP	1,996	2,644	4,235	5,101	8,958

5.2.2 Angle-GP

The Angle-GP method is tested using synthetic data generated in MATLAB. Angle-GP performance is compared to a regular GP and to SinCos-GP. In this experiment, all the previously mentioned methods are used to perform the regression of the following function:

$$y = \operatorname{atan}2\left(x_2, x_1\right) + 45^{\circ}. \tag{102}$$

The atan2 function has a two-dimensional input $\left(\mathbf{p}_y, \mathbf{p}_x\right)$ that corresponds to the Cartesian coordinates of a point $\mathbf{p}$, and the output is the angle $\mathbf{p}_\theta$ of the polar coordinates of $\mathbf{p}$. Regardless of the definition of the output interval of atan2, the function is discontinuous at one point. The definition of the output interval selected for the purposes of this experiment is $\left(-180^{\circ}, 180^{\circ}\right]$. The size of the training set varied from 5 to 100 samples, increasing in increments of 5. For each training set size, 100 trials were performed, with different randomly sampled training sets. For each trial, 400 random test points were used to check the regression accuracy of the methods.

For each trial and for each method, the mean squared error[12] of the prediction was calculated in order to have a global comparison measure. Table 4 and figure 16 show the results in terms of comparative performance for the described regression task using: a regular GP, a SinCos-GP, and an Angle-GP. From these results, it is clear that Angle-GP surpasses the performance of both the regular GP and SinCos-GP. As the number of training samples grows, the three methods being compared improve their predictions and the accuracy of SinCos-GP becomes closer to that of Angle-GP. However, the regular GP is not able to get a low MSE even with a high number of training samples.

---

[12] Note that the angular error cannot be obtained by a simple subtraction. Sometimes $\pm 360^{\circ}$ must be added to the result of the subtraction in order to get an error in the $\left(-180^{\circ}, 180^{\circ}\right]$ interval.

In order to explore the behavior of the described methods in more detail, their prediction performances over different subintervals of the output space were measured. For this purpose, the average prediction error was calculated for different subsets of the test sets.

Table 4. Mean ± STD of the MSE in the described regression task for the compared methods.

Number of Training Samples	Regular GP	SinCos-GP	Angle-GP
5	1.6741 ± 0.5992	1.1444 ± 0.7371	0.5975 ± 0.5531
10	1.1941 ± 0.6416	0.2124 ± 0.3031	0.1203 ± 0.1585
15	1.1104 ± 0.5966	0.0937 ± 0.1603	0.0620 ± 0.0643
20	0.9142 ± 0.3949	0.0593 ± 0.0601	0.0336 ± 0.0324
25	0.7804 ± 0.2432	0.0486 ± 0.0482	0.0275 ± 0.0289
30	0.7626 ± 0.2491	0.0429 ± 0.0463	0.0273 ± 0.0436
35	0.7027 ± 0.2188	0.0299 ± 0.0277	0.0190 ± 0.0214
40	0.6562 ± 0.2503	0.0286 ± 0.0319	0.0157 ± 0.0148
45	0.6317 ± 0.1502	0.0259 ± 0.0290	0.0139 ± 0.0143
50	0.5978 ± 0.1521	0.0221 ± 0.0233	0.0140 ± 0.0141
55	0.5998 ± 0.1133	0.0164 ± 0.0172	0.0113 ± 0.0129
60	0.5428 ± 0.1564	0.0181 ± 0.0230	0.0132 ± 0.0203
65	0.5417 ± 0.1282	0.0137 ± 0.0163	0.0105 ± 0.0131
70	0.5331 ± 0.1226	0.0135 ± 0.0135	0.0114 ± 0.0127
75	0.5325 ± 0.1324	0.0119 ± 0.0106	0.0101 ± 0.0126
80	0.5054 ± 0.1194	0.0118 ± 0.0125	0.0098 ± 0.0113
85	0.4876 ± 0.1165	0.0101 ± 0.0110	0.0084 ± 0.0107
90	0.4817 ± 0.1207	0.0111 ± 0.0120	0.0101 ± 0.0132
95	0.4523 ± 0.1044	0.0091 ± 0.0103	0.0079 ± 0.0095
100	0.4533 ± 0.0982	0.0103 ± 0.0134	0.0081 ± 0.0087

Each subset corresponds to a subinterval of the output space. Figures 17 and 18 show the prediction MSE through the whole experiment for each angle interval, with 5 and 100 training samples respectively. Figures 18.a and 18.b plot the same information, but in figure 18.b the regular GP's results are not shown in order to graphically compare the two other methods in more detail. From figures 17 and 18.a, it is possible to infer that the cause of the poor performance of the regular GP on the angle regression task is mainly due to the discontinuity in 180º. Figures 17 and 18.b show no apparent correlation between the output angle and the performances of SinCos-GP and Angle-GP. These figures also confirm the consistent superiority of Angle-GP over SinCos-GP, which can be explained by the selection of a more adequate log marginal likelihood.

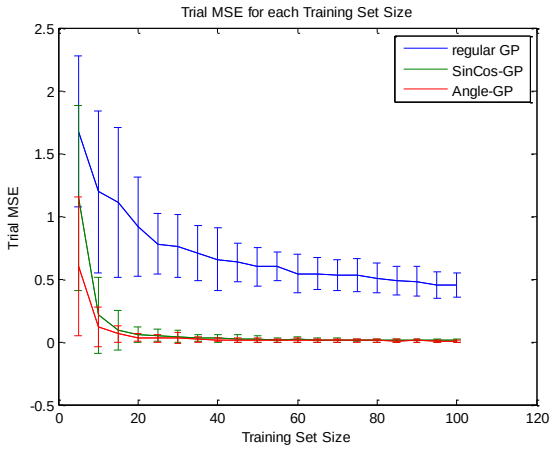


Figure 16. Comparison among methods of trial MSE mean and variance for each training set size.

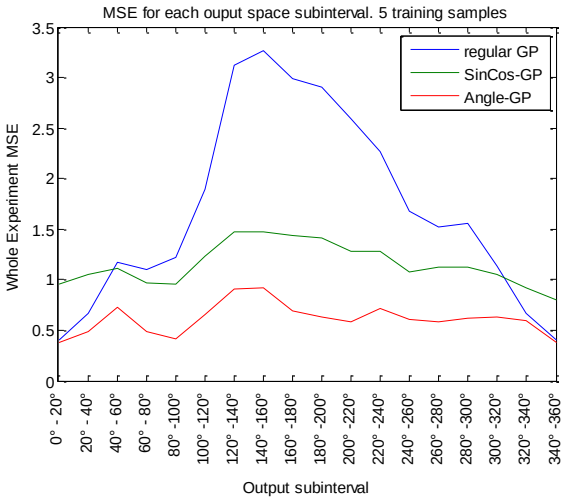
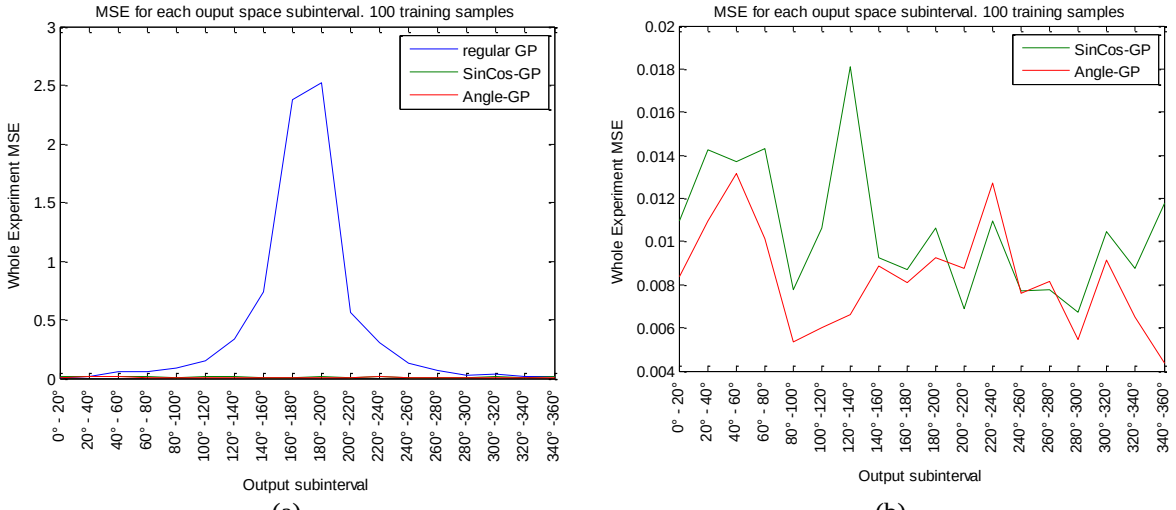


Figure 17. Comparison among methods of whole experiment MSE for 5 training samples.



(a) (b)

Figure 18. Comparison among methods of whole experiment MSE for 100 training samples. (a) The three methods (b) only SinCos-GP and Angle-GP as a means to have more detail.

5.2.3 HGP-EKF

In order to test the proposed HGP-EKF methodology, two similar experiments are performed. The first experiment is run in the HLSim simulator while the second one is run in a real NAO robot. For each experiment, an HGP-EKF is compared with a regular EKF and GP-EKF. The problem to be solved is that of object tracking, the tracked object being a ball. The theoretical observational model to be tested is the identity, i.e., for each state of the ball, the observation is supposed to be equal to the state. In the simulated case, both the observational and process theoretical models are contaminated with synthetic noises that make the regular EKF perform poorly.

5.2.3.1 Training the Models

The method selected for training the models is divided into two procedures: the training procedure for the process model, and the training procedure for the observational model.

5.2.3.1.1 Process Model Training Procedure

The main assumption of this procedure is that a differential process model exists. This means that the process model can be written in its differential form, without losing much relevant information[13]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}^{diff}\left(\mathbf{u}_k, \mathbf{a}_k^f, \mathbf{w}_k\right). \tag{103}$$

For a mobile robot, this assumption is valid when the surface over which the robot will move is somehow homogeneous in the sense of having similar properties in all of its regions.

Additionally, this training procedure assumes that there is a way to accurately estimate the robot's pose while the training process is being carried out. This estimation is based on the observations provided by an external sensor that can perceive the robot's pose or by a

[13] Note that in this definition we are already considering the additional parameters defined in subsection 3.3. Of course, this definition can be analogously made for the standard case without additional parameters.

landmark whose pose can be detected by the robot with high accuracy while the robot is moving around the training area.

5.2.3.1.2 Observational Model Training Procedure

A training data set has been acquired for the observational model of each object. The data set is based on a polar grid over the relative-position space (the space of the positions of the object relative to the robot). Since the objects under consideration can be observed in different ranges of distances, different grid resolutions and limits have been selected for each object. There are errors in the measurements from encoders and in the synchronism between them and the acquired images. Therefore, the observational noise parameters might depend on the velocity of the head and the body. To account for this source of error, for each grid cell the following head behaviors will be executed:

- Point the camera directly to the object
- Alternate between pointing to the object and pointing to other position
- Make a fixed searching movement.

For each of the three head behaviors, two different body behaviors will be executed at each grid cell:

- Walk in place
- Stand

Thus, there are 6 behavior combinations for the head and body behaviors. For each of these combinations, and for each cell grid, the data set contains 5 training samples. As such, each cell in the grid contains 30 training samples. For this experiment, the only object whose observational model is trained is the ball. A 5x3 grid was trained with angles (-80°, -40°, 0°, 40°, 80°) and distances (1m, 2m, 3m). Since the grid has 15 cells, the number of training samples is 450.

5.2.3.2 Description of the Experiments

For both experiments (simulated and real), the procedure for training the models described in section 5.2.3.1 is performed prior to the experiment and the same training samples are

used to train the respective models (a fixed Gaussian for the regular EKF, a GP for GP-EKF, and a HGP for HGP-EKF).

Both the simulated and real experiments consist of measuring the performance of the estimation of the relative position of an object while the robot is moving according to an arbitrary trajectory. The tracked object is a fixed orange ball and the robot's intended trajectory is the one shown in figures 19 and 30. The robot's body intends to follow the mentioned trajectory while the robot's head intends to look towards the ball when it is possible.

The task that all of the compared methods perform during the experiment is the estimation of the relative position of the ball. No other estimation, such as self-localization or the relative estimation of another object, is performed.

While each experiment runs, the robot's perceptions and odometries and the ground-truth data from the simulator are stored. Then, after the execution of the experiment, the perceptions and odometries sequences are used to estimate the relative position of the object using each of the methods being compared.

5.2.3.3 Compared Methods

This subsection describes the methods that are compared in this experiment.

The first method (denoted *EKF*) is a standard EKF in which a theoretical model is used for both the observational and the process models. The noises are considered additive and Gaussian with fixed parameters (mean and covariance matrix) calculated from the training data.

For all the remaining methods, the process model is considered to be composite (see subsection 3.5), with the process GP or HGP estimating the robot dynamics (see equations (71), (73), (76) and (77)).

The second method (denoted *GP-EKF*) under comparison corresponds to the GP-EKF method described in section 1.6.5, while the third method being compared (denoted *HGP-EKF*) corresponds to the HGP-EKF method described in section 3.1. In both GP-EKF and

HGP-EKF, no theoretical information is used but the regression of $f_{dyn}(\mathbf{u},\mathbf{w})$ and $h(\mathbf{x},\mathbf{u},\mathbf{v})$ is performed directly from the data.

Finally, the fourth and the fifth methods (noted respectively *GP-EKF+Model* and *HGP-EKF+Model*) are modified versions of GP-EKF and HGP-EKF that consider information from the theoretical model. This information is considered by means of making $f_{dyn}(\mathbf{u},\mathbf{w})$ and $h(\mathbf{x},\mathbf{u},\mathbf{v})$ additive (see equations (70) and (104)):

$$f_{dyn}(\mathbf{u},\mathbf{w}) = f_{dyn,th}(\mathbf{u},\mathbf{w}) + GP_{\mu}^{\Delta \mathbf{f}_{dyn}}([\mathbf{u}]). \tag{104}$$

Note that equation (104) is analogous to equation (69).

Finally, with the purpose of testing the utility of additional model parameters (proposed in subsection 3.3), each of the tested methods, except for EKF, is tested with and without the use of additional model parameters.

5.2.3.4 Simulated Experiment

In order to theoretically test the proposed HGP-EKF methodology, a simulated experiment is performed.

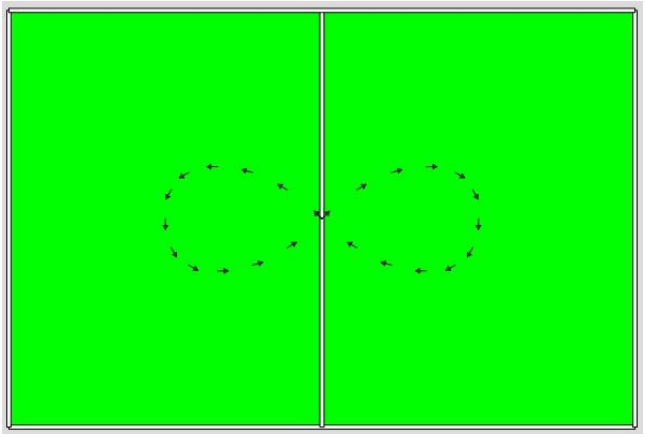


Figure 19. Intended robot trajectory for the HGP-EKF simulated experiment.

The simulator adds a noise to every robot movement or observation. This noise is Gaussian with unknown parameters to the robot but the robot gets the odometries and observations

with their respective additional parameters. These additional parameters are also generated by the simulator using a pdf that is related to the actual robot movement or observation. Again, the pdfs of the additional parameters and their relationships with the actual robot movement or observation are unknown to the robot.

Two vectors of latent variables are defined for the robot movements and observations, respectively, in order to model the relationship between the additional parameters and the observations or movements:

$$\ell_w = N(0,1), \tag{105}$$

$$\ell_v = N(0,1). \tag{106}$$

The observational and process noises are sampled from:

$$\mathbf{w} = N(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w), \tag{107}$$

$$\mathbf{v} = N(\boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v), \tag{108}$$

with

$$\boldsymbol{\mu}_w = \mathbf{A}_w \tilde{\mathbf{u}}, \tag{109}$$

$$\boldsymbol{\Sigma}_w = \sum_i \mathbf{B}_{w,i} \tilde{\mathbf{u}} \left(\mathbf{B}_{w,i} \tilde{\mathbf{u}}\right)^T, \tag{110}$$

$$\boldsymbol{\mu}_v = \mathbf{A}_v \tilde{\mathbf{x}}_o, \tag{111}$$

$$\boldsymbol{\Sigma}_v = \sum_i \mathbf{B}_{v,i} \tilde{\mathbf{x}}_o \left(\mathbf{B}_{v,i} \tilde{\mathbf{x}}_o\right)^T. \tag{112}$$

Where $\mathbf{A}_w$, $\{\mathbf{B}_{w,i}\}$, $\mathbf{A}_v$ and $\{\mathbf{B}_{v,i}\}$ are matrices whose arbitrary values are shown in Appendix 2, and $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{x}}_o$ are defined as:

$$\tilde{\mathbf{u}} = \begin{bmatrix} \mathbf{u} \\ \ell_f \end{bmatrix}, \tag{113}$$

$$\tilde{\mathbf{x}}_o = \begin{bmatrix} \mathbf{x}_o \\ \ell_h \end{bmatrix}. \tag{114}$$

Figure 20 illustrates the resulting observational noises by plotting the training samples for the observational model that the simulated robot obtained (see subsection 5.2.3.1 for details).

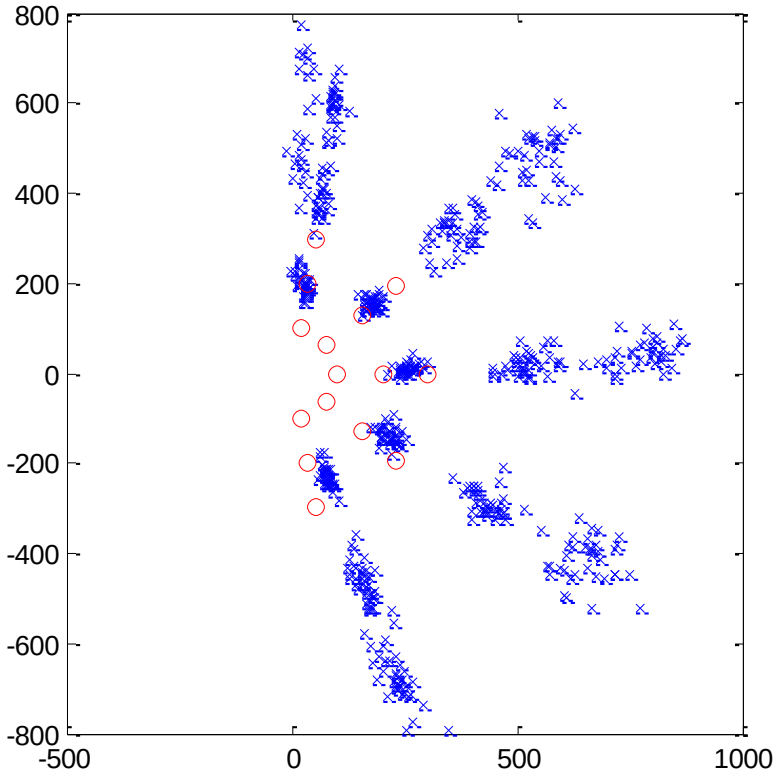


Figure 20. Training samples for the observational model. The red circles correspond to the actual position of the ball while the blue crosses show the perceived positions of the ball.

While the additional-parameters' vectors are sampled from:

$$\mathbf{a}^f = N\left(\boldsymbol{\mu}_{\mathbf{a}}^f, \boldsymbol{\Sigma}_{\mathbf{a}}^f\right), \tag{115}$$

$$\mathbf{a}^h = N\left(\boldsymbol{\mu}_{\mathbf{a}}^h, \boldsymbol{\Sigma}_{\mathbf{a}}^h\right), \tag{116}$$

With

$$\boldsymbol{\mu}_{\mathbf{a}}^f = \mathbf{A}_{\mathbf{a}}^f \ell_f, \tag{117}$$

$$\boldsymbol{\Sigma}_{\mathbf{a}}^f = \sum_i \mathbf{B}_{\mathbf{a},i}^f \ell_f \left(\mathbf{B}_{\mathbf{a},i}^f \ell_f\right)^T, \tag{118}$$

$$\boldsymbol{\mu}_{\mathbf{a}}^{h} = \mathbf{A}_{\mathbf{a}}^{h} \ell_{h}, \tag{119}$$

$$\boldsymbol{\Sigma}_{\mathbf{a}}^{h} = \sum\nolimits_{i} \mathbf{B}_{\mathbf{a},i}^{h} \ell_{h} \left(\mathbf{B}_{\mathbf{a},i}^{h} \ell_{h} \right)^{T}, \tag{120}$$

where $\mathbf{A}_{\mathbf{a}}^{f}$, $\left\{ \mathbf{B}_{\mathbf{a},i}^{f} \right\}$, $\mathbf{A}_{\mathbf{a}}^{h}$ and $\left\{ \mathbf{B}_{\mathbf{a},i}^{h} \right\}$ are matrices whose arbitrary values are shown in Appendix 2.

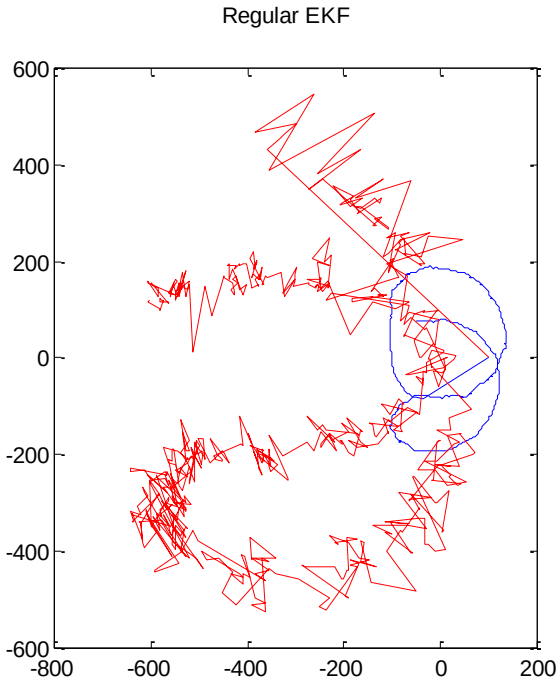


Figure 21. Estimated trajectory of the relative position of the ball using a Regular EKF (red) and the corresponding GT (blue).

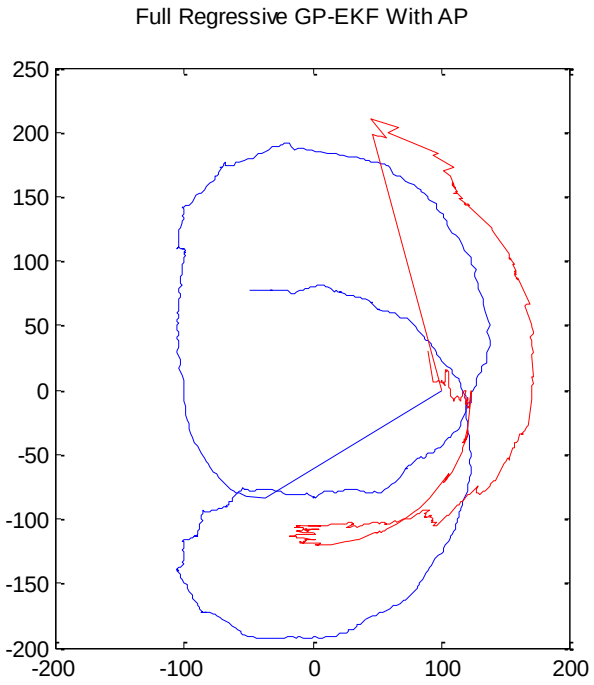


Figure 22. Estimated trajectory of the relative position of the ball using a Full Regressive GP-EKF with additional parameters (red) and the corresponding GT (blue).

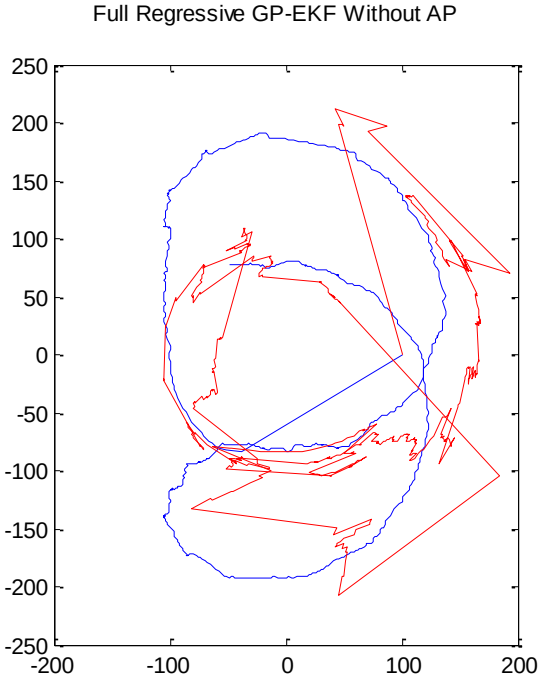


Figure 23. Estimated trajectory of the relative position of the ball using a Full Regressive GP-EKF without additional parameters (red) and the corresponding GT (blue).

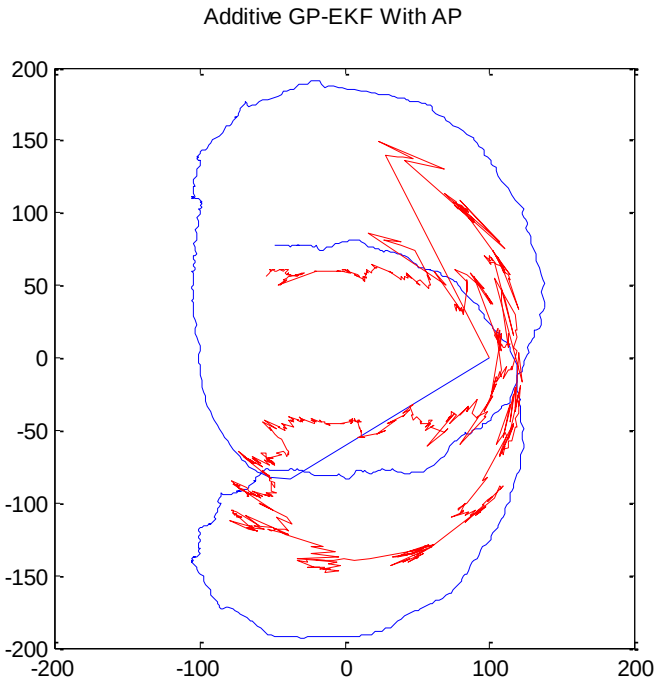


Figure 24. Estimated trajectory of the relative position of the ball using an Additive GP-EKF with additional parameters (red) and the corresponding GT (blue).

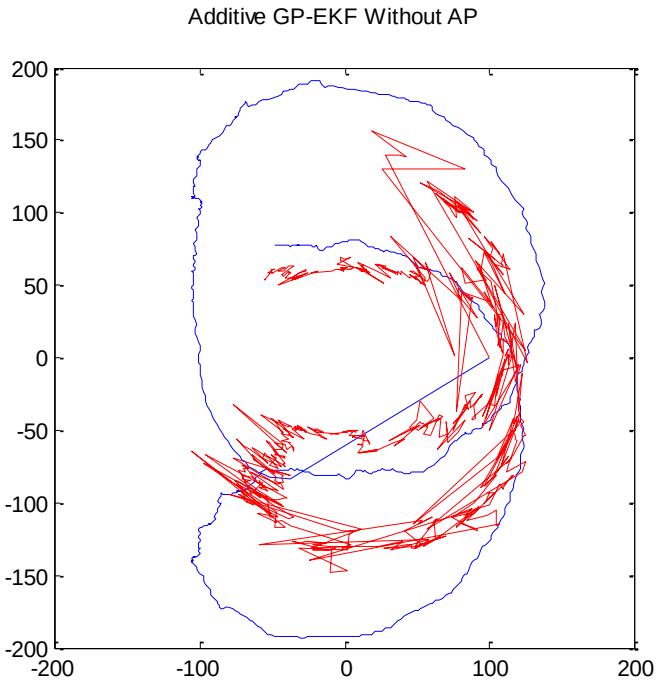


Figure 25. Estimated trajectory of the relative position of the ball using an Additive GP-EKF without additional parameters (red) and the corresponding GT (blue).

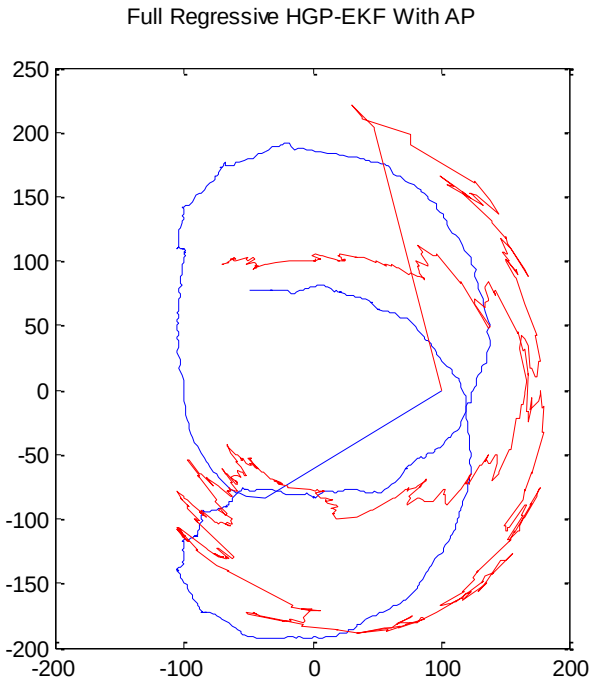


Figure 26. Estimated trajectory of the relative position of the ball using a Full Regressive HGP-EKF with additional parameters (red) and the corresponding GT (blue).

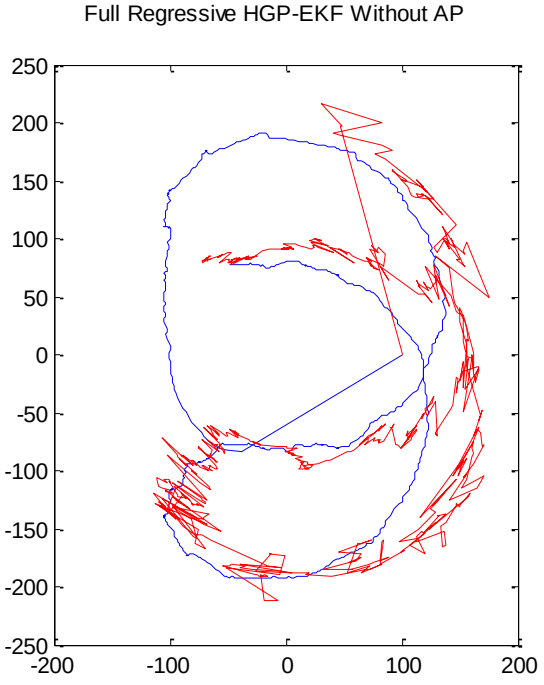


Figure 27. Estimated trajectory of the relative position of the ball using a Full Regressive HGP-EKF without additional parameters (red) and the corresponding GT (blue).

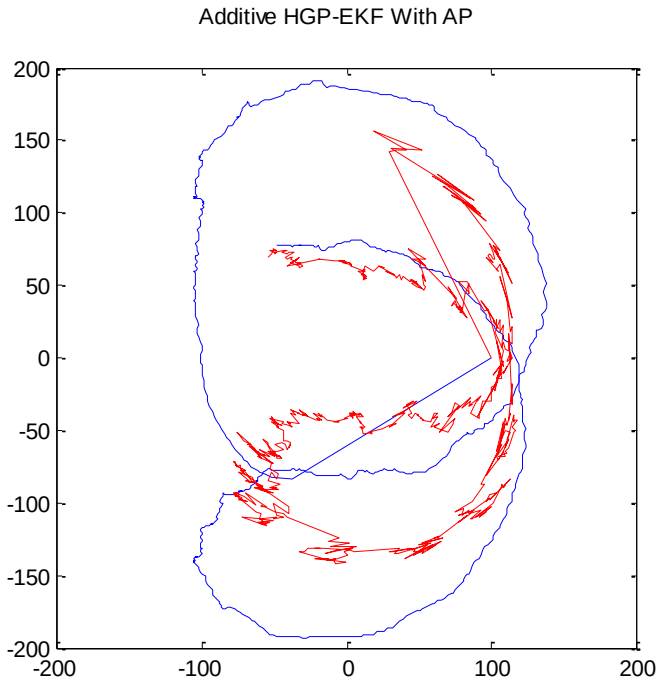


Figure 28. Estimated trajectory of the relative position of the ball using an Additive HGP-EKF with additional parameters (red) and the corresponding GT (blue).

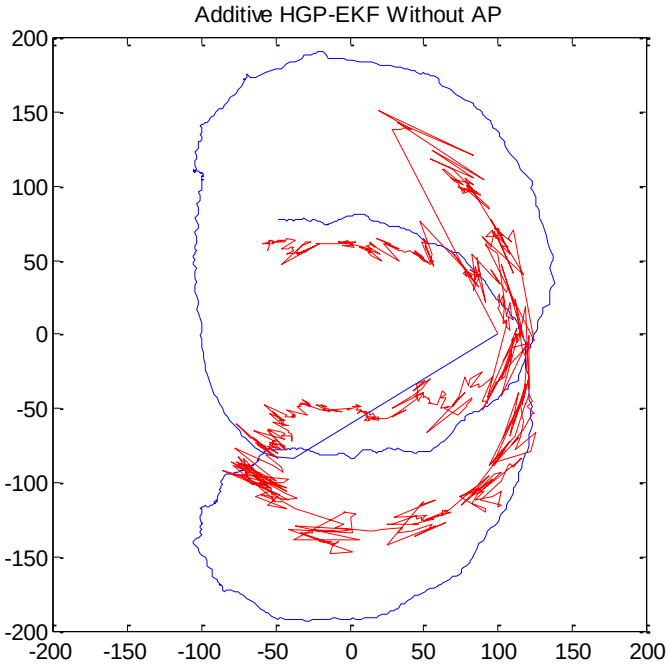


Figure 29. Estimated trajectory of the relative position of the ball using an Additive HGP-EKF without additional parameters (red) and the corresponding GT (blue).

Figures 21 to 29 show the estimated trajectories of the relative position of the ball compared to the GT. In all of these figures, both the GT and the estimations start in an arbitrary position, (100, 0), and after receiving the first observation they follow a trajectory. The first observation for the estimations arrives later than the first observation for the GT (because the laser can see the robot from the very beginning of the experiment while the ball is visible for the robot only in some fractions of the experiment). That is why the estimated trajectories jump to a different point, in the upper part of the plots, from the one the GT trajectory jumps to, which is around (-40, -80).

In figure 21, it is possible to see how a poor model can bring very catastrophic results when using a regular EKF. In this case, there is a clear magnification in the estimated trajectory, probably due to the existence of a similar phenomenon in the observational model.

Table 5 shows the mean square error of the estimation with respect to the ground truth. The error is calculated as the Euclidian distance between the estimated and the GT positions.

Table 5.Mean square error in the estimation of the ball's relative position for the simulated experiment.

Method	Squared error (m^2)	
	With Additional Model Parameters	Without Additional Model Parameters
EKF	-	15.9671
GP-EKF	0.6976	1.5717
GP-EKF + Model	0.5892	0.5638
HGP-EKF	0.5382	0.5100
HGP-EKF + Model	0.5599	0.5578

The first salient fact detectable in table 5, which was already noticeable in figure 21, is the poor performance of a regular EKF. In addition, it is possible to note that HGP-EKFs outperform GP-EKFs for all the cases being considered. The use of additional parameters does not appear to be a clear improvement. The additional parameters produce a noticeable improvement for GP-EKF, yet in the other three cases it causes a decay in the performance of the methods. Finally, the use of a GP or HGP to learn the errors of the theoretical model appears to be of help for GP-EKFs, but in the case of HGP-EKFs it is better to consider the full regression.

5.2.3.5 Experiment on the real robot

The experiment described in subsection 5.2.3.1 was carried out on a real NAO robot using the experimental setup outlined in subsection 5.2.1. The intended trajectory is shown in figure 30.

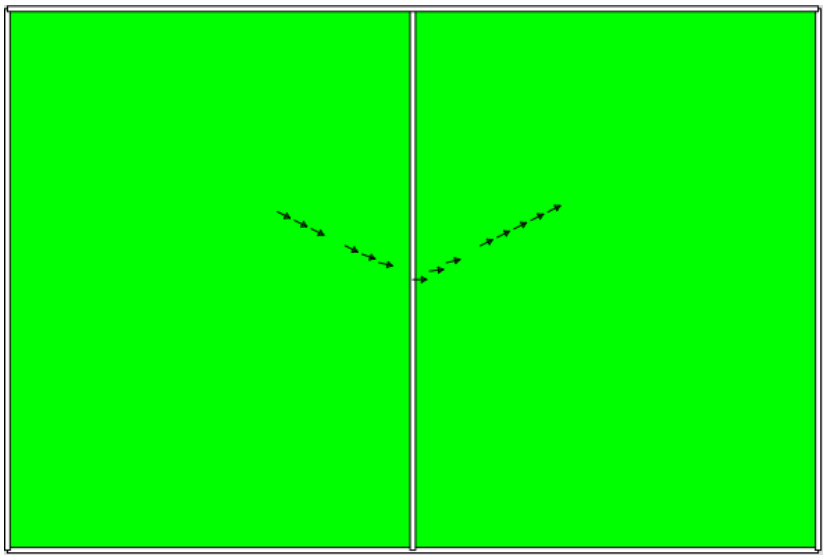


Figure 30. Intended robot trajectory for the HGP-EKF experiment in the real robot.

Figure 31 plots the training samples for the observational model that the real robot obtained (see subsection 5.2.3.1 for details). By comparing figures 20 and 31, it can be easily noted that in the real experiment the observational model mean is much more accurate. It is also possible to note from figure 31 that the variance of the error increases as the ball is farther away.

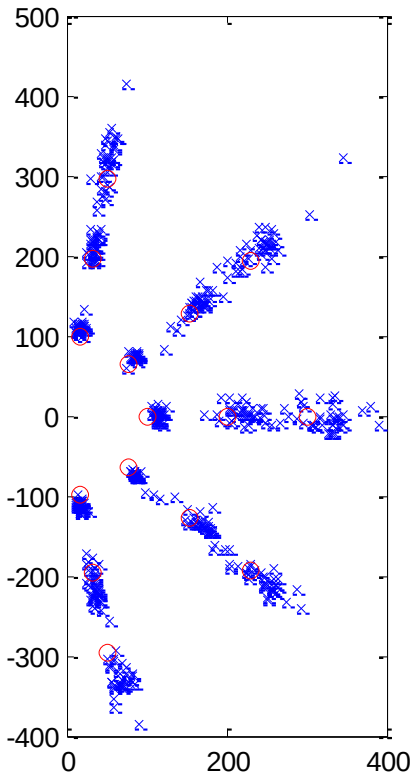


Figure 31. Training samples for the observational and process models. The red circles correspond to the actual position of the ball while the blue crosses show the perceived positions of the ball.

The additional model parameters for the observational model are selected as a set of variables that should be defined with two known possible sources of error in the determination of the pose of the detected ball. These two sources of error are related to two different parts of this process: (i) the estimation, from the detected shape of the object in the image of its position relative to the camera, and (ii) the transformation of the position from the camera system to the robot system, which uses the measurements from the encoders. Figure 32 shows a short visual summary of the process of estimating the ball position relative to the camera. The original image acquired by the camera (figure 32.a) is segmented using previously trained pixel color classes, thereby generating a segmented image (figure 32.b). Then, the orange blob is detected and, from its center, several scan lines are used to detect border points (figure 32.c). A RANSAC-based method is used for estimating the parameters of a circumference from the border points.

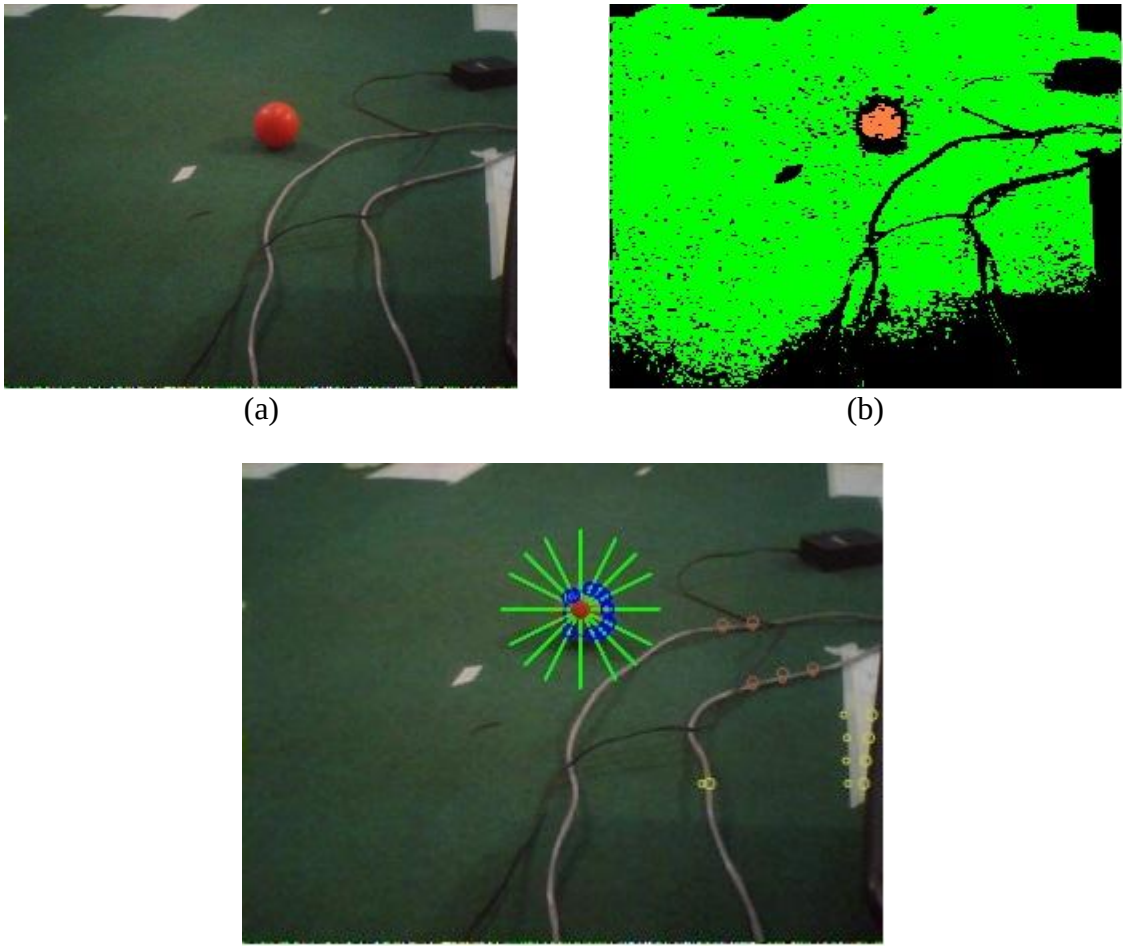


Figure 32. Visual summary of the estimation of the position of the ball relative to the camera. (a) Original acquired image. (b) Segmented Image. (c) Scan lines (light green radial lines) and detected border points (blue circles)

The first subset of additional parameters is intended to detect situations in which the circumference is not well-formed (and consequently more likely to have errors). For each border point, an error vector $\mathbf{e}_i^{bp} = \left(e_{x,i}^{bp}, e_{y,i}^{bp}\right)$ is calculated with respect to the theoretical position of that border point given the circumference parameters and the angle of the scan line that generated the point. Then, the covariance matrix of the error vectors is calculated, and its relevant values, $\left(\mathrm{var}\left(e_{x,i}^{bp}\right), \mathrm{var}\left(e_{y,i}^{bp}\right), \mathrm{cov}\left(e_{x,i}^{bp}, e_{y,i}^{bp}\right)\right)^T$, are the first three selected additional parameters for the observational model.

The second subset of additional parameters for the observational model is designed to detect situations in which the transformation from the camera-relative to the robot-relative

coordinate system can be noisy. One of the sources of noise for this transformation is the lack of synchronization between the encoder measurements and the images. This lack of synchronization may introduce relevant noise when the coordinate-system transformation mentioned above is changing quickly. We now delineate the method for quantifying how the current speed of change of the camera-robot coordinate-system transformation may affect the estimation of the ball position. A different transformation, obtained from the 30ms-old encoders measurements, is used to estimate a different relative-to-the-robot position $\mathbf{b}' = \left(b'_x, b'_y\right)$ of the ball from the same relative-to-the-camera one. Let us denote the normally estimated relative-to-the-robot position of the ball $\mathbf{b} = \left(b_x, b_y\right)$. Then, the two resulting positions are subtracted, $\mathbf{e^b} = \left(e_x^{\mathbf{b}}, e_y^{\mathbf{b}}\right) = \mathbf{b} - \mathbf{b}'$ and the products $\left(e_x^{\mathbf{b}}e_x^{\mathbf{b}}, e_x^{\mathbf{b}}e_y^{\mathbf{b}}, e_y^{\mathbf{b}}e_y^{\mathbf{b}}\right)^T$ are used as the second subset of additional parameters. To summarize, the vector of additional parameters for the observational model is: $\left(\mathrm{var}\left(e_{x,i}^{bp}\right), \mathrm{var}\left(e_{y,i}^{bp}\right), \mathrm{cov}\left(e_{x,i}^{bp}, e_{y,i}^{bp}\right), e_x^{\mathbf{b}}e_x^{\mathbf{b}}, e_x^{\mathbf{b}}e_y^{\mathbf{b}}, e_y^{\mathbf{b}}e_y^{\mathbf{b}}\right)^T$.

The additional parameters for the process model are, in general, more related to direct measurements of sensors in the robot. The first additional parameter for the process model is the elapsed time from the last odometry. Then, for each of the leg motors (11 in the NAO robot) the following are added: the electric current, encoder measurement in the last odometry instant, and encoder measurement in the current odometry instant (33 variables). Finally, the measurements of the 8 pressure sensors (4 per foot) in the robot's feet are also added as additional parameters for the process model. In total, 42 additional parameters are considered for the process model.

Figures 33 to 41 show the estimated trajectories of the relative position of the ball compared to the GT.

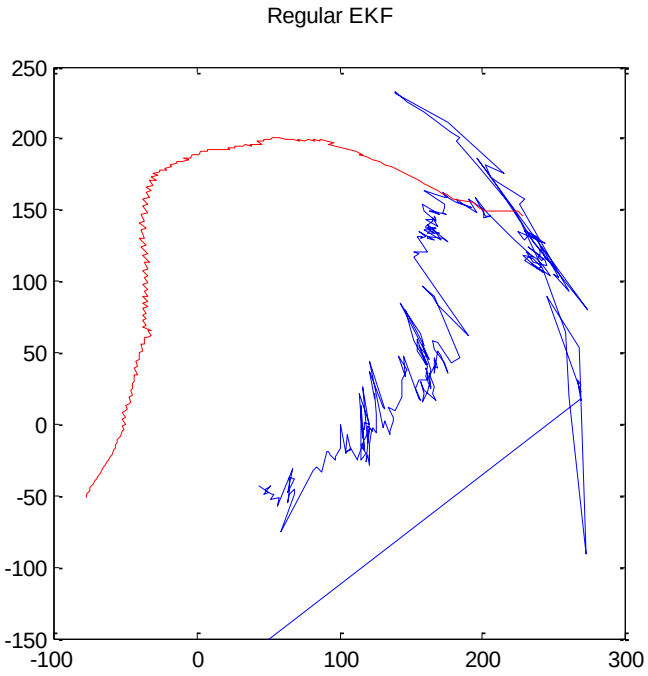


Figure 33. Estimated trajectory of the relative position of the ball using a Regular EKF (red) and the corresponding GT (blue).

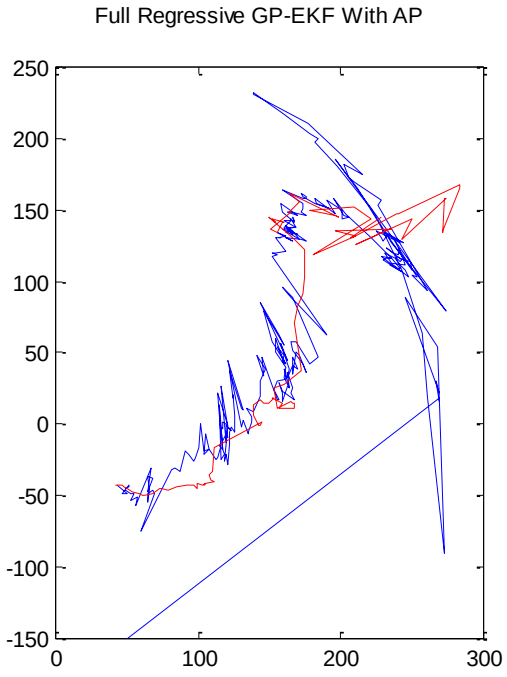


Figure 34. Estimated trajectory of the relative position of the ball using a Full Regressive GP-EKF with additional parameters (red) and the corresponding GT (blue).

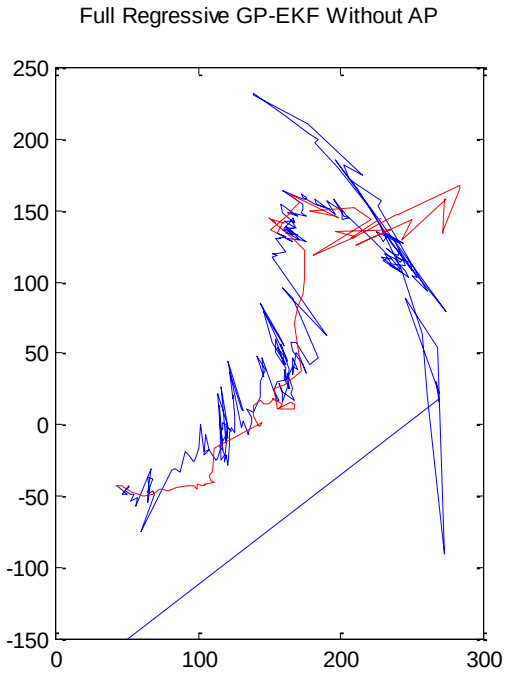


Figure 35. Estimated trajectory of the relative position of the ball using a Full Regressive GP-EKF without additional parameters (red) and the corresponding GT (blue).

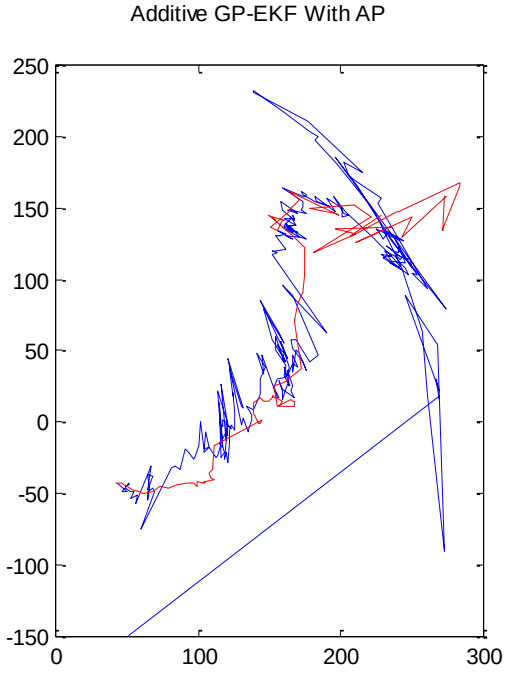


Figure 36. Estimated trajectory of the relative position of the ball using an Additive GP-EKF with additional parameters (red) and the corresponding GT (blue).

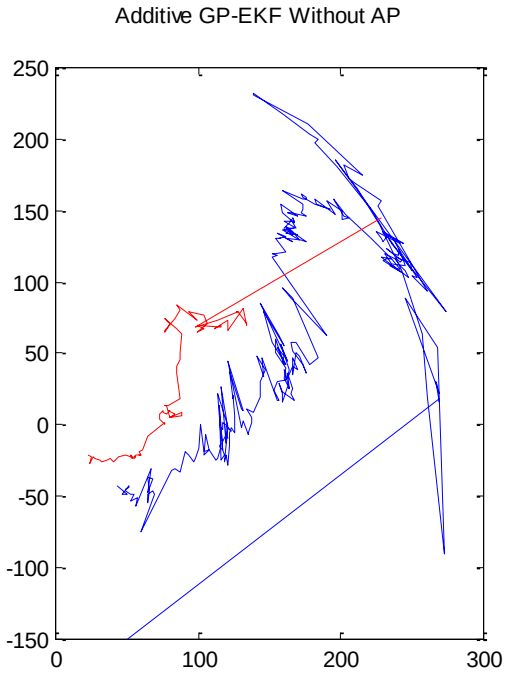


Figure 37. Estimated trajectory of the relative position of the ball using an Additive GP-EKF without additional parameters (red) and the corresponding GT (blue).

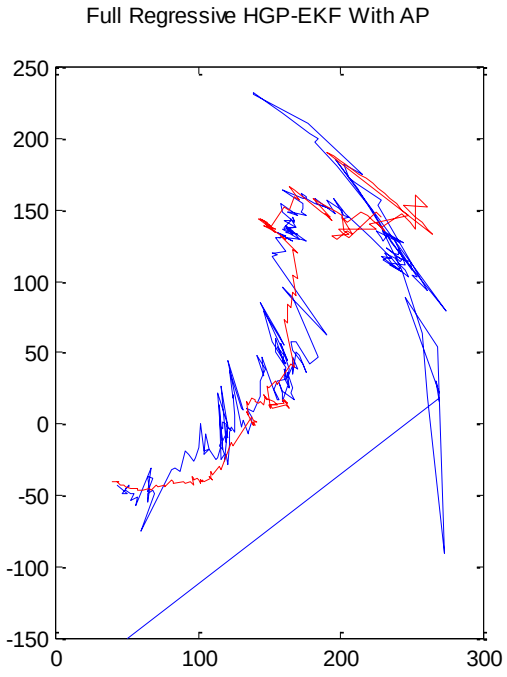


Figure 38. Estimated trajectory of the relative position of the ball using a Full Regressive HGP-EKF with additional parameters (red) and the corresponding GT (blue).

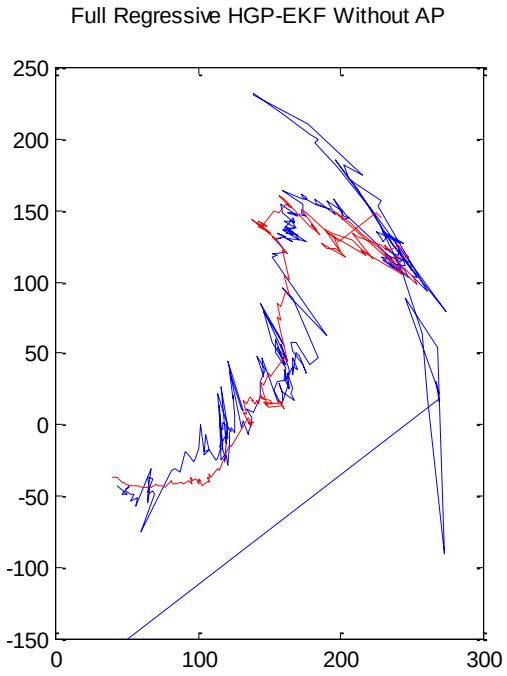


Figure 39. Estimated trajectory of the relative position of the ball using a Full Regressive HGP-EKF without additional parameters (red) and the corresponding GT (blue).

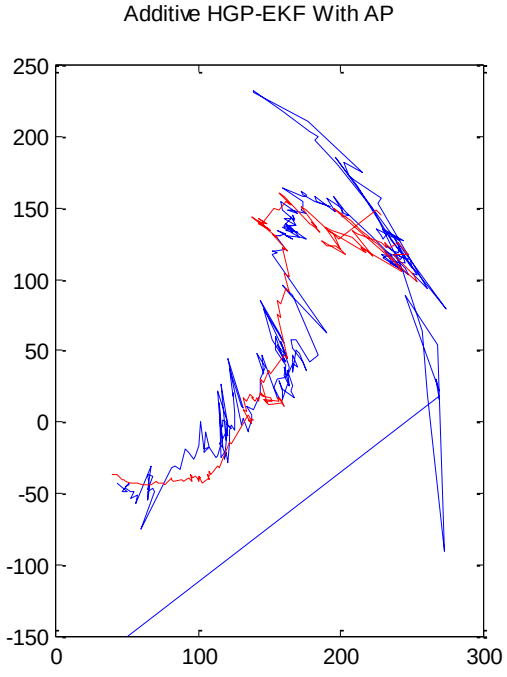


Figure 40. Estimated trajectory of the relative position of the ball using an Additive HGP-EKF with additional parameters (red) and the corresponding GT (blue).

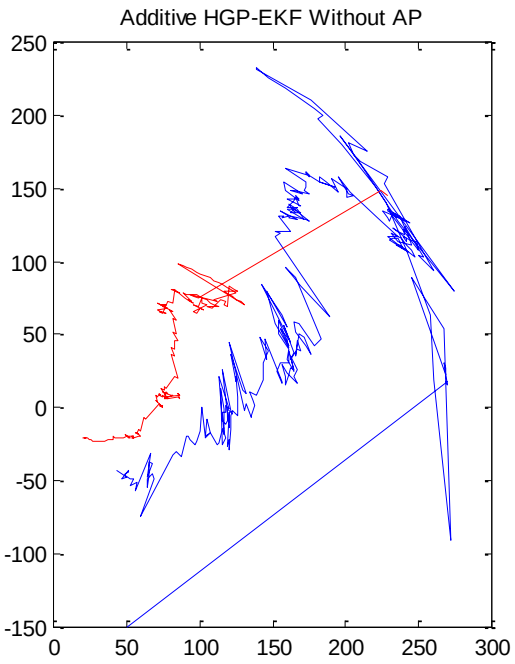


Figure 41. Estimated trajectory of the relative position of the ball using an Additive HGP-EKF without additional parameters (red) and the corresponding GT (blue).

Table 6 shows the mean square error of the estimation of the ball relative position with respect to the ground truth.

Table 6. . Mean square error in the estimation of the ball relative position for the real experiment.

Method	Mean square error (m^2)	
	With Additional Model Parameters	Without Additional Model Parameters
EKF	-	2.7008
GP-EKF	0.2117	0.2117
GP-EKF + Model	0.2117	0.9357
HGP-EKF	0.1647	0.1720
HGP-EKF + Model	0.1720	1.0314

From figures 33 to 41 and table 6, it is possible to see that both GP-EKF and HGP-EKF methods outperform the regular GP. This result is consistent with what was observed in the simulated experiment. Additionally, the performance of the HGP-EKF methods surpassed that of the GP-EKF methods excluding the ones that use an additive theoretical model and without model additional parameters. The best results are obtained by the HGP-EKF

method, with full regression and additional model parameters. In general, the use of the selected additional model parameters improves the results obtained by the methods. For instance, for the method with the best performance, HGP-EKF, the use of additional parameters make the MSE decrease by about 80cm^2.

5.3 Active Vision

In this section, we will show the applicability of the active-vision method under discussion by testing it in the goal-covering task.

5.3.1 Description of the Experiments

To show the effect of the selected active-vision method, we only allow the robot to observe one object at a time, which is directly selected from $\mathbf{u}_k^{sen}$. Additionally, we only allow the robot to change $\mathbf{u}_k^{sen}$ after 300ms since the last selection.

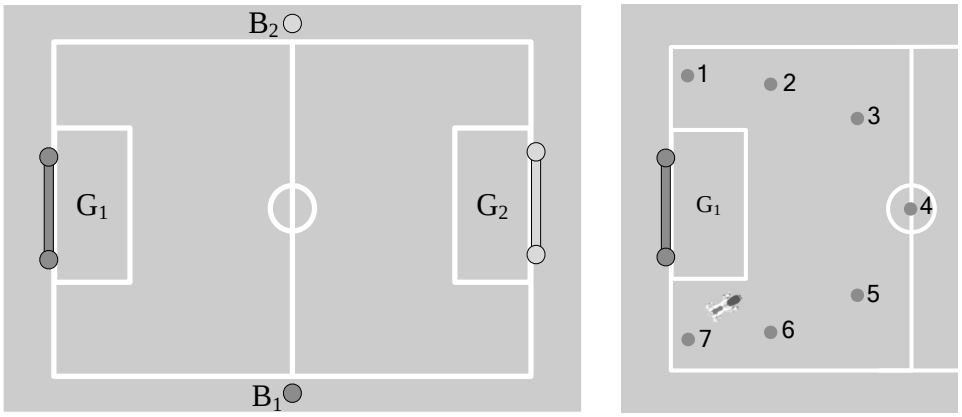
The experiments described in this section have been carried out in a high-level simulator, which simulates the world-modeling and decision-making processes and takes into account process and observation noises, as well as observation probabilities. Only the goalie and the ball are inside the field. The goalie executes the goal-covering task for about 3 minutes, while the ball rolls cyclically (10 repetitions) among seven fixed positions. The robot has no knowledge of the trajectory that the ball will follow. Figure 42.a shows the layout of the field and its fixed objects. Figure 42.b shows the initial position of the goalie and the intermediate positions of the ball.

For the purpose of analysis, we consider each of the 10 cycles of the ball movement as one separate realization of the experiment. For each experiment we have calculated the mean $\bar{V}$ of the task value function $V(\mathbf{x}_k)$ (unknown for the robot) evaluated in the actual state.

The performance measures are the average and standard deviation of $\bar{V}$ over the 10 experiments. With the aim of making the results being presented easier to interpret, we have added two artificial, extreme situations that should serve as a lower and an upper boundary for the active vision performance: (i) a situation in which the robot has very high uncertainty and then it is not even able to cover any portion of the goal (Lost), and (ii) a situation, which is only possible in simulations, without uncertainty (Certain), i.e., in which

the robot has a perfect estimate of its state and thus, executed the described policy with high accuracy. The reader should note that this upper bound of performance is very idealistic and cannot be achieved by any active-vision system with the sensors that currently exist.

In order to compare the computational resources consumed by each of the active-vision methods, we have measured the mean time required by each of them to select $\mathbf{u}_k^{sen}$. These processing times may have considerable variations depending upon the computer on which they are executed. Therefore, we pay more attention to the ratios among them than to their absolute values.



(a) (b)

Figure 42. Experiment performed: (a) layout of the field and its fixed objects: own goal (G_1), opposite goal (G_2), beacon1 (B_1), and beacon 2 (B_2), (b) initial position of the goalie and cyclical positions (1 to 7) of the ball.

While the robot moves, odometry noise introduces errors in the estimations of both the self-localization and the relative position of the ball. If the robot only focuses on one object, either its self-localization or the estimation of the ball location will diverge because it will not receive new observations. Therefore, any reasonable policy for selecting $\mathbf{u}_k^{sen}$ should alternate between observing the ball and observing the landmarks. In addition, because different landmarks can help to reduce different components of the self-localization error, it is also expected that the robot alternate between the observations of different landmarks. In

theory, all of the compared policies are able to alternate between the ball and different landmarks.

5.3.2 Comparison of Methods and Approximation Schemes

The first experiment aims to compare the performance of the active-vision methods being presented, and their approximation alternatives, for the goal-covering task. The methods to be compared are: (i) mutual information (MI), (ii) minimum expected task value variance (VV), and maximum expected action task value (EAV). As is common practice in the active-vision literature [8][28], these methods will be compared to (iv) a random strategy (Ran). The MI method does not need to be approximated[14], but VV and EAV does, so we have tested different combinations of approximation schemes for the state pdf approximation and the observation pdf approximation: (i) mean state, mean observation (ExEz), (ii) SVD for the state, mean observation (Ez), (iii) mean state, SVD for the observation (Ex), and (iv) SVD for the state, SVD for the observation (Full). For the case of the EAV, since we only use the mean of $b_k(\cdot)$ for the policy evaluation, using the mean observation does not allow us to discriminate between the different alternatives for $\mathbf{u}_k^{sen}$. So, we only consider the Ex and Full sampling schemes for EAV. Both VV and EAV have a third sampling step (for the calculation of the task value variance in VV and of the mean in EAV), which is performed using SVD in all cases. In subsection 5.3 we will analyze different sampling strategies. Table 7 shows a comparison of performance among the methods being examined and among their respective approximation alternatives.

As can be seen in table 7, MI performs worse than the random strategy. This result may seem contradictory to those obtained in the literature in which MI always performs better than a random strategy. Thus, how does this experiment differ from all those previously presented? The answer is very simple: in this case, the performance measure is not the reduction of entropy itself. In fact, MI achieves the lowest average entropy. Thus, these

[14] Given that our beliefs are Gaussian, we do not need to approximate the information theory-based methods, since the entropy of a multivariate Gaussian is analytically defined.

results are actually coherent with those of the literature. On the other hand, TOOC based methods perform better than the random strategy. EAV performs better than VV for each of the approximation schemes they have in common. Regarding the approximation schemes, the better the pdf's are approximated, the better the performance. This improvement in performance comes at the cost of higher computational resources consumption. For our application requirements, an interesting tradeoff between performance and computational cost is achieved by EAV-Ex. From this application, the performance of TOOC-based methods is approximately in the middle between a random strategy and the execution of the task without uncertainty. Note that *Certain* operates under conditions which are unreachable for any active-vision system.

Table 7. Performance comparison among the presented methods and approximation schemes for the goal-covering task.

Method	Approx. Scheme	Average $\overline{V}$	STD of $\overline{V}$	Average Belief Entropy	Active Vision Processing Time (ms)
Lost	-	0.33	-	-	-
Ran	-	0.56	0.04	9.56	0.03
MI	-	0.48	0.05	8.21	4.02
VV	ExEz	0.58	0.07	9.46	17.92
	Ez	0.59	0.06	8.96	144.83
	Ex	0.61	0.06	8.97	55.34
	Full	0.65	0.10	8.29	1062.49
EAV	Ex	0.65	0.06	10.12	65.69
	Full	0.66	0.05	8.78	660.52
Certain	-	0.79	-	-	-

Table 8 shows the percentage of the time that each method and approximation scheme looked at each of the observable objects. From the analysis of table 8, the first noticeable fact is that the random behavior has very different time percentages between different objects. This is due to the fact that the active vision system is only able to select between the observable objects. The ball, for example is always observable given the policy explained before.

On the other hand, the own goal is almost never observable. The opposite goal is observable most of the time, while each beacon is observable in a lower percentage of the time. A very evident difference between the MI method and all the TOOC based ones is

that MI looks a small percentage of the time (12%) to the ball, while TOOC methods look a high percentage of the time (35% to 63%) to the ball.

Table 8. Percentage of time that the robot looked at each object using each of the presented methods and approximation schemes.

Method	Approx. Scheme	Ball	Own Goal	Opposite Goal	Beacon 1	Beacon 2
Ran	-	36%	1%	26%	18%	19%
MI	-	12%	5%	55%	14%	14%
VV	ExEz	40%	2%	23%	17%	18%
	Ez	38%	1%	25%	16%	20%
	Ex	47%	0%	21%	16%	16%
	Full	35%	0%	34%	15%	17%
EAV	Ex	63%	0%	8%	16%	13%
	Full	44%	0%	17%	21%	17%

5.3.3 Comparison of Sampling Schemes

The objective of the second experiment is to compare the different sampling schemes that have been presented. For that purpose, we tested all of the presented sampling schemes (excepting mean) using the EAV-Ex method. We selected the EAV-Ex method because it is the best of the methods in terms of consuming an affordable processor time for our application (considering that we select $\mathbf{u}_k^{sen}$ every 300ms and other modules need to share the same computational resources). The two sampling steps of the EAV-Ex method, namely the calculation of $\left\{\left(\boldsymbol{\chi}_{j,i}^{\mathbf{z}_k}, \omega_{j,i}^{\mathbf{z}_k}\right)\right\}$ and $\left\{\left(\boldsymbol{\chi}_{m,j,i}^{\mathbf{x}_{k+1}}, \omega_{m,j,i}^{\mathbf{x}_{k+1}}\right)\right\}$, were performed using different sampling schemes. For the random sampling scheme, we considered three alternatives for the number of samples which correspond to approximately the following: {0.5, 1, 2} times the number of samples resulting in the sigma-point schemes.

As can be seen in Table 9, all the sampling schemes presented have a similar performance (~0.65), except for UT which performs worse (0.57). It is interesting to note that the random strategy performs close to SVD, having the same number of samples. They also have very similar computational costs. If the number of random samples is duplicated in both sampling stages, the performance does not improve noticeably, but of course the computational cost is higher.

Table 9. Comparison of performance for the goal-covering task among the sampling schemes presented.

Sampling Scheme	Observation Samples Number	Next Belief Samples Number	Average $\overline{V}$	STD of $\overline{V}$	Active Vision Processing Time (ms)
Lost	-	-	0.33	-	-
Random (.5)	3	4	0.63	0.05	35.29
Random (1)	5	7	0.65	0.04	60.37
Random (2)	10	14	0.65	0.04	155.96
UT	5	7	0.57	0.08	55.98
SVD	5	7	0.65	0.06	65.69
Certain	-	-	0.79	-	-

On the other hand, if the number of particles is reduced by approximately one half in each sampling stage, then the computational cost is noticeably reduced, with a consequent decline in performance.

Table 10 shows the percentage of the time that each method and approximation scheme looked at each of the observable objects.

Table 10. Percentage of time that the robot looked at each object using each of the tested sampling schemes.

Sampling Scheme	Ball	Own Goal	Opposite Goal	Beacon 1	Beacon 2
Random (.5)	32%	0%	24%	22%	22%
Random (1)	34%	0%	23%	20%	22%
Random (2)	34%	1%	21%	23%	21%
UT	36%	3%	24%	17%	20%
SVD	63%	0%	8%	16%	13%

Chapter 6. Conclusions

The present chapter outlines the conclusions that the author draws from the methods and results that have been presented. The conclusions are divided into three categories: those relative to the model-learning and uncertainty-characterization system, those related to the task-oriented active-vision system, and general conclusions.

6.1 Model Learning and Uncertainty Characterization

We have presented a state estimation method, HGP-EKF, which is able to learn the models and noises from data. In contrast with previous methods that are similar, this method is able to use the covariance of the noise that is expressed on the training data in the estimation process. HGP-EKFs have shown to perform better than GP-EKFs in the simulated and real experiments performed.

The use of additional parameters has shown a clear advantage in some of the tested cases, especially in the real experiment. The cases in which the additional model parameters where not helpful seem clearly due to the incorrect selection of the additional parameters. The additional complexity that the additional parameters, by augmenting the input space, impose to the regression problem could explain the cases in which the additional parameters worsen the performance of the methods. Most likely, the result of the use of additional parameters is strongly dependent on how much information they offer about the correspondent noise parameters. Consequently, the careful selection of these additional parameters could be the key to making them useful.

The method presented for making regression of functions with angle outputs, Angle-GP, has shown to have a better performance than a regular regression. This is relevant in mobile robotics where the estimation of the orientation of objects, as well as the robot itself, are crucial for the understanding of the world.

SERV-HGP has proven to be a valid alternative to ML-HGP when working with multiple dimensions. Additionally, SERV-HGP has showed to require a lower training time than

ML-HGP for the presented one-dimensional problem. However, SERV-HGP has not demonstrated a superior performance in a one-dimensional problem in comparison with ML-HGP.

Regarding the hypotheses, the results show that it is worthwhile to make the statistics of the process and observational noises a function of the current estimated state, action and observation. Furthermore, the results confirm that it is beneficial for the state estimation process to collect and consider some additional parameters in the perception and actuation processes.

In terms of future work, there are several possibilities to be explored. In relation to Angle-GPs, other possible ways to take into account the correlation between the sine and the cosine could be tested. For SERV-HGP it would be useful to explore other methodologies for learning the covariance and the mean of the output together. Additionally, it would be interesting to explore the possible use of the crossed terms of the learned covariance matrix in the inference of the output mean. Regarding HGP-EKFs, it seems to be valuable to test different HGP methodologies from SERV-HGP. In particular, a more straightforward generalization of ML-HGP (based on the matrix exponential and matrix logarithm functions) could be tested. In addition, it would be valuable to test the HGP proposed methodology to other GP-based Bayesian filters such as GP-PF, GP-UKF [89][90] and GP-ADF [91]. Regarding the use of additional model parameters, it could be interesting to develop an automated procedure to select, using the training data, which subset of them would be advisable to use.

6.2 Active Vision

A probabilistic and task-oriented active vision system that is able to select which object the robot should focus on has been discussed. The system is applicable in a wide variety of tasks. A particular robot soccer application, the goal-covering task, is selected to demonstrate its applicability.

Based on the results of the experiments conducted, it is possible to conclude that minimizing the belief entropy is not guaranteed to be a useful optimality criterion when the task is not the reduction of uncertainty itself. On the one hand, the author believes that the poor performance showed by MI for this application is caused by the fact that the entropy-

based methods are not able to discriminate which components of the uncertainty are more relevant for the execution of the task, given the current belief. In other words, they try to reduce the uncertainty blindly. On the other hand, TOOC methods explicitly intend to reduce the uncertainty in the relevant components from the task execution point of view and this explains their superior performance. For example, as might be intuitive, to correctly execute the goal-covering task, looking at the ball as well as at the landmarks must be reasonably balanced in order to simultaneously track the ball and self-localize. Regarding the sampling strategies, for this application, there is no noticeable difference in performance between using SVD and the random strategy with any of the tested sample numbers, but UT showed a lower performance.

Regarding the hypotheses, the results show that it is possible and beneficial for an active-vision system to explicitly consider the task that the robot is performing in order to direct the reduction of uncertainty to the most relevant components.

A potential improvement for the proposed system that could be interesting would be to make the implementation of the methods that showed a better performance more efficient. One possible option for achieving this purpose would be to use an augmented state that includes the state and noises in one single vector, and to make a single sampling step from this vector instead of making two or three nested sampling steps.

Regarding future work, it might be worthwhile to enable the system to consider the possibility of focusing on more than one object with the same sensing action, for example, when the field of view of the camera is able to show more than one object of interest. In that case, the sensing action space should be the gaze direction. It could be also convenient to apply a similar reasoning structure inside the decision-making module. Finally, further research on the use of different POMDP approximation assumptions and solution methods should be carried out in the future.

6.3 General Conclusions

The present thesis addresses the problems of reducing and characterizing the uncertainty in a mobile robot. In relation to the characterization of the uncertainty in the models, a state estimation method that is able to learn the models and noises from data has been presented.

For the reduction of the uncertainty in the state, a probabilistic and task-oriented active vision system has been detailed.

It has been demonstrated that the correct handling of uncertainty can improve the performance of a robot both in terms of the estimation of the state of its environment and on the task being executed.

Bibliography

[1] C. Rothkopf, D. Ballard and M. Hayhoe. Task and context determine where you look. *Journal of Vision*, **7**(14) (2007) 1–20.

[2] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa and H. Matsubara. RoboCup: "A challenge problem for AI". *The AI Magazine*, **18**(1) (1997) 73–85.

[3] Y. Aloimonos. Purposive and Qualitative Active Vision. In *Proc. IAPR Int'l Conf. Pattern Recognition (ICPR '90)*, (Atlantic City, USA, 1990) vol. 1, pp. 346–360.

[4] R. Bajcsy. Active Perception. *Proc. IEEE*, **76**(8) (1988) 996–1005.

[5] D. Ballard. Animate Vision. *Artificial Intelligence*, **48**(1) (1991) 57–86.

[6] J. Denzler, C. Brown and H. Niemann. Optimal Camera Parameter Selection for State Estimation with Applications in Object Recognition. In *Pattern Recognition: 23rd DAGM Symposium*, (Munich, Germany, 2001), pp. 305–312.

[7] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne and J. D. Schutter. Active Sensing for Robotics - A Survey. In *Proc. of the 5th International Conference on Numerical Methods and Applications*, (Borovets, Bulgaria, 2002) pp. 316–324.

[8] T. Arbel and F.P. Ferrie. Entropy-based gaze planning. *Image and Vision Computing*, **19**(11) (2001) 779–786.

[9] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots, *Robotics and Autonomous Systems*. **25** (1998) 195–207.

[10] J. M. Porta, B. Terwijn and B. Kröse. Efficient Entropy-Based Action Selection for Appearance-Based Robot Localization. In *IEEE Int. Conf. Robotics and Automation (ICRA)* (Taipei , Taiwan, 2003) pp. 2842–2847.

[11] S. Czarnetzki, S. Kerner, M. Kruse. Real-time Active Vision by Entropy Minimization Applied to Localization. *Proc. of the RoboCup 2010 Symposium*, (Singapore, 2010). In press.

[12] T. Fukase, M. Yokoi, Y. Kobayashi, R. Ueda, H. Yuasa and T. Arai. Quadruped Robot Navigation Considering the Observational Cost. *Lecture Notes in Computer Science: RoboCup 2001, Robot Soccer World Cup V*, pp. 350–355, 2002.

[13] F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky and H. Durrant-Whyte. Information Based Adaptive Robotic Exploration. *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS (2002)*, pp. 540–545, 2002.

[14] N. Mitsunaga and M. Asada. Visual Attention Control by Sensor Space Segmentation for a Small Quadruped Robot Based on Information Criterion. *Lecture Notes in Computer Science: RoboCup 2001, Robot Soccer World Cup V,* (2002) pp. 154–163.

[15] P. Guerrero, J. Ruiz-del-Solar and M. Romero. Explicitly Task Oriented Probabilistic Active Vision for a Mobile Robot. *Lecture Notes in Computer Science: RoboCup 2008, Robot Soccer World Cup XII*, (2009) pp. 85–96.

[16] P. Guerrero, J. Ruiz-del-Solar, M. Romero and S. Angulo. Task-Oriented Probabilistic Active Vision. *Int. Journal of Humanoid Robotics*, **7**(3) (2010) 451–476.

[17] R. Palma, P. Guerrero, P. Vallejos and J. Ruiz-del-Solar. Context-dependent color segmentation for Aibo robots. *3rd IEEE Latin American Robotics Symposium – LARS 2006*, (Santiago, Chile, 2006), (CD Proceedings).

[18] P. Guerrero, J. Ruiz-del-Solar, J. Fredes, and R. Palma-Amestoy. Automatic On-Line Color Segmentation using Classes-Relative Color Spaces. *Lecture Notes in Computer Science. RoboCup 2007: Robot Soccer World Cup XI*. (2007) pp. 246 – 253.

[19] P. Guerrero, J. Ruiz-del-Solar, and R. Palma-Amestoy. Spatiotemporal Context in Robot Vision: Detection of Static Objects in the RoboCup Four Legged League. *Proc. 1st Int. Workshop on Robot Vision, in 2nd Int. Conf. on Computer Vision Theory and Appl. – VISAPP 2007*, (2007) pp. 136–148.

[20] J. Ruiz-del-Solar, P. Guerrero, R. Palma-Amestoy. Context Integration in Humanoid Robot Vision. *Proc. Cognitive Humanoid Vision Workshop, in Humanoids 2008*, (Daejeon, Korea, 2008) (CD Proceedings).

[21] R. Palma-Amestoy, P. Guerrero, J. Ruiz-del-Solar and C. Garretón. Bayesian Spatiotemporal Context Integration Sources in Robot Vision Systems. *Lecture Notes in Computer Science: RoboCup 2008, Robot Soccer World Cup XII*, (2009) pp. 212–224.

[22] P. Guerrero and J. Ruiz-del-Solar. Improving Robot Self-Localization using Landmarks' Poses Tracking and Odometry Error Compensation. *Lecture Notes in Computer Science, RoboCup 2007, Robot Soccer World Cup XI*, (2008) pp. 148–158.

[23] P. Guerrero, J. Ruiz-del-Solar and G. Díaz. Probabilistic Decision Making in Robot Soccer. *Lecture Notes in Computer Science, RoboCup 2007, Robot Soccer World Cup XI*, (2008) pp. 29–40.

[24] P. Guerrero, J. Ruiz-del-Solar, M. Romero, L. Herrera. An Integrated Multi-Agent Decision Making Framework for Robot Soccer. *6th IEEE Latin American Robotics Symposium – LARS 2009*, (Valparaíso, Chile, 2009) (CD Proceedings).

[25] J. Ruiz-del-Solar, P. Guerrero, M. Arenas, P. Loncomilla, J. Fredes, G. Díaz, R. Palma, P. Vallejos, M. Valenzuela, R. Dodds and D. Monasterio. UChile Kiltros 2007 Team Description Paper. *RoboCup 2007 Symposium*, (Atlanta, USA, 2007) (CD Proceedings).

[26] R. Dodds, L. Iocchi, P. Guerrero, and J. Ruiz-del-Solar. Benchmarks for Robotic Soccer Vision. In RoboCup 2011: Robot Soccer World Cup XV. To appear.

[27] R. Marchant, P. Guerrero, and J. Ruiz del Solar. A portable ground-truth system based on a laser sensor. In RoboCup 2011: Robot Soccer World Cup XV. To appear.

[28] J. Denzler and C. M. Brown. Information Theoretic Sensor Data Selection for Active Object Recognition and State Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24** (2002) 145–157.

[29] A. Seekircher, T. Laue and T. Rofer. Entropy-based Active Vision for a Humanoid Soccer Robot. *Proc. of the RoboCup 2010 Symposium*, (Singapore, 2010). In press.

[30] D. Stronger and P. Stone. Selective Visual Attention for Object Detection on a Legged Robot. *Lecture Notes in Computer Science: RoboCup 2006, Robot Soccer World Cup X* (2007) 158–170.

[31] G. De Croon, I. Sprinkhuyzen-Kuyper and E Postma. Comparing Active Vision Models. *MICC-IKAT Technical Report :06-02* (2006).

[32] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[33] P. Boyle and M. Frean. Dependent Gaussian processes. In *Advances in Neural Information Processing Systems* 17 (2005) 217-224.

[34] P. Goldberg, C. Williams and C. Bishop. Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems 10*. The MIT Press, Cambridge, MA, 1998.

[35] M. Yuan and G. Wahba. Doubly penalized likelihood estimator in heteroscedastic regression. *Statistics & Probability Letters*. **69**(1) pp 11–20. 2004.

[36] K. Kersting, C. Plagemann, P. Pfaff and W. Burgard. Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th international Conference on Machine Learning* (Corvalis, Oregon, June, 2007) pp. 393–400. 2007.

[37] Q. Le, A. Smola and S. Canu. Heteroscedastic Gaussian process regression. In *Proceedings of the 22nd international Conference on Machine Learning* (Bonn, Germany, August, 2005). pp. 489–496. 2005.

[38] A. Wilson and Z. Ghahramani. Copula Processes. *Advances in Neural Information Processing Systems* **23**. 2460–2468. 2010.

[39] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, (MIT Press, Cambridge, MA, A Bradford Book, 1998).

[40] J. Ruiz-del-Solar, J. Zagal, P. Guerrero, P. Vallejos, C. Middleton and X. Olivares. UChile1 Team Description Paper. *Proc. of the RoboCup 2003 Symposium*, (Padova, Italy, 2003).

[41] P. Vallejos, J. Ruiz-del-Solar and A. Duvost. Cooperative Strategy using Dynamic Role Assignment and Potential Fields Path Planning. *1st IEEE Latin American Robotics Symposium – LARS 2004*, (México City, México , 2004) pp. 48 – 53.

[42] W. Press, S. Teukolsky, W. Vetterling and B. Flannery. *Numerical Recipes in C*, 2nd edn, (Cambridge University Press, 1992).

[43] S. Julier and. J. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. *Proc. of 11th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, (Orlando, USA, 1997) pp. 182–193.

[44] L. Kaelbling, M. Littman and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, **101**(1-2), pp. 99–134.

[45] C. H. Papadimitriou and J. N. Tsitsiklis. The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, **12**(3), 1987, pp. 441–450.

[46] M. Littman, A. R. Cassandra and L. Kaelbling. Learning policies for partially observable environments: Scaling up. *Proc. of the 12th International Conference on Machine Learning*, (Morgan Kaufmann, San Francisco, USA, 1995) pp. 362–370.

[47] H. Kurniawati, D. Hsu and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Proc. Robotics: Science and Systems IV*, (Zurich, Switzerland, 2008) pp. 65–72.

[48] Z. Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. On line: http://soma.crl.mcmaster.ca/#zhechen/homepage.htm. 2003.

[49] R. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME—Journal of Basic Engineering*, **82**(Series D), 35-45. 1960.

[50] B. Anderson and J. Moore. *Optimal Filtering*. Prentice-Hall. 1979.

[51] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques and Software*. Artech House. 1993.

[52] T. Lefebvre, H. Bruyninckx, and J. De Schutter. Kalman filters for nonlinear systems: a comparison of performance. Internal Report 01R033, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium. 2001.

[53] R. van der Merwe. Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models. PhD thesis, OGI School of Science & Engineering at Oregon Health & Science University, Portland, OR, April 2004.

[54] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. *In Proc. AeroSense*, 1997.

[55] T.S. Schei. A finite-difference method for linearization in nonlinear estimation algorithms. *Automatica*, **33**(11) pp. 2053–2058. 1997.

[56] M. Nørgaard, N.K. Poulsen, and O. Ravn. Advances in derivative-free state estimation for nonlinear systems. Technical Report IMM-REP-1998-15 (revised edition), Technical University of Denmark, Denmark. 2000.

[57] R. van der Merwe and E. Wan. The square-root unscented kalman filter for state and parameter estimation. *Proceedings of the International Conference on Accoustics, speech, and Signal Processing (ICASSP)*. 2001.

[58] D. Fox, W. Burgard and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, **11** 391–427. 1999.

[59] R. Bellman. *Dynamic Programming*. Princeton University Press. 1957.

[60] N. Gordon, D. Salmond and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Proc. Inst. Elect. Eng. F*. **140**(2) 107–113. 1993.

[61] M. Isard and A. Blake. Condensation—Conditional density propagation for visual tracking. *Int. J. Comput. Vision*. **29**(1) 5–28. 1998.

[62] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. 4th European Conf. Computer Vision*. pp. 343–356. 1996.

[63] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state-space models. *J. Comput. Graph. Statist.* **5** 1–25. 1996.

[64] G. Kitagawa. Self-organising state space model. *J. Amer. Statist. Assoc*. **93** 1203–1215. 1998.

[65] G. Kitagawa and W. Gersch. *Smoothness Priors Analysis of Time Series*. *Lecture Notes in Statistics*. **116**, New York: Springer-Verlag. 1996.

[66] E. Bølviken, P. Acklam, N. Christophersen and J-M. Størdal. Monte Carlo filters for nonlinear state estimation. *Automatica*. **37** 177–183. 2001.

[67] P. Müller. Monte Carlo integration in general dynamic models. *Contemp. Math*. **115** 145–163. 1991.

[68] J. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *J. Amer. Statist. Assoc.* **93**(443) 1032–1044. 1998.

[69] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Proc. Inst. Elect. Eng. F*. **140**(2) 107–113. 1993.

[70] C. Berzuini, N. Best, W. Gilks and C. Larizza. Dynamic conditional independence models and Markov chain Monte Carlo methods. *J. Amer. Statist. Assoc.* **92** 1403–1412. 1997.

[71] M. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *J. Amer. Statist. Assoc.* **94**(446) 590–599. 1999.

[72] A. Doucet, N. de Freitas, K. Murphy and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proc. UAI2000*, pp. 176–183. 2000.

[73] R. van der Merwe, J. F. G. de Freitas, A. Doucet, and E. Wan. The unscented particle filter. In *Adv. Neural Inform. Process. Syst*. **13**, Cambridge, MA: MIT Press. 2001.

[74] J. Yu, D. Xiao and X. Yang. Square Root Unscented Particle Filter with Application to Angle-Only Tracking. *Intelligent Control and Automation*. **1** 1548-1552. 2006.

[75] D. Alspach and H. Sorenson. Nonlinear Bayesian estimation using gaussian sum approximation. *IEEE Trans. Automat. Contr.* **20** 439–447. 1972.

[76] H. Sorenson and D. Alspach. Recursive Bayesian estimation using Gaussian sums. *Automatica*. **7** 465–479. 1971.

[77] J.-S. Gutmann and D. Fox. An experimental comparison of localization methods continued. *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*. 2002.

[78] J.-S. Gutmann, T. Weigel and B. Nebel. Fast, Accurate, and Robust Self-Localization in Polygonal Environments. In *Proc. of the 1999 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* 1412—1419. 1999.

[79] P. Goel, S. Roumeliotis and G. Sukhatme. Robust Localization using Relative and Absolute Position Estimates. In *Proc. of the 1999 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* 1134—1140. 1999

[80] F. Chenavier and J. Crowley. Position estimation for a mobile robot using vision and odometry. In: *Proceedings of the IEEE Int. Conference on Robotics and Automation (ICRA92)*. 2588—2593. 1992.

[81] D. Fox, S. Thrun, W. Burgard and F. Dellaert. Particle filters for mobile robot localization. In *Sequential Monte Carlo Methods in Practice*, pages 499–516. Springer Verlag. 2001.

[82] S. Thrun, D. Fox, W. Burgard and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence.* **128** 99–141. 2001.

[83] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proc. IEEE International Conference on Robotics and Automation (ICRA-2000)*, (San Francisco, CA. 2000).

[84] H. Köse and H. L. Akin. Robots from Nowhere. *Lecture Notes in Computer Science. RoboCup 2004: Robot Soccer World Cup VIII*, 594–601.

[85] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, Springer. 1990.

[86] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI-2002*.

[87] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI).* 2003.

[88] C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In *RoboCup 2004: Robot Soccer World Cup VIII*. pp. 18–33. 2004.

[89] J. Ko, D. Klein, D. Fox and D. Hähnel. GP-UKF: Unscented Kalman filters with Gaussian process prediction and observation models. In *Proc. of the IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), (San Diego, USA, 2007).

[90] J. Ko and D. Fox. GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models. *Autonomous Robots* **27**(1) 75–90. 2009.

[91] M. Deisenroth, M. Huber and U. Hanebeck. Analytic moment-based Gaussian process filtering. *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*, (Montreal, Canada, 2009).

[92] R. Bellman. A Markov decision process. *Journal of Mathematical Mech.* **6** 679–684. 1957.

[93] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*. **13** 33–94. 2000.

[94] E. Sondik. The Optimal Control of Partially Observable Markov Processes. PhD thesis, Stanford. 1971.

[95] A. Drake. Observation of a Markov Process through a Noisy Channel. PhD thesis, Massachusetts Institute of Technology. 1962.

[96] K. Aström. Optimal control of Markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.* **10** 174–205. 1965.

[97] L. Kaelbling, M. Littman and A. Moore. Reinforcement learning: A survey. *JAIR*. **4** 1996.

[98] J. Rust. Numerical dynamic programming in economics. In *Handbook of Computational Economics*. 1996. Elsevier.

[99] G. Pauletto, H. Benitez-Silva, S. Brook, G. Hall, G. J. Hitsch and J. Rust. A Comparison of Discrete and Parametric Approximation Methods for Continuous-State Dynamic Programming Problems. *Computing in Economics. Finance* 2000. 2000.

[100] R. Smallwood and E. Sondik (1973). Optimal control of partially observable processes over the finite horizon. *Operations Research* **21**, 1071–1088.

[101] S. Thrun, W. Burgard and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA. 2005.

[102] M. Littman, A. Cassandra and L. Kaelbling. Learning policies for partially observable environments, scaling up. In *Proceedings of the Fifteenth Conference on Machine Learning*, 362–370. 1995.

[103] A. Cassandra, L. Kaelbling and M. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. (Seattle, WA, 1994).

[104] N. Roy, W. Burgard, D. Fox and S. Thrun. Coastal Navigation --- Mobile Robot Navigation with Uncertainty in Dynamic Environments. *IEEE Int. Conf. on Robotics and Automation*. (Detroit, MI. 1999).

[105] S. Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems* **12** 1064–1070. 1999.

[106] C. Watkins. Learning from Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England. 1989.

[107] Online source code and documentation: http://www.boost.org.

[108] Web page of Aldebaran Robotics: http://www.aldebaran-robotics.com

Chapter 7. Appendices

7.1 Appendix 1

In this appendix, we will prove Eq. 90.

From the definition of $V_{k+1}^{E}(\cdot)$:

$$V_{k+1}^{E}\left(\mathbf{u}_{k}^{sen}\right)=\int V_{k+1}\left(\mathbf{x}_{k+1}\right)p\left(\mathbf{x}_{k+1}\left|U_{k},Z_{k-1}\right.\right)d\mathbf{x}_{k+1}\,. \tag{121}$$

Applying the total probabilities law,

$$p\left(\mathbf{x}_{k+1}\left|U_{k},Z_{k-1}\right.\right)=\int p\left(\mathbf{x}_{k+1}\left|\mathbf{x}_{k},U_{k},Z_{k-1}\right.\right)p\left(\mathbf{x}_{k}\left|U_{k},Z_{k-1}\right.\right)d\mathbf{x}_{k}\,. \tag{122}$$

Note that $\mathbf{u}_{k}^{sen}$ does not give additional information about $\mathbf{x}_{k}$ given U_{k-1}, Z_{k-1}, and $\mathbf{u}_{k}^{act}$, then

$$p\left(\mathbf{x}_{k}\left|U_{k},Z_{k-1}\right.\right)=b_{k}^{-}\left(\mathbf{x}_{k}\right). \tag{123}$$

If the total probabilities law is applied to the first probability inside the integral in (122),

$$p\left(\mathbf{x}_{k+1}\left|\mathbf{x}_{k},U_{k},Z_{k-1}\right.\right)=\int p\left(\mathbf{x}_{k+1}\left|\mathbf{x}_{k},U_{k},Z_{k-1},\mathbf{u}_{k+1}^{act}\right.\right)p\left(\mathbf{u}_{k+1}^{act}\left|\mathbf{x}_{k},U_{k},Z_{k-1}\right.\right)d\mathbf{u}_{k+1}^{act}\,. \tag{124}$$

Applying the Markov property of the state,

$$p\left(\mathbf{x}_{k+1}\left|\mathbf{x}_{k},U_{k},Z_{k-1}\right.\right)=\int p\left(\mathbf{x}_{k+1}\left|\mathbf{x}_{k},\mathbf{u}_{k+1}^{act}\right.\right)p\left(\mathbf{u}_{k+1}^{act}\left|\mathbf{x}_{k},U_{k},Z_{k-1}\right.\right)d\mathbf{u}_{k+1}^{act}\,. \tag{125}$$

The total probabilities law may be applied to the last term:

$$p\left(\mathbf{u}_{k+1}^{act}\left|\mathbf{x}_{k},U_{k},Z_{k-1}\right.\right)=\int p\left(\mathbf{u}_{k+1}^{act}\left|\mathbf{x}_{k},U_{k},Z_{k-1},\mathbf{z}_{k}\right.\right)p\left(\mathbf{z}_{k}\left|\mathbf{x}_{k},U_{k},Z_{k-1}\right.\right)d\mathbf{z}_{k}\,. \tag{126}$$

Given the current state $\mathbf{x}_{k}$ and sensing action $\mathbf{u}_{k}^{sen}$, the current observation does not depend on the past observations or actions. Then,

$$p\left(\mathbf{u}_{k+1}^{act}\left|\mathbf{x}_{k},U_{k},Z_{k-1}\right.\right)=\int p\left(\mathbf{u}_{k+1}^{act}\left|\mathbf{x}_{k},U_{k},Z_{k}\right.\right)p\left(\mathbf{z}_{k}\left|\mathbf{x}_{k},\mathbf{u}_{k}^{sen}\right.\right)d\mathbf{z}_{k}\,. \tag{127}$$

The reader should note that $\mathbf{u}_{k+1}^{act}$ is selected as a function of the information that the robot has about the state after perceiving $\mathbf{z}_{k}$, i.e. $b_{k}\left(\cdot\left|\mathbf{z}_{k}\right.\right)$, and not as a function of the state itself,

because it is unknown to the robot. Thus, given U_k and Z_k, $\mathbf{x}_k$ does not influence $\mathbf{u}_{k+1}^{act}$. In a deterministic policy scheme, $\mathbf{u}_{k+1}^{act} = \pi\left(b_k\left(\cdot|\mathbf{z}_k\right)\right)$ and then,

$$p\left(\mathbf{u}_{k+1}^{act}|\mathbf{x}_k, U_k, Z_k\right) = \delta_{\mathbf{u}_{k+1}^{act}}\left(\pi\left(b_k\left(\cdot|\mathbf{z}_k\right)\right)\right). \quad (128)$$

Then,

$$p\left(\mathbf{x}_{k+1}|\mathbf{x}_k, U_k, Z_{k-1}\right) = \int p\left(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_{k+1}^{act}\right)\int \delta_{\mathbf{u}_{k+1}^{act}}\left(\pi\left(b_k\left(\cdot|\mathbf{z}_k\right)\right)\right) p\left(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_k^{sen}\right) d\mathbf{z}_k d\mathbf{u}_{k+1}^{act}. \quad (129)$$

Reordering and applying the integral of a Dirac delta function:

$$p\left(\mathbf{x}_{k+1}|\mathbf{x}_k, U_k, Z_{k-1}\right) = \int p\left(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_k^{sen}\right) p\left(\mathbf{x}_{k+1}|\mathbf{x}_k, \pi\left(b_k\left(\cdot|\mathbf{z}_k\right)\right)\right) d\mathbf{z}_k. \quad (130)$$

Replacing all the terms, and reordering, we get (90):

$$V_{k+1}^{E}\left(\mathbf{u}_k^{sen}\right) = \int b_k^{-}\left(\mathbf{x}_k\right)\int p\left(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_k^{sen}\right)\int V_{k+1}\left(\mathbf{x}_{k+1}\right) p\left(\mathbf{x}_{k+1}|\mathbf{x}_k, \pi\left(b_k\left(\cdot|\mathbf{z}_k\right)\right)\right) d\mathbf{x}_{k+1} d\mathbf{z}_k d\mathbf{x}_k. \quad (131)$$

7.2 Appendix 2

In this appendix, the matrices used for the generation of the noises in the simulated experiment for HGP-EKF are shown.

$$\mathbf{A}_w = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.00184784 & 0.0486883 & -0.00434644 \\ -0.0442525 & -0.0419647 & 0.00181666 \\ 0.0077776 & -0.000575537 & 0.00414025 \\ 0.027588 & 0.0269114 & -0.00128534 \\ -0.000430892 & 0.0313048 & -0.000449276 \\ 0.0223764 & 0.0244104 & 0.0047718 \\ 0.0107611 & 0.0336207 & -0.00003.418 \\ -0.00428381 & 0.025677 & 0.00199791 \\ -0.0470645 & 0.0449157 & 0.000263378 \\ 0.0260617 & 0.0152622 & -0.0000865692 \\ -0.00221957 & -0.0377493 & -0.000761095 \\ -0.0493921 & -0.00583674 & 0.00336598 \\ 0.00993593 & -0.0269428 & 0.000395047 \\ -0.00417493 & 0.00397914 & -0.0000748089 \end{bmatrix}^T$$

$$\mathbf{B}_{w,0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.0444721 & 0.0194856 & -0.00463632 \\ 0.0356193 & -0.0249845 & 0.000177072 \\ -0.0214058 & -0.00680917 & -0.00278333 \\ 0.0253863 & 0.00469748 & -0.000572097 \\ 0.0440163 & 0.0269437 & -0.00416914 \\ -0.0449397 & -0.00481488 & -0.00394815 \\ -0.0328821 & -0.0340242 & -0.00429665 \\ 0.0473356 & -0.0452361 & 0.00171903 \\ -0.0255414 & -0.0140342 & -0.0011094 \\ -0.0148681 & 0.00386613 & 0.00211564 \\ -0.0240986 & 0.0221872 & 0.00363643 \\ 0.0165503 & -0.00158195 & 0.00435242 \\ -0.00756641 & 0.0161994 & -0.000540701 \\ 0.0454028 & -0.0189831 & 0.00446045 \end{bmatrix}^T$$

$$\mathbf{B}_{w,1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.0313266 & -0.0290438 & -0.00172772 \\ 0.0491533 & 0.043403 & 0.00341576 \\ -0.0168825 & 0.0062602 & -0.00339917 \\ -0.0311724 & -0.0275199 & -0.00376151 \\ -0.0420083 & -0.0266309 & 0.000767759 \\ -0.00103483 & -0.0386324 & -0.000784213 \\ 0.00199885 & -0.0375055 & -0.00395031 \\ 0.0238959 & 0.0474252 & -0.00418994 \\ 0.0195637 & -0.012469 & 0.00302019 \\ -0.0279315 & 0.0153012 & -0.000402372 \\ 0.0179332 & 0.0271944 & 0.00076173 \\ 0.0187414 & -0.00322586 & -0.00137916 \\ 0.0315932 & 0.0473346 & -0.00365961 \\ 0.00528621 & -0.0464412 & 0.00213547 \end{bmatrix}^T$$

$$\mathbf{B}_{w,2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.00423931 & 0.0114126 & -0.00294659 \\ -0.0220488 & -0.00635743 & -0.00482055 \\ -0.00442484 & 0.00362668 & 0.000935457 \\ -0.0349688 & -0.0324026 & -0.00328621 \\ 0.0181164 & -0.0334072 & -0.000983078 \\ -0.0374067 & -0.0470365 & 0.0011661 \\ -0.0498343 & 0.0121046 & 0.0016882 \\ 0.0338102 & 0.0249252 & -0.00236337 \\ -0.00535104 & -0.00000446688 & -0.00219711 \\ -0.0383429 & -0.0170944 & -0.00210234 \\ -0.0143075 & -0.0446306 & -0.00283614 \\ -0.0376363 & -0.0245043 & -0.0025956 \\ -0.023638 & 0.00191749 & -0.00248472 \\ -0.012245 & 0.0464851 & 0.00463476 \end{bmatrix}^T$$

$$\mathbf{A}_v = \begin{bmatrix} 1.61481 & 0.130442 \\ –0.32846 & 1.29184 \\ –0.909618 & 2.35494 \\ –2.20392 & –0.744666 \\ –0.180111 & –0.16559 \\ –1.78418 & 0.215429 \\ –0.499008 & –0.607572 \\ –0.125937 & 0.214927 \\ –1.34966 & –0.68862 \\ 1.4446 & –1.79932 \\ –0.262969 & 0.813034 \\ 0.482908 & –1.27815 \\ –1.32286 & 0.211044 \end{bmatrix}^T$$

$$\mathbf{B}_{v,0} = \begin{bmatrix} -0.114214 & -0.0379078 \\ -0.04519 & 0.0859883 \\ -0.0576854 & -0.0862119 \\ -0.187342 & -0.0210084 \\ -0.0832262 & 0.0583743 \\ 0.0972179 & -0.0208783 \\ -0.148711 & -0.0770422 \\ 0.00479711 & 0.0210305 \\ 0.0310628 & 0.00256729 \\ 0.0108711 & 0.0159934 \\ -0.011021 & -0.272097 \\ 0.0315625 & 0.21578 \\ 0.120146 & -0.0582386 \end{bmatrix}^T$$

$$\mathbf{B}_{v,1} = \begin{bmatrix} -0.160435 & 0.00540103 \\ 0.0301122 & -0.186089 \\ 0.0307346 & 0.0207976 \\ 0.0577201 & 0.0903018 \\ 0.0599567 & 0.121497 \\ 0.0571521 & 0.0506207 \\ 0.00812269 & -0.0426301 \\ -0.140168 & 0.0183013 \\ 0.177049 & 0.018278 \\ 0.0726335 & -0.0542498 \\ -0.119118 & -0.035265 \\ 0.0480869 & 0.122924 \\ 0.0913286 & -0.044206 \end{bmatrix}^T$$

$$\mathbf{B}_{v,2} = \begin{bmatrix} 0.0443834 & 0.112257 \\ -0.0791717 & 0.0568388 \\ 0.272815 & 0.0500298 \\ -0.210177 & 0.037802 \\ 0.0269782 & 0.0141681 \\ 0.0768728 & -0.118712 \\ -0.0361155 & -0.0135977 \\ -0.0608904 & -0.134198 \\ 0.0809307 & -0.0688239 \\ 0.158064 & -0.0290176 \\ -0.0328533 & -0.00825043 \\ 0.0562919 & 0.0993173 \\ -0.0118177 & -0.0976889 \end{bmatrix}^T$$

$$\mathbf{A}_{\mathbf{a}}^{f} = \begin{bmatrix} -0.836854 & -2.20798 & -1.09612 \\ 0.925709 & -1.13145 & 0.925093 \\ 0.871673 & 0.853521 & -1.43916 \\ -2.23105 & -0.845014 & -0.400239 \\ -0.157581 & 0.990905 & 0.488483 \\ 0.519206 & 0.400124 & 2.03364 \\ -0.649426 & 0.264659 & 1.17783 \\ 2.3968 & 0.604024 & 0.2267 \\ 0.297616 & -0.58203 & 0.59399 \\ 1.73441 & 1.53493 & 0.419548 \\ -1.14679 & 0.0411374 & -0.166434 \\ -0.0863592 & -1.65465 & 2.1141 \\ -0.535585 & -0.611868 & -1.41254 \end{bmatrix}^T$$

$$\mathbf{B}^{f}_{\mathbf{a},0} = \begin{bmatrix} 0.187117 & -0.348305 & -0.839174 \\ 1.53055 & 2.53379 & -1.03929 \\ 0.332079 & 1.4206 & 0.0844056 \\ -0.0877757 & -0.105239 & -1.97431 \\ 0.659809 & -0.220419 & 0.260235 \\ 0.325304 & -0.253789 & 0.952576 \\ 1.87727 & 0.0694806 & 1.10965 \\ 1.69614 & 0.289689 & 0.602117 \\ 0.190965 & -0.605211 & 1.77972 \\ -0.457514 & 1.577 & 0.105615 \\ 1.75454 & 0.933655 & 0.287295 \\ 0.628562 & 0.211775 & 0.171124 \\ 0.784261 & 0.711426 & -0.298795 \end{bmatrix}^{T}$$

$$\mathbf{B}^{f}_{\mathbf{a},1} = \begin{bmatrix} -0.176774 & -2.32758 & 0.500645 \\ 0.0514889 & 0.647281 & -1.58997 \\ -0.169367 & -1.01762 & -0.959179 \\ 0.286812 & 1.95946 & 0.24624 \\ 1.16512 & 1.76419 & -0.921122 \\ -0.871454 & -0.119492 & -1.65571 \\ -1.68368 & 0.0943341 & -0.302924 \\ 2.78748 & 0.268209 & 0.0836968 \\ 1.75047 & 0.264141 & -0.579505 \\ -0.767572 & -1.13393 & 0.39777 \\ 1.23326 & -0.0546266 & -0.784715 \\ -1.25854 & 0.24317 & 0.498266 \\ -0.375808 & 1.17754 & 0.377823 \end{bmatrix}^{T}$$

$$\mathbf{B}^{f}_{\mathbf{a},2} = \begin{bmatrix} -0.546364 & 0.0741737 & 0.272275 \\ -1.22616 & -1.02021 & -1.3604 \\ -0.295643 & 0.548881 & -0.664477 \\ 1.10391 & 1.42608 & 0.997151 \\ 0.817815 & 1.81922 & 1.20141 \\ 0.00988832 & 0.594701 & -0.00215018 \\ 0.145749 & 0.981545 & 0.461642 \\ 0.104824 & 0.619294 & 0.0275598 \\ -0.953632 & -0.00658556 & 0.680106 \\ 0.91147 & -0.807789 & -0.322516 \\ -1.78737 & 0.15242 & -2.1924 \\ -0.420092 & -0.490136 & 0.426788 \\ -0.218086 & -0.29047 & 1.11785 \end{bmatrix}^{T}$$

$$\mathbf{A}^{h}_{\mathbf{a}} = \begin{bmatrix} -0.107901 & 0.283888 \\ 0.609109 & -1.17299 \\ -1.38344 & -1.49154 \\ 1.25195 & -0.143795 \\ 0.73636 & 1.20127 \\ -0.260766 & -1.03375 \\ -0.702819 & -0.349936 \\ 1.0738 & 0.914147 \\ 0.412485 & 0.137393 \\ 0.745477 & -0.921148 \end{bmatrix}^{T}$$

$$\mathbf{B}^{h}_{\mathbf{a},0} = \begin{bmatrix} 0.0839882 & 0.226872 \\ 0.766422 & -1.78732 \\ -0.614102 & 0.18582 \\ -2.56068 & -0.454141 \\ 1.83043 & 0.258832 \\ 1.09123 & 0.0607288 \\ -0.471509 & -0.376947 \\ -1.21984 & -0.183102 \\ -2.10741 & 0.94889 \\ -0.0167712 & -0.142518 \end{bmatrix}^{T}$$

$$\mathbf{B}^{h}_{\mathbf{a},1} = \begin{bmatrix} 0.680675 & 0.73318 \\ 1.15584 & -1.06202 \\ 0.773375 & -0.699824 \\ -0.879419 & -0.381928 \\ -0.172145 & 1.2991 \\ -0.212243 & 0.205641 \\ 0.629485 & 1.04005 \\ -1.24175 & -0.303126 \\ -0.86476 & 0.0418026 \\ -0.0180546 & 1.15025 \end{bmatrix}^{T}$$

$$\mathbf{B}^{h}_{\mathbf{a},2} = \begin{bmatrix} 0.137624 & -0.376343 \\ 1.69677 & -0.259935 \\ 0.932938 & 1.39553 \\ 1.94762 & -0.95171 \\ 1.70703 & 0.145292 \\ 0.777626 & 0.670528 \\ 0.0867094 & 1.17914 \\ 2.65515 & -0.599345 \\ 0.858964 & 1.76782 \\ -0.13846 & -0.862903 \end{bmatrix}^{T}$$