Technical University of Crete, Greece
School of Electronic and Computer Engineering

# Robust Localization for the RoboCup Standard Platform League

Nikolaos Kargas

Thesis Committee
Associate Professor Michail G. Lagoudakis (ECE)
Professor Minos Garofalakis (ECE)
Assistant Professor Aggelos Bletsas (ECE)

Chania, September 2013

Πολυτεχνειο Κρητης

Σχολη Ηλεκτρονικων Μηχανικων και Μηχανικων Υπολογιστων

# Αξιόπιστος Εντοπισμός Θέσης για το Πρωτάθλημα Standard Platform του RoboCup

Νικόλαος Κάργας

Εξεταστική Επιτροπή

Αναπληρωτής Καθηγητής Μιχαήλ Γ. Λαγουδάκης (ΗΜΜΥ)

Καθηγητής Μίνως Γαροφαλάκης (ΗΜΜΥ)

Επίκουρος Καθηγητής Άγγελος Μπλέτσας (ΗΜΜΥ)

Χανιά, Σεπτέμβριος 2013

# Abstract

RoboCup is an international competition that promotes research in the field of Robotics and Artificial Intelligence and focuses mainly on the game of soccer. In the Standard Platform League (SPL) all teams use identical robots, namely the Aldebaran NAO humanoid robots and focus on software development aiming to develop fully autonomous robots. A key aspect of robot autonomy is the ability of self-localization. Robot localization is the process of determining the pose (position and orientation) of a robot relative to a given map of the environment with known static landmarks. Robots operating in the SPL league compete on a symmetric field of fixed size with same-colored goals and field lines. Localization can be quite challenging, because of the high uncertainty in locomotion, due to hardware failures, as well as in object recognition, due to landmark ambiguities. This thesis addresses the problem of self-localization in SPL relying on extended Kalman filtering. Our localization algorithm uses observations corresponding to field landmarks and odometry information and provides an estimate for the robot's pose in the field. To deal with the problem of data association, a multiple-hypothesis tracking approach has been applied along with a merging procedure to avoid excessive growth of hypotheses. An augmented state with three odometry error parameters is used to cope with odometry errors; the error parameters are estimated dynamically and individually for each robot along with its pose, as part of the filtering process. The proposed localization approach offers more accurate results and is computationally more efficient compared to a previous approach based on particle filters. The output of our localization module is used in our RoboCup team "Kouretes" as input to the module estimating the global game state, which in turn is used for developing game strategies and assigning appropriate roles to each robot in the team.

# Περίληψη

Το RoboCup είναι ένας διεθνής ρομποτικός διαγωνισμός ο οποίος προωθεί την έρευνα στους τομείς της ρομποτικής και της τεχνητής νοημοσύνης και επικεντρώνεται κυρίως στο παιχνίδι του ποδοσφαίρου. Στο πρωτάθλημα Standard Platform League (SPL) όλες οι ομάδες χρησιμοποιούν πανομοιότυπα ρομπότ, τα ανθρωποειδή ρομπότ Aldebaran Nao και επικεντρώνονται στην ανάπτυξη λογισμικού με σκοπό την δημιουργία πλήρως αυτόνομων ρομπότ. Βασική προϋπόθεση για την δημιουργία αυτόνομων ρομπότ είναι η δυνατότητα εντοπισμού θέσης (συντεταγμένες και προσανατολισμός), δηλαδή η διαδικασία καθορισμού της θέσης του στο περιβάλλον βάσει κάποιων γνωστών σταθερών αντικειμένων. Τα ρομπότ στο πρωτάθλημα SPL αγωνίζονται σε ένα συμμετρικό γήπεδο καθορισμένου μεγέθους, το οποίο περιέχει δύο τέρματα ίδιου χρώματος και διαγράμμιση με λευκές γραμμές. Το πρόβλημα του εντοπισμού θέσης είναι αρκετά απαιτητικό λόγω της αβεβαιότητας στην κίνηση (αστοχία υλικού) και στην αναγνώριση αντικειμένων (αμφισημία σταθερών αντικειμένων). Η παρούσα διπλωματική εργασία ασχολείται με το πρόβλημα του αυτο-εντοπισμού θέσης βασιζόμενη σε εκτεταμένα φίλτρα Kalman (extended Kalman filters). Ο αλγόριθμος που υλοποιήθηκε δέχεται ως είσοδο πληροφορίες για τα αντικείμενα που έχουν αναγνωρισθεί (παρατηρήσεις) και πληροφορίες για την μετακίνηση του ρομπότ (οδομετρία) και παρέχει μία εκτίμηση για την θέση του στο γήπεδο. Για την αντιμετώπιση του προβλήματος της συσχέτισης αντικειμένων και παρατηρήσεων έχει προστεθεί η δυνατότητα παρακολούθησης πολλαπλών υποθέσεων (multiple hypotheses tracking) σε συνδυασμό με μια διαδικασία συγχώνευσης για την αποφυγή της υπερβολικής αύξησης των παραγόμενων υποθέσεων. Η εκτιμώμενη κατάσταση επαυξάνεται με τρεις μεταβλητές που μοντελοποιούν τα συστηματικά σφάλματα στο σύστημα οδομετρίας του ρομπότ. Αυτές εκτιμώνται δυναμικά και ατομικά για κάθε ρομπότ παράλληλα με τη θέση του, ως μέρος της διαδικασίας εκτίμησης. Η προτεινόμενη μέθοδος προσφέρει πιο ακριβή αποτελέσματα και έχει καλύτερη υπολογιστική απόδοση σε σύγκριση με μια προηγούμενη μέθοδο που βασιζόταν σε φίλτρα σωματιδίων. Η εκτίμηση του αλγορίθμου μας χρησιμοποιείται στην ομάδα «Κουρήτες» ως είσοδος για την από κοινού εκτίμηση της καθολικής κατάστασης του παιχνιδιού, η οποία με την σειρά της χρησιμοποιείται για την ανάπτυξη στρατηγικών και την ανάθεση ρόλων σε κάθε ρομπότ της ομάδας.

# Acknowledgements

First, I would like to thank my advisor Michail G. Lagoudakis for his trust and guidance during the course of this thesis.

Next, I would like to thank the members of team "Kouretes", namely Lefteris Chatzilaris, Manolis Orfanoudakis, Nikos Kofinas, Nikos Pavlakis, Stelios Piperakis, and Vagelis Michelioudakis, who supported and shared with me some great ideas.

My friends from Chania and Irakleio, Eleni (Bouklou), George (E-Senin), Eleni (Paoki), Katerina (Chan...), Kostas (Bouclas), Marinio , Nikolas, and Billy. Together we had some amazing moments during the last six years.

Last, but not least, I would like to thank my family for their love, support, and constant encouragement.

# Contents


**1 Introduction** **1**

1.1 Thesis Contribution . . . . . 2

1.2 Thesis Outline . . . . . 3

**2 Background** **5**

2.1 RoboCup . . . . . 5

2.1.1 Standard Platform League . . . . . 6

2.1.2 Aldebaran Nao Humanoid Robot . . . . . 7

2.2 RoboCup SPL Team Kouretes . . . . . 10

2.2.1 Monas Software Architecture . . . . . 11

2.3 Mobile Robot Localization . . . . . 14

2.3.1 Robot Pose . . . . . 15

2.3.2 Motion Model . . . . . 15

2.3.3 Sensor Model . . . . . 17

2.4 Pose Estimation . . . . . 17

2.4.1 Bayes Filter . . . . . 17

2.4.2 Particle Filter . . . . . 18

2.4.3 Kalman Filter . . . . . 20

2.4.4 Extended Kalman Filter . . . . . 21

**3 Problem Statement** **25**

3.1 Robot Localization in RoboCup . . . . . 25

3.2 Related Work . . . . . 27

3.2.1 Kalman Filter Approaches . . . . . 27

3.2.2 Particle Filter Approaches . . . . . 29

3.2.3 Constraint Localization Approaches . . . . . 29

3.2.4 Hybrid Approaches . . . . . 29

**4 Our Approach** **31**
4.1 Prediction Step . . . . . 32
4.2 Update Step . . . . . 34
4.3 Multiple Hypothesis Tracking . . . . . 36
4.3.1 Hypothesis Representation . . . . . 37
4.3.2 Initialization . . . . . 38
4.3.3 Incorporating Odometry . . . . . 39
4.3.4 Incorporating Observations . . . . . 40
4.4 Merging Hypotheses . . . . . 42
4.5 Odometry Calibration . . . . . 44
4.6 The Proposed Algorithm . . . . . 47

**5 Results** **51**
5.1 Algorithm Accuracy . . . . . 52
5.2 Algorithm Performance . . . . . 54
5.3 Multi-Hypothesis Tracking Results . . . . . 55
5.4 Odometry Calibration Results . . . . . 56

**6 Conclusion** **59**
6.1 Conclusion . . . . . 59
6.2 Future Work . . . . . 59
6.2.1 Additional Landmarks . . . . . 59
6.2.2 Vision System Calibration . . . . . 60

**References** **65**

# List of Figures


2.1 Standard Platform League at RoboCup 2013 . . . . . 7
2.2 Aldebaran Nao v3.3 (Academic edition) components . . . . . 8
2.3 Embedded and desktop software for the Nao robot . . . . . 9
2.4 The NAOqi process . . . . . 10
2.5 Team Kouretes at RoboCup 2013 in Eindhoven, The Netherlands . . . . 11
2.6 The pose of a mobile robot in a planar environment . . . . . 16

3.1 Specifications of the SPL 2013 field and dimensions in mm . . . . . 26

4.1 Actual robot trace and reported odometry for a straight walk command . 32
4.2 Odometry motion model . . . . . 33
4.3 Observation of landmark at $(lx, ly)$ from robot at pose $(x, y, \theta)$ . . . . . 35
4.4 Hypotheses splitting after two ambiguous observations . . . . . 37
4.5 Belief initialization for one robot in Ready state . . . . . 39
4.6 Predetermined positions for manual placement of both teams in Set state 40
4.7 Initial (left) and resulting (right) hypotheses with an ambiguous goalpost 42
4.8 Hypotheses before (left) and after (right) merging . . . . . 44

5.1 Estimated traces of MHT-EKF and MCL algorithm . . . . . 52
5.2 Sequence of walk commands for comparing MHT-EKF and MCL estimates 53
5.3 Final poses of four representative executions . . . . . 53
5.4 Execution times of MHT-EKF and MCL algorithms during a run . . . . 54
5.5 Estimated robot pose after 8 update steps with a single-hypothesis EKF 55
5.6 Estimated robot pose after 8 update steps with a multi-hypothesis EKF . 56
5.7 Estimated pose traces with and without odometry calibration . . . . . 57
5.8 Evolution of estimated values of odometry error parameters . . . . . 58

## LIST OF FIGURES

# List of Algorithms


1 Bayes Filter Algorithm . . . . . 18
2 Monte-Carlo Localization Algorithm . . . . . 20
3 Kalman Filter Algorithm . . . . . 21
4 Extended Kalman Filter Algorithm . . . . . 22
5 MHT-EKF robot localization algorithm . . . . . 48

# Chapter 1

# Introduction

The RoboCup Competition is an international annual robotics competition which aims to promote Robotics and Artificial Intelligence research. RoboCup Soccer is one of the five RoboCup divisions and focuses mainly on the game of soccer. The research goals involve cooperative multi-robot and multi-agent systems in dynamic adversarial environments. All the participating teams have to find real-time solutions to some of the most challenging problems in robotics, such as perception, cognition, action, and coordination.

The Standard Platform League (SPL) constitutes one of the RoboCup Soccer leagues. The participating teams are restricted to use the same hardware platform, the Aldebaran Nao humanoid robot, and therefore concentrate on software development. They aim to build a fully autonomous team of robots that are able to plan and act without any human interaction in order to achieve their common goal. One of the main problems that needs to be addressed in this context is robot self-localization. Localization is a problem of state estimation, where the robot has to estimate its pose (state) relative to the environment. To accomplish this task, the robot has to exploit information provided by its perceptions and actions.

In RoboCup Soccer competitions it is essential for a robot to know its pose as most of the decisions that has to make depend on it. A simple example would be the correct positioning to support a teammate, defend its own half of the field, or kick the ball into the right direction, i.e. towards the opponent goal. The localization task must be performed on-board, in real time, on the limited robot CPU and robustly, dealing with noisy, and even incorrect, sensing.

## 1.1 Thesis Contribution

This thesis describes a new approach developed for localization in our RoboCup team based on an Extended Kalman Filter (EKF) that replaced our previously Monte-Carlo localization, also known as particle filter localization.

Our localization module integrates odometry and observations information, estimating the pose of the robot at any time. The initial pose of the robot in the field is approximately known, as it is limited to a small number of candidate field areas. Therefore, the robot must disambiguate its hypotheses about these initial poses and track afterwards its pose maintaining a reliable belief using all the available information. At each time step a prediction and an update process are executed; the former estimates the new pose of the robot based on the previous pose and the information about the actions taken in that time step, while the latter refines this estimate using information provided by the observations made in that time step. The prediction step requires a motion model, while the update step requires a sensor model.

The colors of the two goals in the SPL field used to be distinct (cyan and yellow), however, since 2012, it was decided that they will be uniformly colored yellow, leading to a fully symmetric field and thus to ambiguous landmark observations. This ambiguity leads to landmark association issues during the update step. In our approach, multiple filters are used to track different hypotheses dealing with the landmark association problem. To overcome the exponential growth of these filters, the ones with similar estimations are merged based on a Mahalanobis distance metric to ensure that their number does not exceed a predefined threshold.

The locomotion actions are taken into account in the prediction step through open-loop computations within the motion model, known as odometry, which estimate changes in robot pose. The rather inaccurate estimates provided by odometry led us into investigating ways of calibrating robot odometry individually for each robot, so that the robot can predict poses over longer periods of time, especially when observation information is not available. The state to be estimated, apart from robot pose, was augmented with three more state variables related to tunable parameters of the motion model; their values are estimated online in real time and are used within the motion model in the prediction step of the algorithm for accurate odometry computations.

The new localization module yields more accurate results compared to the previous approach and is computationally efficient for on-board execution. In addition, it enables the easier integration of the individual robot beliefs into a global, shared world model, which is used to support the development of team strategies.

## 1.2 Thesis Outline

In Chapter 2 we present all the background information needed for this thesis. We give an overview of the RoboCup Soccer competition, the Aldebaran Nao humanoid robot, team Kouretes, and our software architecture Monas. Furthermore, we provide basic information about localization and filtering. In Chapter 3 we state our robot localization problem related to the RoboCup competition and the SPL and we refer to different approaches that have been used by other teams. In Chapter 4 we describe our robust self-localization method which is based on extended Kalman filtering and augmented states. In Chapter 5 the performance of the proposed approach is evaluated and compared with our previous localization method. Finally, Chapter 6 acts as an epilogue for this thesis, presenting our conclusions along with future improvements.

# Chapter 2

# Background

## 2.1 RoboCup

The RoboCup competition is an international annual robotics competition inspired by Hiroaki Kitano in 1993 [1] and established in 1997. Its goal is to promote the research fields of artificial intelligence, multi-agent systems, and robotics. Participating teams focus on developing fully autonomous agents capable of operating into dynamic environments. The official goal of the project as stated by the RoboCup Federation is as follows:

> *"By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, complying with the official rule of the FIFA, against the winner of the most recent World Cup".*

While this ambitious objective remains open, RoboCup has since expanded into other relevant application domains based on the needs of modern society. Today, RoboCup covers the following themes:

**RoboCup Soccer**: The main focus of the RoboCup competitions is the game of football/soccer, where the research goals concern cooperative multi-robot and multi-agent systems in dynamic adversarial environments. All robots in this league are fully autonomous.

**RoboCup Rescue**: The intention of the RoboCup Rescue division is to promote research and development of highly mobile, dexterous, and fully- or semi- autonomous robots for search and rescue missions.

**RoboCup @Home**: This division aims at designing autonomous and naturally interactive assistant robots that can help people in their daily lives at home and in public.

**RoboCup @Work**: It is a new competition in RoboCup that targets the use of robots in work-related scenarios. It aims to foster research and development that enables use of innovative mobile robots equipped with advanced manipulators for current and future industrial applications.

**RoboCup Junior**: It is designed to introduce RoboCup to primary and secondary school children, as well as undergraduates who do not have the resources to get involved in the senior leagues of RoboCup. The focus of the Junior League lies on education.

### 2.1.1 Standard Platform League

RoboCup Soccer consists of five different leagues (Humanoid, Middle Size, Simulation, Small Size, and Standard Platform). In the Standard Platform League (SPL) all the teams use identical robots, the Aldebaran Nao humanoid robot. The teams are prohibited to make any changes to the hardware of the robot, neither can they interact with the robots during the games, therefore they concentrate on algorithm design and software development aiming at developing fully autonomous robots. The only interaction allowed is among the robots in the field through the wireless network and between the robots and the Game Controller, a computer that broadcasts information about the state of the game (score, time, penalties, etc.).

Currently, the SPL games are conducted on a field with dimensions $6m \times 9m$ (Figure 2.1). The field consists of a green carpet marked with white lines and two yellow goals. The appearance of the field is similar to a real soccer field, but it is scaled to the size of the robots. The ball is an orange street hockey ball. Each team consists of five robots, one goal keeper and four field players. Each robot wears a colored jersey shirt as team marker; jerseys are blue for one team and red for the other. The total game time is 20 minutes and is broken in two halves; each half lasts 10 minutes. There are strict rules about player pushing, ball holding, leaving the field, etc. enforced by human referees; violation of theses rules results in player penalizations. The complete rules of the SPL games are stated in detail in the RoboCup Standard Platform League (Nao) Rule Book [2].

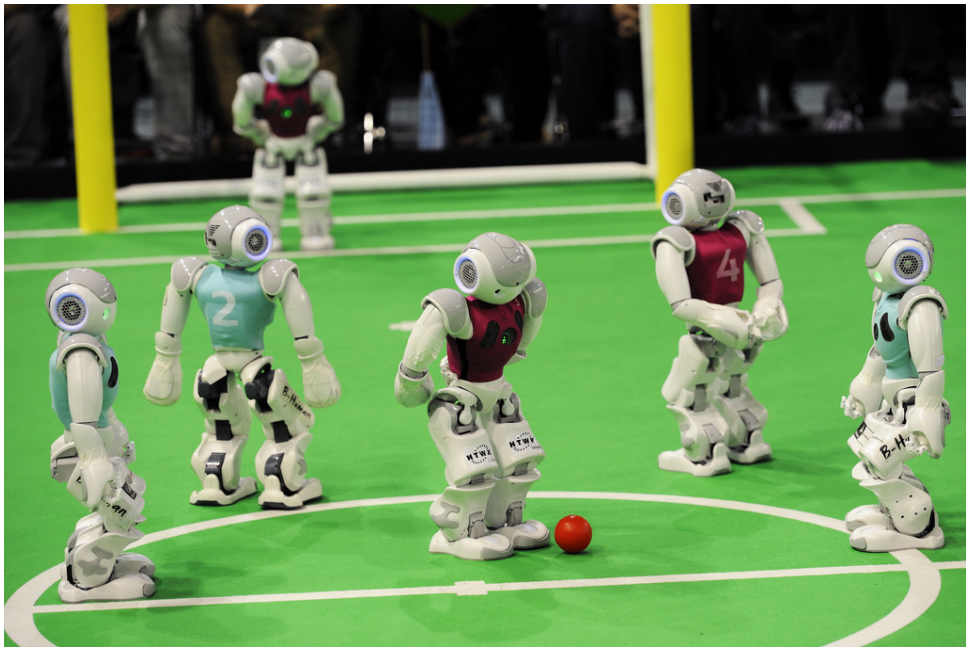


Figure 2.1: Standard Platform League at RoboCup 2013

### 2.1.2 Aldebaran Nao Humanoid Robot

The hardware platform that is currently used for the standard platform league is Nao, an integrated, programmable, medium-sized humanoid robot developed by Aldebaran Robotics in Paris, France. The robot's development began with the launch of Project Nao [3] in 2004. In August 2007, Nao officially replaced Sony's Aibo quadruped robot in the RoboCup SPL. In the past few years Nao has evolved over several designs and several versions.

Nao (version 3.3), shown in Figure 2.2, is a 58cm, 5kg humanoid robot. The Nao robot carries a fully capable computer on-board with an x86 AMD Geode processor at 500 MHz, 256 MB SDRAM, and 2 GB flash memory running an Embedded Linux distribution. It is powered by a 6-cell Lithium-Ion battery which provides about 30 minutes of continuous operation and communicates with remote computers via an IEEE 802.11g wireless or a wired ethernet link.

Nao RoboCup edition has 21 degrees of freedom; 2 in the head, 4 in each arm, 5 in each leg and 1 in the pelvis (there are two pelvis joints which are coupled together

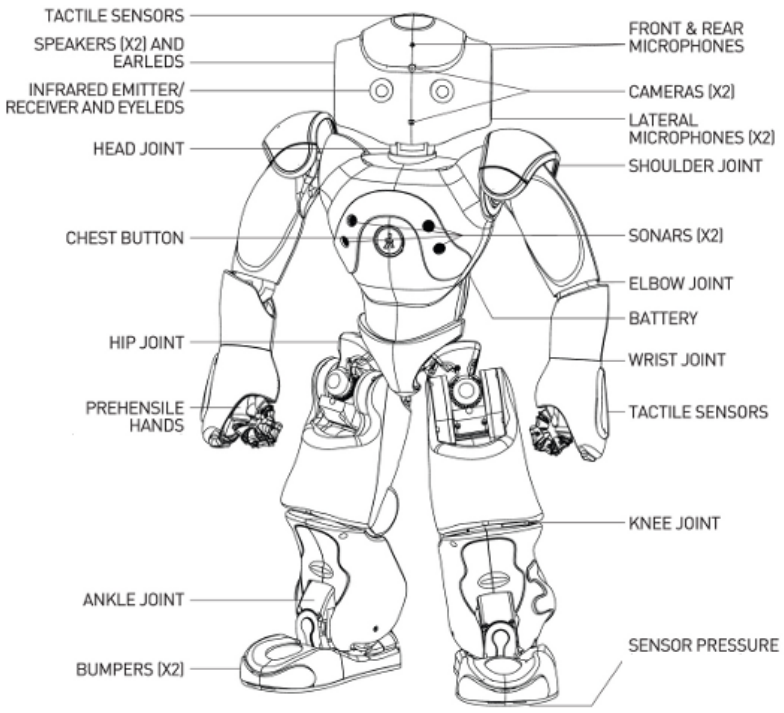



Figure 2.2: Aldebaran Nao v3.3 (Academic edition) components

on one servo and cannot move independently). Nao, also, features a variety of sensors. Two cameras are mounted on the head in vertical alignment providing non-overlapping views of the lower and distant frontal areas, but only one is active each time and the view can be switched from one to the other almost instantaneously. Each camera is a $640 \times 480$ VGA devise operating at 30fps. Four sonars (two emitters and two receivers) on the chest allow Nao to sense obstacles in front of it. In addition, the Nao has a rich inertial unit, with one 2-axis gyroscope and one 3-axis accelerometer, in the torso that provides real-time information about its instantaneous body movements. Two bumpers located at the tip of each foot are simple ON/OFF switches and can provide information on collisions of the feet with obstacles. Finally, an array of force sensitive resistors on each foot delivers feedback of the forces applied to the feet, while encoders on all servos record the actual values of all joints at each time.

Aldebaran Robotics has equipped Nao with both embedded and desktop software [4] to be used as a base for further development (Figure 2.3). The embedded software,

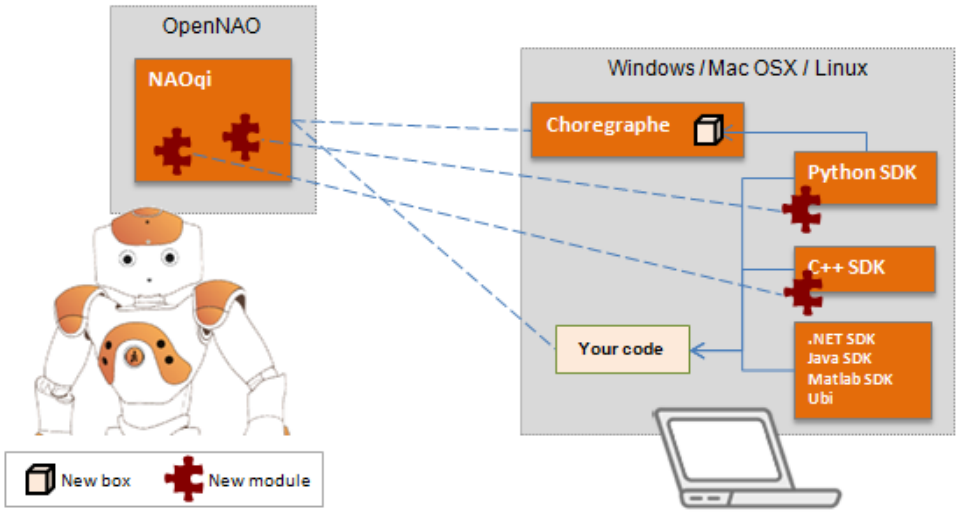



Figure 2.3: Embedded and desktop software for the Nao robot

running on the motherboard located in the head of the robot, includes an embedded GNU/Linux distribution and NAOqi, the main proprietary software that runs on the robot and controls it. Nao's desktop software includes Choregraphe, a visual programming application which allows the creation and the simulation of animations and behaviors for the robot before the final upload to the real Nao, and Telepathe which provides elementary feedback about the robot's hardware and a simple interface to accessing its camera settings.

As far as the NAOqi framework is concerned, it is cross-platform, cross-language, and provides introspection which means that the framework knows which functions are available in the different modules and where. It provides parallelism, resources, synchronization, and events. NAOqi, also, allows homogeneous communication between different modules (motion, audio, video), homogeneous programming, and homogeneous information sharing. Software can be developed in C++, Python, and Urbi. The programmer can state which libraries have to be loaded when NAOqi starts via a preference file called `autoload.ini`. The available libraries contain one or more modules, which are typically classes within the library and each module consists of multiple methods (Figure 2.4).

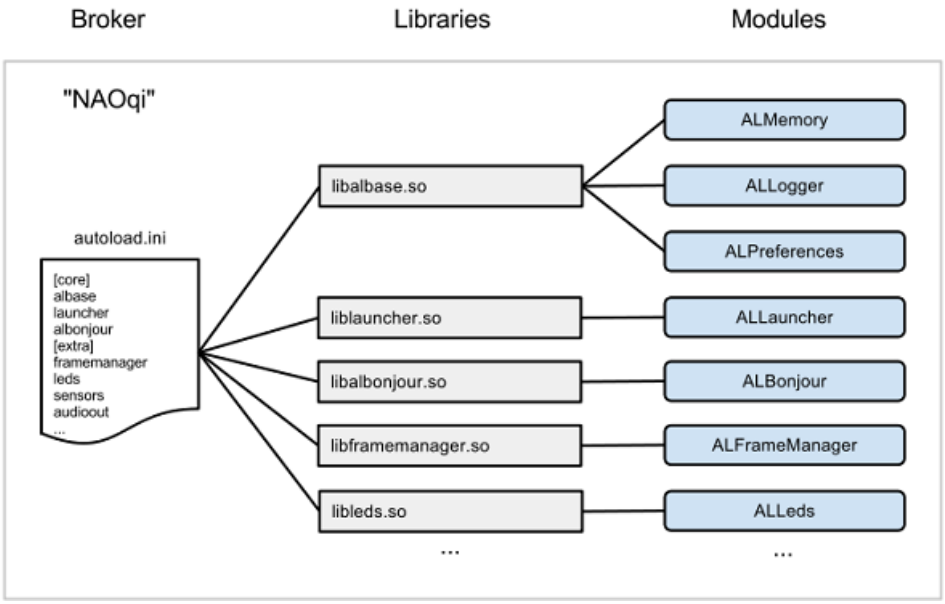



Figure 2.4: The NAOqi process

## 2.2 RoboCup SPL Team Kouretes

Team Kouretes is the RoboCup team of the Technical University of Crete and currently the only RoboCup SPL team founded in Greece. The team was founded in 2006 and participates in the main RoboCup competition ever since in various leagues (Four-Legged, Standard Platform, MSRS, Webots), as well as in various local RoboCup events (German Open, Mediterranean Open, Iran Open, RC4EW, RomeCup) and RoboCup exhibitions (Athens Digital Week, Micropolis, Schoolfest). Distinctions of the team include: 2nd place in MSRS at RoboCup 2007; 3rd place in SPL-Nao, 1st place in SPL-MSRS, among the top 8 teams in SPL-Webots at RoboCup 2008; 1st place in RomeCup 2009; 6th place in SPL-Webots at RoboCup 2009; 2nd place in SPL at RC4EW 2010; and 2nd place in SPL Open Challenge Competition at RoboCup 2011 (joint team Noxious-Kouretes). In the RoboCup 2012 competition, the team succeeded to proceed to the second round-robin round and rank among the top-16 SPL teams in the world. Recently, the team participated in AutCup 2012, in RoboCup Iran Open 2013, and in the RoboCup 2013 competition in Eindhoven (Figure 2.5). The members of the team are senior undergraduate and postgraduate ECE students of the Technical University of Crete working on


Figure 2.5: Team Kouretes at RoboCup 2013 in Eindhoven, The Netherlands

their diploma thesis on RoboCup-related topics.

Kouretes started developing their own robotic software framework in 2008 and the code is constantly growing and gets maintained ever since. The team's publicly-available code repository[1] includes a custom software architecture, a custom communication framework, a graphical application for behavior specification, and modules for object recognition, state estimation, obstacle avoidance, behavior execution, and team coordination, which are briefly described below.

### 2.2.1 Monas Software Architecture

Monas [5] is a flexible software architecture which provides an abstraction layer from the hardware platform and allows the synthesis of complex robot software as XML-specified Monas modules, Provider modules, and/or Statechart modules. Monas modules, the so-called agents, focus on specific functionalities and each one of them is executed independently at any desired frequency completing a series of activities at each execution.

[1] https://github.com/kouretes/Monas

The base activities, that an agent may consist of, are described briefly below:

- `Vision` [6] is a light-weight image processing method for humanoid robots, via which Kouretes team has accomplished visual object recognition. The vision module determines the exact camera position in the 3-dimensional space and subsequently the view horizon and the sampling grid, so that scanning is approximately uniformly projected over the ground (field). The identification of regions of interest on the pixels of the sampling grid follows next utilizing an auto-calibrated color recognition scheme. Finally, detailed analysis of the identified regions of interest seeks potential matches for corresponding target objects. These matches are evaluated and filtered by several heuristics, so that the best match (if any) in terms of color, shape, and size for a target object is finally extracted. Then, the corresponding objects are returned as perceived, along with an estimate of their current distance and bearing. Currently the detectable objects are the ball and the field goals.

- `LocalWorldState` is the end product of this thesis; it is responsible for estimating and providing the local belief of each robot and operates on the outcomes of the `Vision` activity. Details will be provided in Chapters 4 and 5.

- `SharedWorldModel` [7] is the activity which combines the local beliefs of all robots to create a common and shared estimation of the current state of the world (robot poses within the field, location of the ball, etc.) consistent with these local beliefs. The communication between the robots is accomplished through our communication framework, Narukom.

- `PathPlanning` [8] is the activity which accomplishes path planning with obstacle avoidance by first building a local, polar, obstacle occupancy map, which is updated constantly with real-time sonar information, taking into consideration the robot's locomotion. Afterwards, an A* search algorithm is used for path planning, the outcome of which suggests an obstacle-free path for guiding the robot to a desired destination. The way-points of the planned path are finally translated into walk commands to guide the robot along the path.

- `Behavior` is the activity which implements the desired robotic behavior. It operates on the outcomes of the `Vision`, `LocalWorldState`, `ObstacleAvoidance` and

`SharedWorldModel` activities and decides which one is the most appropriate action to be executed next (walk, kick, etc.). Locomotion actions are passed to the `PathPlanning` activity for obstacle-free navigation, while motion actions are sent to the `MotionController` activity for execution. Behavior also includes a separate module which is responsible for determining a team strategy and assigning a role to each robot in the team [9]. This coordination mechanism dynamically selects the most appropriate roles depending on the robots' belief about the global world state, provided by the `SharedWorldModel` activity.

- `HeadController` manages the movements of the robot head (camera). It receives the desired commands from the `Behavior` activity.
- `MotionController` is used for managing and executing robot locomotion commands and special actions.
- `RobotController` handles external signals on the game state.
- `LedHandler` controls the robot LEDs (eyes, ears, chest button, feet).

Provider modules accomplish the complete decoupling of the robotic hardware by collecting and filtering measurements from the robot sensors and cameras and forming them as messages in order to be utilized as input data by any interested Monas agents. Each provider module can be executed independently and at any desired frequency.

Custom Forward and Inverse Kinematics [10, 11], designed specifically for the Nao humanoid robot, have been implemented as an independent software library optimized for speed and efficiency. The library is currently being used in other team projects, such as omni-directional walk engine and dynamic kick engine.

Statechart modules, which offer an alternative intuitive graphical specification of robot behavior, have also been integrated into Monas [12]. Kouretes Statechart Engine (KSE) [13, 14] is our own graphical tool for designing and editing statecharts for robot behavior. Statecharts are automatically transformed into code and are executed on-board using a generic multi-threaded statechart engine, which provides the required concurrency and meets the real-time requirements of the activities on each robot.

Robot communication is accomplished through Narukom [15], the communication framework developed for the needs of the team's code and it is based on the publish/-subscribe messaging pattern. Narukom supports multiple ways of communication, including local communication among the Monas modules, the Providers modules, and the Statechart modules that constitute the robot software, and remote communication via multicast connection among multiple robot nodes and among robot and external computer nodes. The information that needs to be communicated between nodes is formed as messages which are tagged with appropriate topics and host IDs.

KMonitor [16] is a debugging tool created specifically for the Monas architecture that takes advantage of the modularity of Kouretes code and helps in finding errors or verifying that newly implemented features work correctly. It also allows for the easy creation of colortables, the transmission of remote commands over the network, etc.

## 2.3 Mobile Robot Localization

Localization is one of the most fundamental problems in mobile robot navigation. Robot localization is the process of determining the position of a robot relative to a given map of the environment exploiting information based on sensor readings. This information typically comes from two different sources; motion sensors and perception sensors. Motion sensors provide data related to robot displacements (e.g odometer readings), while perception sensors provide information related to the environment (e.g camera images). The localization problem of a mobile robot can be divided into three main categories; **pose tracking**, **global localization**, and **kidnapped robot problem** [17] with an increasing degree of difficulty.

- `Pose Tracking`

  Pose tracking is defined as the problem of localization of a moving robot, when the initial pose of the robot is known. The goal is to compensate incremental errors in odometry and maintain a reliable estimate about its pose. Therefore, external sensors are needed to provide additional information related to the environment.

- `Global Localization`

The problem when the robot has no information about its initial pose and has to place itself to the map is known as global localization or absolute positioning. Initially, the robot must exploit multiple observations from the environment, possibly while moving, to infer its current pose within the map. Once this is achieved, the problem is reduced to pose tracking.

- `Kidnapped robot problem`

  A variant of the global localization problem is the kidnapped robot problem. The robot can be moved at any time, while it operates, and placed to a different location in the map without any notification. It must have the ability to recognize such changes and adapt accordingly.

Due to high uncertainty in motion and perception, often a probabilistic framework is used, thus formulating localization as a Bayesian estimation problem.

### 2.3.1 Robot Pose

The state of a mobile robot in a planar environment at any given time step can be described by a three-dimensional vector and is known as robot's *pose*. The pose consists of the two-dimensional planar coordinates $x$ and $y$ and its heading direction (orientation, bearing) $\theta$ forming the following vector (at time step $k$):

$$\mathbf{x}_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.1}$$

Robot poses, in general, are depicted with a circle, which denotes the coordinates $(x, y)$, and a small line, which denotes the orientation $\theta$, as shown graphically in Figure 2.6. The belief, which reflects the robot's internal knowledge about its pose (state), is represented by a probability distribution (belief) to express the uncertainty related to it denoted as $bel(\mathbf{x}_k)$.

### 2.3.2 Motion Model

A mobile robot needs locomotion mechanisms to be able to move around its environment. There are several mechanisms to accomplish this, including two, four, and six legged

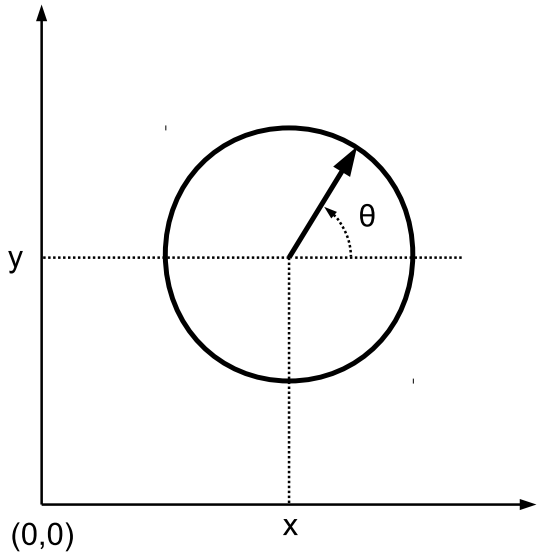



Figure 2.6: The pose of a mobile robot in a planar environment

locomotion and many configurations of wheeled locomotion. The way the robot moves is directly connected to the localization problem, as it contains valuable information for estimating its current pose.

Robots are usually aware of the controls (velocities, torques, currents, etc.) they issue at any time, however their connection to the actual locomotion may be complex. In some cases, this connection can be computed analytically. Alternatively, robots are equipped with internal sensors that measure relative displacements and together with the type of the locomotion mechanism give a strong indication of what kind of locomotion action took place regardless of the actual control. This information about the locomotion action that took place at any time step is available at the end of each time step (after issuing controls) but is uniformly treated as the control input $u$ whether it was computer or measured. Given the current state of the robot and the current control input, the robot makes a transition to a new state. The probabilistic model that describes these transitions is called the *motion model* and is written as

$$P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \tag{2.2}$$

where $\mathbf{x}$ is the state of the robot. This model describes the posterior distribution of the robot pose $\mathbf{x}_k$ at time step $k$ given that at time step $k-1$ the robot was at pose $\mathbf{x}_{k-1}$

and performed the action $\mathbf{u}_k$. Note that this model adopts the Markov property, namely transitions are independent of history given the latest state and control input.

### 2.3.3 Sensor Model

Motion sensors alone are insufficient for reliable localization. The robot must also be equipped with external sensors that can provide additional information about features or landmarks of the environment. These may include range finders, magnetic compasses, cameras, or global positioning systems. Due to sensor uncertainty, the sensing of the robot can also be described in probabilistic terms and is called sensor (or perceptual) model.

$$P(\mathbf{z}_k|\mathbf{x}_k) \tag{2.3}$$

where $x$ is the state of the robot and $\mathbf{z}$ is the perceptual information (observation). The sensor model describes the likelihood of making the observation $\mathbf{z}_k$ at time step $k$ given the pose $\mathbf{x}_k$ at the same time step. Again, this model adopts the Markov property, namely observations are independent of history given the latest state.

## 2.4 Pose Estimation

The general localization problem can be described as a Bayesian estimation problem, where the robot seeks to estimate a posterior distribution over the space of possible poses conditioned on the available data (control inputs and observations). The method that one should apply may differ depending on the environment the robot moves in, the available resources (e.g. computational power, sensors), etc. In this section we will describe the general estimation algorithm (Bayes filter) and three popular instantiations used for mobile robot localization, namely particle, Kalman, and extended Kalman filters.

### 2.4.1 Bayes Filter

Probabilistic localization algorithms are variants of the Bayes filter [18]. The Bayes filter recursively estimates the robot's belief from the previous belief and the latest control input and observations. The key idea is that the whole problem is to estimate a probability

density function over the state $\mathbf{x}_k$ conditioned on data collected up to that time:

$$bel(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}) \tag{2.4}$$

assuming some belief $bel(\mathbf{x}_0)$ about the initial state. The data $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ denote the control inputs (or odometer readings) and $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k$ the observations at the corresponding time steps. Applying Bayes rule, Equation 2.4 can be written as follows:

$$\begin{aligned} bel(\mathbf{x}_k) &= p(\mathbf{x}_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}) \\ &= \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_{1:k}, \mathbf{z}_{1:k-1})p(\mathbf{x}_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k-1})} \\ &= \eta p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_{1:k}, \mathbf{z}_{1:k-1})p(\mathbf{x}_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k-1}) \end{aligned} \tag{2.5}$$

where $\eta$ is a normalization term to ensure that $bel(\mathbf{x}_k)$ integrates to one. Bayes filters assume that the environment is Markov, that is, past and future data are conditionally independent, if one knows the current state. Making use of the Markov assumption, Algorithm 1, which makes use of the motion and sensor models, is derived.

**Algorithm 1** Bayes Filter Algorithm

1: **for** $k = 1, 2, \ldots, n$ **do**
2: $\quad bel(\mathbf{x}_{k|k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) bel(\mathbf{x}_{k-1}) dx$
3: $\quad bel(\mathbf{x}_{k|k}) = p(\mathbf{z}_k|\mathbf{x}_k) bel(\mathbf{x}_{k|k-1})$
4: **end for**
5: **return** $bel(\mathbf{x}_{n|n})$

The algorithm has two essential steps. The first step, called *prediction*, computes the distribution of the current state after integrating the control information $\mathbf{u}_k$ (Line 2). The second step, called *update*, corrects the predicted distribution using the latest observation $\mathbf{z}_k$ (Line 3). To implement the Bayes filter algorithm, one needs to know the probability densities $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ and $p(\mathbf{z}_k|\mathbf{x}_k)$ of the motion and sensor models respectively, mentioned in Sections 2.3.2 and 2.3.3. Instantiations of the Bayes filter are derived by choosing appropriate representations for the beliefs and the two models.

### 2.4.2 Particle Filter

The implementation of the Bayes filter in the continuous state space of mobile robot localization may be challenging, if one cares about efficiency. The particle filter is a non-

parametric implementation of the Bayes filter. The first application of particle filters in robot localization problem was introduced by W. Burgard and S. Thrun in 1999 and became known as Monte Carlo Localization [17]. Monte Carlo localization is one of the most popular algorithms in robotics, as it is able to solve both pose tracking and the global localization problem and in some cases the kidnapped robot problem. The key idea is the representation of the probability density using a set of $N$ weighted samples (particles).

$$\mathbf{X}_k = \{\mathbf{x}_k^1, \mathbf{x}_k^2, \ldots, \mathbf{x}_k^N\} \tag{2.6}$$

Each sample is represented by a tuple $(x, y, \theta, w)$ where $x, y, \theta$ is the robot pose (state) and $p$ a non-negative factor called importance factor (weight) denoting the probability of a specific particle to be correct. The belief of the robot at time step $k$, $bel(\mathbf{x}_{k|k})$ is constructed recursively from the belief $bel(\mathbf{x}_{k-1|k-1})$ one time step earlier and just like the Bayes filter algorithm, Monte-Carlo localization proceeds in two steps. When the robot moves, the $N$ samples are propagated according to the motion model (prediction step) and the resulting set represents the prior belief $bel(\mathbf{x}_{k|k-1})$. When an observation is acquired, the importance factor of each sample is weighted by the sensor model (update step) and the resulting set represents the updated belief $bel(\mathbf{x}_{k|k})$.

The last step of the algorithm implements what is called resampling or sequential importance resampling [19]. The old set of particles is replaced by a new one formed by sampling particles with replacement in accordance to their weights. Only the samples with the highest weights are therefore more likely to be part of the next set. Resampling is essential for keeping particles near the true state. After the resampling step, the weights of all samples are reset to $\frac{1}{N}$. By adjusting the number of samples the desired balance between accuracy and computational cost can be succeeded. The entire Monte Carlo localization algorithm is shown in Algorithm 2.

Once the particles have converged to a single location, it is not possible for the algorithm to recover, if this pose happens to be incorrect. The same problem arises when the robot is moved and placed to another location of the map. An extension which is often applied to the MCL algorithm is the injection of random samples to the particle set at each iteration. Various methods exist for adding samples, most of them relying on

**Algorithm 2** Monte-Carlo Localization Algorithm

1: **for** $i = 1$ to $N$ **do**
2: $\quad \mathbf{x}_k^i = \text{motion\_model}\ (\mathbf{x}_{k-1}^i, \mathbf{u}_k)$
3: $\quad w_k^i = \text{measurement\_model}\ (\mathbf{z}_k^i, \mathbf{x}_k^i)$
4: $\quad \overline{\mathbf{X}}_k = \overline{\mathbf{X}}_k + < \mathbf{x}_k^i, w_k^i >$
5: **end for**
6: **for** $i = 1$ to $N$ **do**
7: $\quad$ draw $i$ with probability $\propto w_k^i$
8: $\quad$ add $\mathbf{x}_k^i$ to $\mathbf{X}_k$
9: **end for**
10: **return** $bel(\mathbf{x}_k)$

the sensor readings [18, 20, 21]. A comparison between different particle filter localization methods, including the above, tested on Sony's Aibo robots has been performed by Gutmann and Fox [22].

### 2.4.3 Kalman Filter

The Kalman filter (KF) [23] provides a solution to the problem of estimating the state of processes described by linear stochastic dynamic models in which the system and measurement noises are white and Gaussian. The Kalman filter model assumes the true state at time $k$ is evolved from the state at $k-1$ according to

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \tag{2.7}$$

Matrix $\mathbf{F}$ is a $n \times n$ matrix that relates the state $\mathbf{x}_{k-1}$ of the previous time step $k-1$ to the state $\mathbf{x}_k$ at the current time step $k$. Matrix $\mathbf{B}$ is the control-input model which is applied to the control vector $\mathbf{u}$ and $\mathbf{w}_k \sim N(0, \mathbf{Q})$ is the zero-mean Gaussian process noise.

The KF assumes that the measurements are linearly related to the state of the system

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{2.8}$$

where $\mathbf{H}$ is the sensor model, which maps the true state space into the observed space and $\mathbf{v}_k \sim N(0, \mathbf{R})$ is the zero-mean Gaussian observation noise.

**Algorithm 3** Kalman Filter Algorithm

**Predict**

$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k$

$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{Q}$

**Update**

$\hat{\mathbf{z}}_k = \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}$

$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}$

$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1}$

$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k (\mathbf{z}_k - \hat{\mathbf{z}}_k)$

$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1}$

**return** $\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}$

The Kalman filter is a recursive estimator and consists of an iterative prediction-correction process, shown in Algorithm 3. Here $\mathbf{x}_{k|k-1}$ denotes the a priori estimate of the robot pose at time step $k$ using all measurements until time step $k-1$ and $\mathbf{P}_{k|k-1}$ the a priori estimate error covariance. Matrix $\mathbf{K}_k$ is the Kalman gain and $\mathbf{S}_k$ is the innovation (or residual) covariance. The resulting probability distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k}, \mathbf{u}_{1:k})$ is a Gaussian distribution $N(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$.

### 2.4.4 Extended Kalman Filter

When either the motion or sensor model is non-linear, the conditional probability density functions are no longer Gaussian. This is true for the systems we study. Hence, Kalman filter cannot be applied directly to the problem of robot localization. The extension to the non-linear case is called the Extended Kalman Filter (EKF) [24] and is based on the linearization of the system around the current state estimate. The success of the EKF depends on how well the system is approximated by the linearization. The non-linear equations describing the robot's motion and sensor models are the following:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{2.9}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{2.10}$$

where $f$ and $h$ are non-linear functions. We assume that the control input $\mathbf{u}_k$ is also corrupted by Gaussian noise $\mathbf{p}_k \sim N(0, \mathbf{M}_k)$.

$$\mathbf{u}_k = \mathbf{u}_k^{true} + \mathbf{p}_k \tag{2.11}$$

**Algorithm 4** Extended Kalman Filter Algorithm

**Predict**
$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$
$\mathbf{P}_{k|k-1} = \mathbf{F}_x \mathbf{P}_{k-1|k-1} \mathbf{F}_x^\top + \mathbf{F}_u \mathbf{M}_k \mathbf{F}_u^\top + \mathbf{Q}$
**Update**
$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_{k|k-1})$
$\mathbf{S}_k = \mathbf{H}_x \mathbf{P}_{k|k-1} \mathbf{H}_x^\top + \mathbf{R}$
$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_x^\top \mathbf{S}_k^{-1}$
$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k)$
$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1}$
**return** $\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}$

The EKF linearizes 2.9 and 2.10 around the current estimate $\hat{\mathbf{x}}_{k-1}$ and the current motion command $\mathbf{u}_k$. Linearizing the state transition equation we get

$$\mathbf{x}_k = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) + \mathbf{F}_x(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{F}_u(\mathbf{u}_k^{true} - \mathbf{u}_k) + \mathbf{w}_k \tag{2.12}$$

where $\mathbf{F}_x$ and $\mathbf{F}_u$ are the Jacobians of the function $f$ with respect to $\mathbf{x}_k$ and $\mathbf{u}_k$ respectively.

$$\mathbf{F}_x = \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}, \mathbf{u}_k} \tag{2.13}$$

$$\mathbf{F}_u = \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\hat{\mathbf{x}}_{k|k-1}, \mathbf{u}_k} \tag{2.14}$$

The measurement equation is linearized following the same procedure

$$\mathbf{z}_k = h(\hat{\mathbf{x}}_{k-1}) + \mathbf{H}_x(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{v}_k \tag{2.15}$$

$$\mathbf{H}_x = \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}} \tag{2.16}$$

The EKF uses the linear Equations 2.12 and 2.15 to estimate $\mathbf{x}_k$. As these equations are linear, one can directly use the Kalman filter equations presented before. The extended Kalman filter algorithm is shown in Algorithm 4.

EKF gives only an approximation of the optimal estimate. Other techniques using Gaussian distributions have been proposed in the literature, such as the Unscented

Kalman Filter (UKF) [25] that addresses some of the approximation issues of the EKF and has been applied by several teams in RoboCup SPL. The UKF transformation uses a set of deterministically chosen weighted samples that parametrize the mean and covariance of the belief. The system function is applied to each sample, which results in a group of transformed points. The mean and covariance of this group of points are the propagated mean and covariance. Since there is no linearization involved in the propagation of the mean and covariance, the Jacobians of the system and measurement model do not have to be calculated.

# Chapter 3

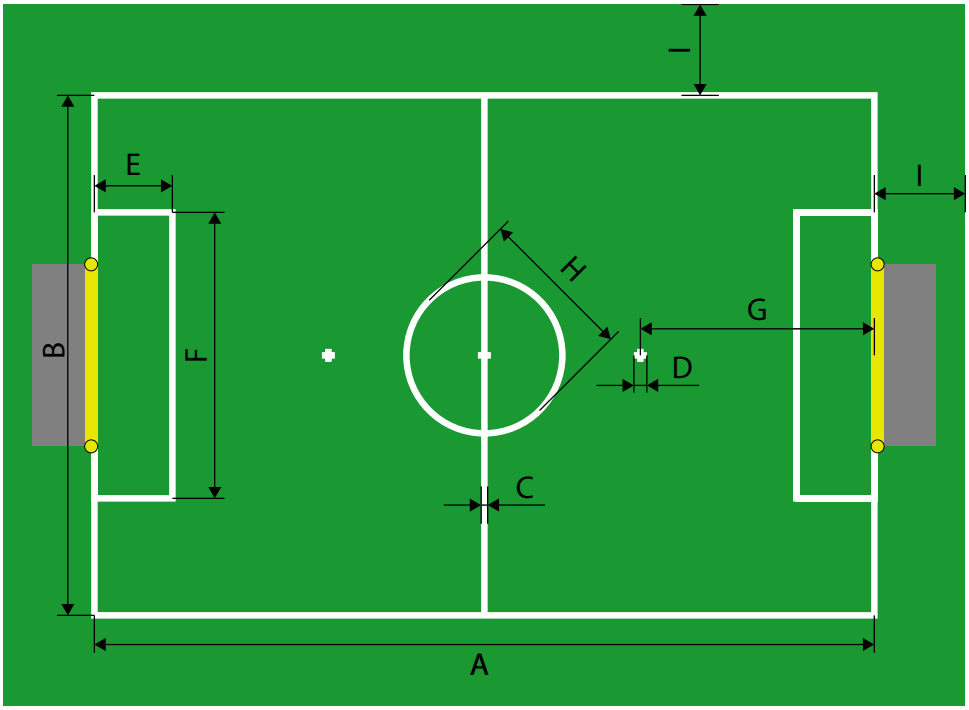
# Problem Statement

## 3.1 Robot Localization in RoboCup

Localization is one of the most fundamental problems in most robotic systems. In a dynamic adversarial environment, like RoboCup, most of the decisions a robot needs to make depend on its location. To reliably estimate its pose, a robot, as already stated, has to exploit information provided by its sensors.

RoboCup SPL games take place on a $9 \times 6$ field, according to the 2013 SPL rules [2]. The field is built on a total carpet area of length 10.4m and width 7.4m and includes a number of static features (Figure 3.1), which may be used as landmarks: two same-colored goals, the center circle, two penalty marks, several field lines, field-line corners, and field-line T-junctions. These static landmarks, if correctly recognized, can play an important role in the localization process. However, there is a high degree of perceptual aliasing, because of the field symmetry. More specifically, some of these landmarks are not unique and/or may be perceived the same from different locations of the field. The teams competing in the SPL have implemented various methods of recognizing these landmarks and extracting information about their type, distance, and bearing, which acts as input to the localization algorithm.

The hardware platform currently used in the SPL, Aldebaran's Nao humanoid robot, has omni-directional walk capabilities, that is, it has the ability to move in any possible direction on the ground with or without rotation. Several teams have implemented their own walk engine, which is responsible for determining foot and/or arm trajectories in order for the robot to execute a specific walk command. The expected robot movement




| ID | Description | Length |
|---|---|---|
| A | Field length | 9000 |
| B | Field width | 6000 |
| C | Line width | 50 |
| D | Penalty mark size | 100 |
| | | |

| ID | Description | Length |
|---|---|---|
| E | Penalty area length | 600 |
| F | Penalty area width | 2200 |
| G | Penalty mark distance | 1300 |
| H | Center circle diameter | 1500 |
| I | Border strip width | 700 |

Figure 3.1: Specifications of the SPL 2013 field and dimensions in mm

can then be computed analytically and provided as odometry. In our team we currently make use of the walk engine provided by Aldebaran Robotics, which reports odometry information as described above.

Nao is also equipped with two CCD cameras which provide images of 640 × 480 pixels and operate at 30 fps. An image processing system is responsible for recognizing objects of interest in the images and extracting information about their type, distance, and bearing. Currently, our vision module is able to detect the ball and the two goals; only the latter can be used as landmark for localization.

The localization problem in RoboCup SPL is a special case of the global localization problem, closer to the pose tracking problem. More specifically, the robot has some prior knowledge about its initial pose, namely that this pose is restricted to a limited number of candidate field areas, and therefore it doesn't have to infer this pose from scratch, neither does it know this pose precisely. These candidate field areas include: the two sides of the own half of the field and certain predetermined poses within the own half of the field used at kick-off. It is important for the robot to disambiguate its initial pose quickly and track its pose thereafter, because after the switch to same-colored goals in 2012, the field has become fully symmetric and it is difficult to recover from failures during the game.

Apart from its own pose, a robot must be able to track and quantify the positions of moving objects, such as the ball or other robots. Their global position in the field can be estimated by transforming their robot-relative position to the global coordinate frame. Clearly, the accuracy of these estimations heavily depend the accuracy of the robot own pose estimation. This information about the global positions of other objects can benefit the team, if it is shared among the robots towards building a comprehensive estimate of the global state by combining the individual beliefs.

Finally, team coordination and strategies have started to play an important role in SPL games in recent years. The strategy a team will develop depends on the global state of the game (ball location, team robot poses, opponent robot poses, etc.). Therefore, it is essential for our robots to employ a robust localization method and maintain a correct local belief, so that they can build an accurate global belief and acquire appropriate roles towards a beneficial team strategy.

## 3.2 Related Work

Most localization algorithms which have been applied in the domain of RoboCup are algorithms based on the Bayes filter, such as particle or Kalman filters. Below, an overview of the most notable localization methods used in RoboCup Soccer SPL is given.

### 3.2.1 Kalman Filter Approaches

Team **Austin Villa** uses a multi-modal 7-dimensional Unscented Kalman Filter (UKF) for localization [26]. The 7-dimensional filter is used to track both the pose of the robot

along with the position and velocity of the ball on the plane. A multiple hypothesis approach has been used, representing the belief by a weighted sum of $N$ Gaussians. Each of the $N$ models has its own state estimate, covariance matrix, and a weight. Other teams that rely on UKF for robot localization are **Nao Devils** [27] and **RoboEireann** [28].

Team **Bembelbots** also make use of a multi-modal Kalman filter [29]. Their vision system is capable of detecting lines, corners, and the center circle which are used for computing all likely positions of the robot in the field. These positions act as absolute measurements for their Kalman filter. To solve the problem of field symmetry, they have introduced two additional measurements: the use of infrared emission by the goalkeeper and the fused ball position. Both these measurements aid the localization of field players by providing a strong hint about their orientation.

Team **NTU RoboPAL** creates an augmented state, which contains the poses of all robots of the team and information about the ball and other moving objects [30]. Their robots perform EKF filtering over this augmented state. In the prediction step of the algorithm, they use the odometry motion model for the teammate robots and they apply a multiple model tracking approach for moving objects, as there is no odometry information, to account for the motion mode uncertainty. In the update step of the algorithm, three types of measurements are aggregated: relative information between the teammate robots and the map, relative information between two teammate robots, and relative information between teammate robots and moving objects. Before updating with the incoming observations, the data associations must be established. A maximum likelihood data association algorithm is applied, with a threshold gating on the Mahalanobis distance between the incoming observation and the expected observation. Finally, tracks of moving objects are initialized when new ones have been detected and others are pruned when they have not been observed for a long time.

Team **Runswift** also uses a Kalman filter-based localization technique. In 2012, they extended their work on field-line matching by implementing a variation of the Iterative Closest Point (ICP) algorithm which can be used together with a Kalman filter as an alternative to maximum likelihood data association [31]. The proposed algorithm iteratively examines the set of observed non-unique landmarks extracted from a frame trying to come up with a single robot pose. At each iteration a new pose is estimated rotating and translating the prior robot pose, to approximately minimize the squared positioning error over all observed features simultaneously. If the mean squared error is sufficiently

small, the robot pose estimate can then be provided as a measurement to the localization filter.

### 3.2.2 Particle Filter Approaches

Team **B-Human**'s self-localization is realized by a particle filter implementation [32]. A number of extensions have been applied for improving localization performance such as an efficient method for extracting robot poses from a particle set [33]. A particle swarm optimization has also been used to optimize particle filter parameters [34]. In recent years, different additional components have been added to their particle filter. For achieving a higher precision, an Unscented Kalman Filter was recently implemented to locally refine the particle filter's estimate, a validation module compares recent landmark observations to the estimated robot pose, and, finally, a side disambiguation method enables self-localization to deal with the two yellow goals. Other teams that rely on a particle filter for localization are Cerberus, DAInamite, Mi-Pal, and Northern Bites.

### 3.2.3 Constraint Localization Approaches

Team **Humbolt** investigates constraint-based techniques as alternatives to Bayesian approaches for localization [35]. The problem of localization is modeled as a constraint satisfaction problem (CSP). The key idea is to restrict the domains of the desired variables, such as the pose of the robot, by taking the conjunction of all constraints resulting from an image. The creation of the constraints is done as follows: constraints given by distances to landmarks are defined as circular rings and constraints given by observed field lines are defined by a set of rectangles and angles. Iteratively examining the constrains resulting from an observation, a set of possible poses that satisfy all the existing constraints is derived. When the robot moves, the constraint boundaries are shifted towards the movement direction and together with the sensor constraints at the current time step provide a set of new possible poses.

### 3.2.4 Hybrid Approaches

Team **SPI team** combines Markov localization together with Kalman filters [36]. The environment is divided into a probabilistic coarse grain grid and each of the cells of

the grid has an associated probability which is updated using the available perceptual information. This grid is used to define new locations in the environment where it is appropriate to initialize a new extended Kalman filter. Once an extended Kalman filter is created, it runs in parallel with the rest of the filters in the population. Each filter independently incorporates motion and perceptual information. The robot pose is selected among the filters by taking into account the Markovian grid. To control the population dynamics additional operations regarding the combination and destruction of the filters are applied.

# Chapter 4

# Our Approach

A new approach, based on Extended Kalman Filter (EKF), has been implemented for our localization module, which replaced our previous Monte-Carlo localization approach with particle filters. The benefit of EKF, besides being computationally more efficient, is that it enables easier integration of robot and ball positions into a shared global state model, which can be used for developing team strategies or informing uncertain robots about the global state.

The localization algorithm that runs as a separate activity, called `LocalWorldState`, is responsible for estimating and providing a local robot belief. This local robot belief includes an estimate of the robot pose in the SPL field, which is described by a three-dimensional vector:

$$\mathbf{x}_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad \begin{array}{l} x \in [-5.2, +5.2] \\ y \in [-3.7, +3,7] \\ \theta \in [-\pi, +\pi] \end{array} \tag{4.1}$$

where $x$ and $y$ are the Cartesian coordinates and $\theta$ is the orientation at time step $k$. The algorithm consists of two main phases, prediction and update. In the prediction phase, we estimate the resulting pose of the robot given its previous pose and the displacement acquired from the odometry system. In the update phase, the robot uses information from the observed landmarks (goals) to refine the estimate.

Given the state defined above, the system transitions and observations are described by the following equations:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{4.2}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{4.3}$$

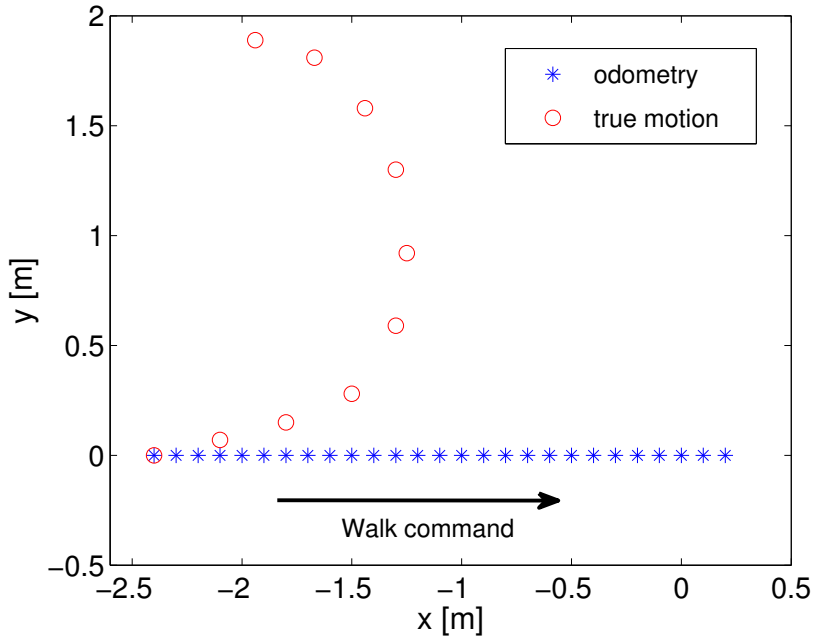



Figure 4.1: Actual robot trace and reported odometry for a straight walk command

in which $f(\mathbf{x}_{k-1}, \mathbf{u}_k)$ is a non-linear function that relates the previous pose $\mathbf{x}_{k-1}$ and the odometry displacement $\mathbf{u}_k$ to the current pose $\mathbf{x}_k$, and $h(\mathbf{x}_k)$ is a non-linear function that gives us the expected observation $\mathbf{z}_k$ given the current pose estimate $\mathbf{x}_k$ and the known location of the landmark observed. The terms $\mathbf{w}_k \sim N(0, \mathbf{Q})$ and $\mathbf{v}_k \sim N(0, \mathbf{R})$ are the motion and observation noise respectively. The key filtering steps (predict and update) are described in the following sections.

## 4.1 Prediction Step

In this phase we calculate the pose estimate $\hat{\mathbf{x}}_{k|k-1}$ and the covariance $\mathbf{P}_{k|k-1}$ according to the odometry motion model. The odometry provided by the robot is computed based on the motion commands received without feedback from sensors, making it rather inaccurate, thus it cannot be used alone for reliable navigation. Figure 4.1 shows the discrepancy between the actual robot trace and the reported odometry for a straight walk command; odometry "believes" that the robot moved on a straight line because of the given command, whereas in reality the robot slipped to the left. Such inaccuracies

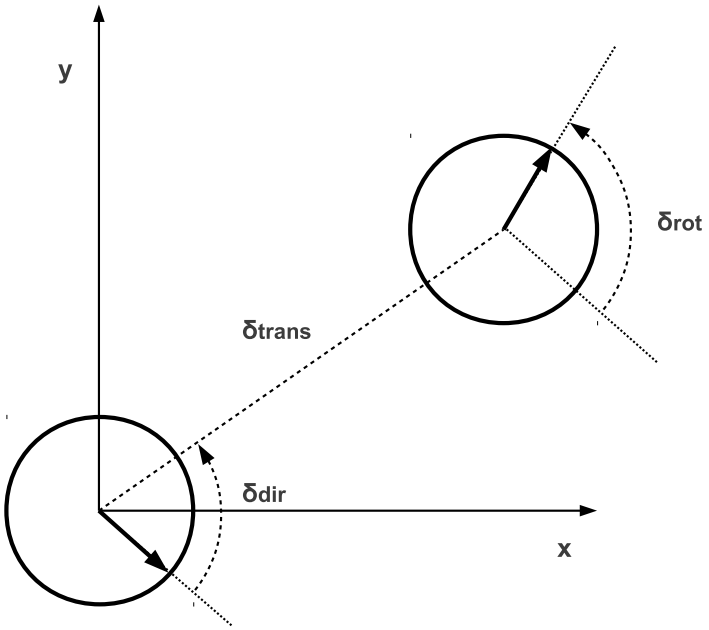



Figure 4.2: Odometry motion model

are due to partial hardware failures, out-of-specs components, variable frictions, etc.

The robot odometry updates a pose $\bar{\mathbf{x}}_k$ with respect to an internal robot pose coordinate system which is totally independent from and unrelated to our field coordinate system. Therefore, we compute the change between consecutive poses $\bar{\mathbf{x}}_{k-1}$ and $\bar{\mathbf{x}}_k$ in the robot internal coordinate system to obtain the odometry displacement $u_k$ used in our localization system. Robot movement in the interval $[k-1, k]$ is decomposed into three variables; the change in distance $\delta_{trans}$, direction $\delta_{dir}$, and rotation $\delta_{rot}$ (Figure 4.2). These quantities that form the odometry displacement $\mathbf{u}_k$ are obtained by the following equations:

$$\mathbf{u}_k = \begin{bmatrix} \delta_{trans,k} \\ \delta_{dir,k} \\ \delta_{rot,k} \end{bmatrix} = \begin{bmatrix} \sqrt{(\bar{x}_k - \bar{x}_{k-1})^2 + (\bar{y}_k - \bar{y}_{k-1})^2} \\ \texttt{atan2}(\bar{y}_k - \bar{y}_{k-1}, \bar{x}_k - \bar{x}_{k-1}) - \bar{\theta}_{k-1} \\ \bar{\theta}_k - \bar{\theta}_{k-1} \end{bmatrix} \tag{4.4}$$

Transforming $\mathbf{u}_k$ into our field coordinate system, $\hat{\mathbf{x}}_{k|k-1}$ can then be calculated as

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) = \hat{\mathbf{x}}_{k-1|k-1} + \begin{bmatrix} \delta_{trans,k} \cos(\hat{\theta}_{k-1|k-1} + \delta_{dir,k}) \\ \delta_{trans,k} \sin(\hat{\theta}_{k-1|k-1} + \delta_{dir,k}) \\ \delta_{rot,k} \end{bmatrix} \tag{4.5}$$

The corresponding covariance matrix is updated according to

$$\mathbf{P}_{k|k-1} = \mathbf{F}_x \mathbf{P}_{k-1|k-1} \mathbf{F}_x^\top + \mathbf{F}_u \mathbf{M}_k \mathbf{F}_u^\top + \mathbf{Q} \tag{4.6}$$

where $\mathbf{P}$ is the state covariance matrix and represents the uncertainty of the estimated state, $\mathbf{M}$ is the covariance matrix of the noise that corrupts the input, $\mathbf{Q}$ is a diagonal matrix that represents the uncertainty of the model. Note that the diagonal form of $\mathbf{M}$ and $\mathbf{Q}$ implies our assumption that the noise is independent in each of the three dimensions. Matrices $\mathbf{F}_x$ and $\mathbf{F}_u$ are the Jacobians of the system. The Jacobian $\mathbf{F}_x$ is the derivative of $f(\mathbf{x}, \mathbf{u})$ with respect to $\mathbf{x}$ evaluated at $\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k$:

$$\mathbf{F}_x = \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k} = \begin{bmatrix} 1 & 0 & -\delta_{trans,k} \sin(\hat{\theta}_{k-1|k-1} + \delta_{dir,k}) \\ 0 & 1 & +\delta_{trans,k} \cos(\hat{\theta}_{k-1|k-1} + \delta_{dir,k}) \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

The Jacobian $\mathbf{F}_u$ is the derivative of $f(\mathbf{x}, \mathbf{u})$ with respect to $\mathbf{u}$ evaluated at $\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k$:

$$\mathbf{F}_u = \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k} = \begin{bmatrix} -\delta_{trans,k} \sin(\hat{\theta}_{k-1|k-1} + \delta_{dir,k}) & \cos(\hat{\theta}_{k-1|k-1} + \delta_{dir,k}) & 0 \\ +\delta_{trans,k} \cos(\hat{\theta}_{k-1|k-1} + \delta_{dir,k}) & \sin(\hat{\theta}_{k-1|k-1} + \delta_{dir,k}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

$$\mathbf{M}_k = \begin{bmatrix} a_1 \delta_{trans,k}^2 & 0 & 0 \\ 0 & a_2 \delta_{trans,k}^2 & 0 \\ 0 & 0 & a_3 \delta_{rot,k}^2 + a_4 \delta_{trans,k}^2 \end{bmatrix} \tag{4.9}$$

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{4.10}$$

The parameters $a_1$, $a_2$, $a_3$, and $a_4$ of matrix $\mathbf{M}$, as well as the values $\sigma_x$, $\sigma_y$, and $\sigma_\theta$ of matrix $\mathbf{Q}$ have been estimated experimentally.

## 4.2 Update Step

The pose estimate made in the prediction step is updated using observations of fixed landmarks. Each observation is represented by a vector $\mathbf{z} = [r, \phi]^\top$, where $r$ is the distance and $\phi$ the bearing of the observed landmark relative to robot's local (egocentric) coordinate frame (Figure 4.3). Each observation $\mathbf{z}$ is also associated with a specific

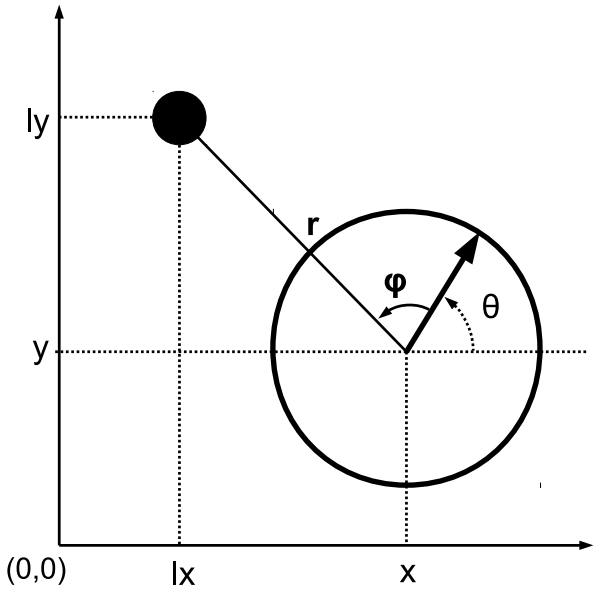



Figure 4.3: Observation of landmark at $(lx, ly)$ from robot at pose $(x, y, \theta)$

landmark $\mathbf{l}$, which is described by its fixed coordinates in the field $(lx, ly)$. The expected observation $\hat{\mathbf{z}}_k$, given a pose estimate $\hat{\mathbf{x}}_k$, is computed as follows:

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k) = \begin{bmatrix} \hat{r}_k \\ \hat{\phi}_k \end{bmatrix} = \begin{bmatrix} \sqrt{(lx_k - \hat{x}_k)^2 + (ly_k - \hat{y}_k)^2} \\ \texttt{atan2}(ly_k - \hat{y}_k, lx_k - \hat{x}_k) - \hat{\theta}_k \end{bmatrix} \tag{4.11}$$

The variables $\hat{r}_k$ and $\hat{\phi}_k$ denote the expected distance and bearing to the observed landmark $\mathbf{l}_k$ of observation $\mathbf{z}_k$ made at time $k$. The actual observation $\mathbf{z}_k$ is compared to the predicted observation $\hat{\mathbf{z}}_k$ and used to obtain the updated state and covariance using the Kalman gain matrix:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\big(\mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1})\big) \tag{4.12}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1} \tag{4.13}$$

The difference between the actual and the expected observation $\mathbf{z}_k - \hat{\mathbf{z}}_k$ is called innovation or measurement residual. Matrix $\mathbf{K}_k$ is known as the optimal Kalman gain, because it minimizes the a posteriori estimate error covariance and is computed as follows:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^\top \mathbf{S}_k^{-1} \tag{4.14}$$

where

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R} \tag{4.15}$$

$$\mathbf{H}_k = \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}} = \begin{bmatrix} -\frac{(lx_k - \hat{x}_{k|k-1})}{\sqrt{q}} & -\frac{(ly_k - \hat{y}_{k|k-1})}{\sqrt{q}} & 0 \\ \frac{(ly_k - \hat{y}_{k|k-1})}{q} & -\frac{(lx_k - \hat{x}_{k|k-1})}{q} & -1 \end{bmatrix} \tag{4.16}$$

$$q = (lx_k - \hat{x}_{k|k-1})^2 + (ly_k - \hat{y}_{k|k-1})^2 \tag{4.17}$$

$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \tag{4.18}$$

The Jacobian matrix $\mathbf{H}_k$ contains the partial derivatives of the sensor model $h$ with respect to the state $\mathbf{x}$, evaluated at the prior state estimate $\hat{\mathbf{x}}_{k|k-1}$. Matrix $\mathbf{R}$ is a diagonal matrix that represents the uncertainty of the observation, where we have assumed that the noise is independent in each of the two dimensions. The quantities $\sigma_r^2$ and $\sigma_\phi^2$ of matrix $\mathbf{R}$ have been estimated experimentally.

Finally, assuming that landmark observations are conditionally independent we can process multiple observations consecutively in case there are more than one landmark visible in a frame.

## 4.3 Multiple Hypothesis Tracking

The extended Kalman filter described so far assumes that each observation made corresponds to a unique landmark. However, landmark correspondences between observations and the known map are often difficult to determine with certainty. For example, when a robot observes only the lower part of a goalpost, it cannot determine its type (right or left). Even if the whole post is visible it is not clear if it corresponds to the own- or opponent- side goal of the field. The same applies to other landmarks, such as field-line corners and field-line T-junctions. In this case the update step of the algorithm cannot be applied without first associating the observation made to the correct landmark.

The observations currently used for localization in our team come from the two goals in the field. When observing the upper part of a goalpost, the robot can report its type (right or left), otherwise an ambiguous goalpost observation is reported. To cope with

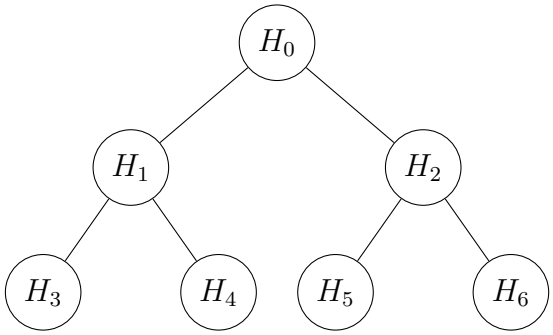



Figure 4.4: Hypotheses splitting after two ambiguous observations

the data association problem, a Multiple Hypothesis Tracking (MHT) technique has been applied [37].

The key idea behind MHT is to maintain a number of robot poses each one deriving from a different measurement correspondence. More specifically, when an observation is made, each existing hypothesis splits into multiple hypotheses, one for each possible association. The resulting hypotheses are then split again, when new observations are ambiguous and so on. The result is a tree structure (Figure 4.4). Each of the resulting hypotheses is a separate extended Kalman filter, considering the corresponding pose of the robot with a different Gaussian distribution.

### 4.3.1 Hypothesis Representation

Each hypothesis $H^{(i)}$ in the current pool of hypotheses (leaves of the tree) is defined by a robot pose estimate $\mathbf{x}^{(i)}$ and a corresponding covariance matrix, $\mathbf{P}^{(i)}$. Each hypothesis also has a weight attached to it, which measures the probability that the hypothesis is correct, $\pi^{(i)}$.

$$H^{(i)} = \{\mathbf{x}^{(i)}, \mathbf{P}^{(i)}, \pi^{(i)}\} \tag{4.19}$$

The Probability Density Function (PDF) of the robot pose can be approximated by a Gaussian mixture

$$bel(\mathbf{x}_k) = \sum_{i=1}^{N_k} p\left(\mathbf{x}_k \mid \mathbf{H}_k^{(i)}\right) \pi_k^{(i)} \tag{4.20}$$

where $N_k$ is the number of hypotheses at time step $k$ and the weights satisfy

$$\sum_{i=1}^{N_k} \pi_k^{(i)} = 1 \tag{4.21}$$

The fact that the probabilities $\pi_k^{(i)}$ at a given time step $k$ sum up to 1 is equivalent to assuming that one of the hypotheses is correct.

### 4.3.2 Initialization

The initialization of a robot's belief depends on the game state as communicated by the game controller in case wireless communication is available or by the robot button interface.

- **Ready State**

  Right before kick-off at the beginning of each half of the game (Ready state), the robots are placed besides the sideline, in their own half of the field, in positions chosen by the team. Knowing the exact initial (chosen) pose of the robot, a single hypothesis is sufficient to represent its belief. Figure 4.5 shows the initial belief of a robot and the corresponding covariance in Ready state, when the chosen initial pose is on the right side of the own goal. In this state, the robots walk to their legal kick-off positions, until the head referee decides that there is no significant progress anymore or a maximum of 45 seconds has passed.

- **Set State**

  In the Set state that follows the Ready state, the robots stop and wait for kick-off. If they have not reached their legal kick-off positions as specified by the rules, they are placed manually by the assistant referees at the predetermined positions for manually placement 4.6. In this case, a single hypothesis is sufficient to represent each robot's belief, because these predetermined positions are known.

- **Penalized State**

  In the Penalized state, the robot is removed from the field for 30 seconds and its return position depends on the current location of the ball. The robot is placed on the own half of the field, on the side line near the point where the penalty mark

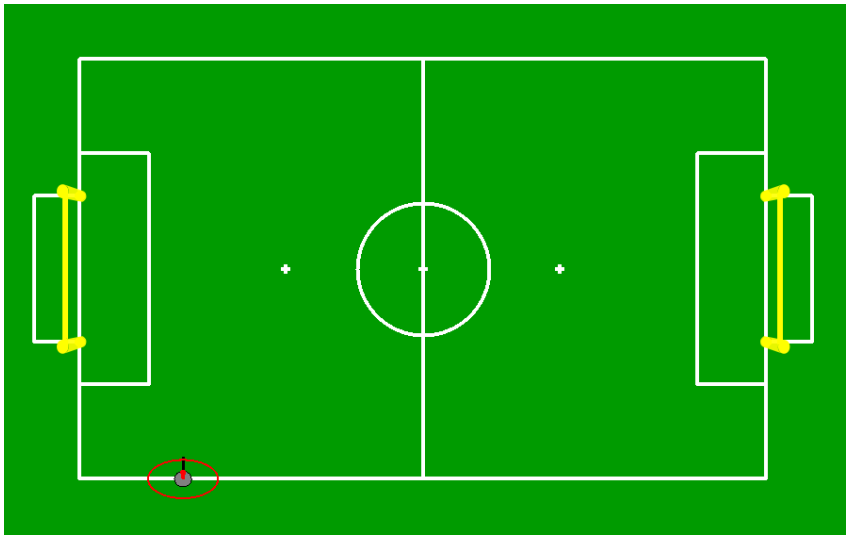


Figure 4.5: Belief initialization for one robot in Ready state

projects, and on the left or right side of the own goal that is currently farthest from the ball. For belief initialization, two hypotheses are generated, one at each side (left or right) of the own half of the field having equal weight.

- **No Communication**

  Finally, when a robot is unable to communicate with the game controller, the robot button interface is used to indicate the game state, which includes only transitions between Penalized and Playing states. In this case, the true game state of Set or Penalized is viewed uniformly as Penalized. Therefore, the robot cannot distinguish whether it comes out of a Set state or a Penalized state, when transitioning to Playing state. Thus, to cover all possibilities for belief initialization, three hypotheses are created having equal weights; one at the predetermined manual position and two at the left and right sides of the own half of the field.

### 4.3.3 Incorporating Odometry

Each time the robot receives the odometry displacement $\mathbf{u}_k$, it incorporates the information in each of the EKF hypotheses using the EKF prediction equations (Section 4.1).

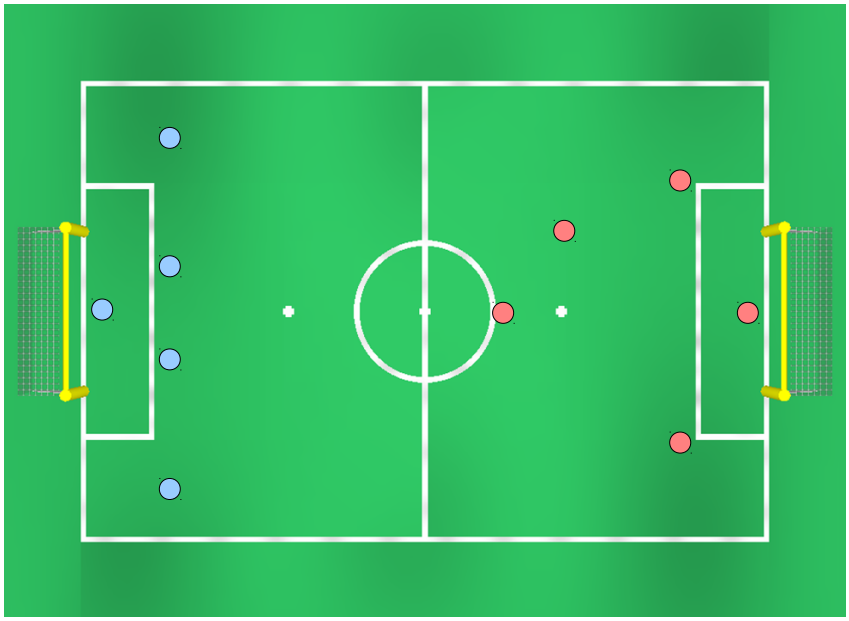


Figure 4.6: Predetermined positions for manual placement of both teams in Set state

$$\hat{\mathbf{x}}_{k|k-1}^{(i)} = f(\hat{\mathbf{x}}_{k-1|k-1}^{(i)}, \mathbf{u}_k) \tag{4.22}$$

$$\mathbf{P}_{k|k-1}^{(i)} = \mathbf{F}_x^{(i)} \mathbf{P}_{k-1|k-1}^{(i)} {\mathbf{F}_x^{(i)}}^\top + \mathbf{F}_u^{(i)} \mathbf{M}_k {\mathbf{F}_u^{(i)}}^\top + \mathbf{Q} \tag{4.23}$$

$$\pi_{k|k-1}^{(i)} = \pi_{k-1|k-1}^{(i)} \tag{4.24}$$

Note that the weights of the hypotheses remain the same during the prediction step, because the odometry information cannot affect their correctness.

### 4.3.4 Incorporating Observations

There are four types of observations currently available for localization in our team:

- **Left Goalpost**

  In this case, the left goalpost of a goal has been detected. Since there are two left goalposts in the field, this observation is associated to two non-distinguishable landmarks with symmetric known locations (Figure 3.1).

- **Right Goalpost**

  In this case, the right goalpost of a goal has been detected. Since there are two right goalposts in the field, this observation is associated to two non-distinguishable landmarks with symmetric known locations (Figure 3.1).

- **Left and Right Goalposts**

  In this case, the left and the right goalposts of one goal have been detected. Due to physical limitations, these two goalposts must belong to the same goal, meaning that it is physically impossible for the robot to observe the left goalpost of one goal and the right goalpost of the other goal at the same time with its camera. Since there are two goals in the field, this observation is associated to two non-distinguishable landmarks with symmetric known locations (Figure 3.1).

- **Ambiguous Goalpost**

  In this case, a goalpost has been detected without any indication of being left or right. Since there are four goalposts in total in the field, this observation is associated to four non-distinguishable landmarks with symmetric known locations (Figure 3.1).

When an observation $\mathbf{z}_k$ is reported that may come from some landmark $\mathbf{l}^{(j)}$ of a set of $L_k$ non-distinguishable landmarks, each of the $N_k$ existing hypotheses $H_k^{(i)}$ is updated using all possible data associations, as described above, producing $N_k L_k$ resulting hypotheses.

$$\hat{\mathbf{x}}_{k|k}^{(i,j)} = \hat{\mathbf{x}}_{k|k-1}^{(i)} + \mathbf{K}_k^{(i,j)} \left( \mathbf{z}_k - \hat{\mathbf{z}}_k^{(i,j)} \right) \tag{4.25}$$

$$\mathbf{P}_{k|k}^{(i,j)} = \mathbf{P}_{k|k-1}^{(i)} - \mathbf{K}_k^{(i,j)} \mathbf{H}_k^{(i,j)} \mathbf{P}_{k|k-1}^{(i)} \tag{4.26}$$

where $(i, j)$ implies the resulting hypothesis from the association of the existing hypothesis $i$ with the landmark $j$. The EKF only updates the pose estimates of the different hypotheses, but not their probability/weight of being the correct pose. The $L_k$ non-distinguishable landmarks are taken to be equal probable, therefore these weights will be affected only by the likelihood of each landmark for different hypotheses:

$$\pi_{k|k}^{(i,j)} = \alpha p \left( \mathbf{z}_k = \mathbf{l}_k^{(j)} \mid H_k^{(i)} \right) \pi_{k|k-1}^{(i)} \tag{4.27}$$

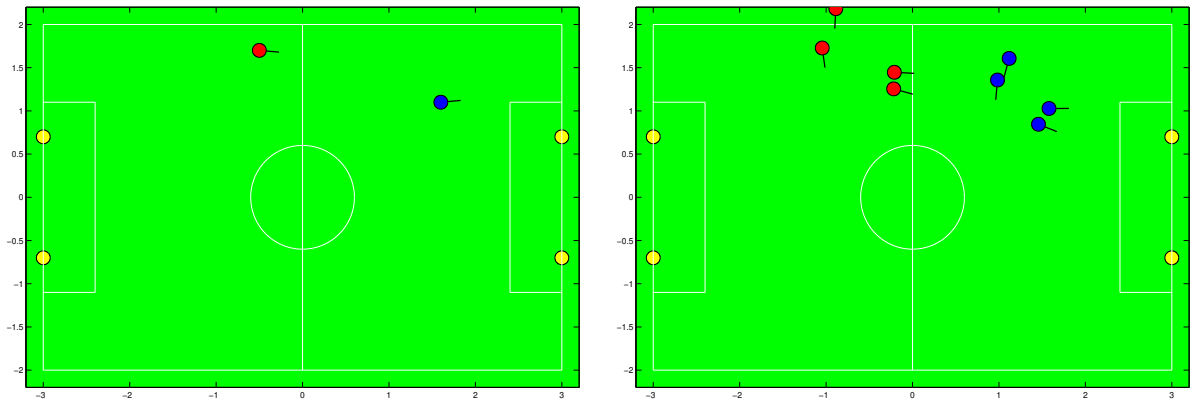


Figure 4.7: Initial (left) and resulting (right) hypotheses with an ambiguous goalpost

where $\alpha$ is a normalizing factor. The likelihood of observing the unique landmark $\mathbf{l}_k^{(j)}$ given that hypothesis $H_k^{(i)}$ is correct is given by:

$$p\left(\mathbf{z}_k = \mathbf{l}_k^{(j)} \mid H_k^{(i)}\right) = \frac{1}{2\pi^{\frac{1}{2}} \det\left(\mathbf{S}_k^{(i,j)}\right)^{-\frac{1}{2}}} e^{-\frac{1}{2}\left(\mathbf{z}_k - h\left(\hat{\mathbf{x}}_{k|k-1}^{(i)}\right)\right)^{\top}\left(\mathbf{S}_k^{(i,j)}\right)^{-1}\left(\mathbf{z}_k - h\left(\hat{\mathbf{x}}_{k|k-1}^{(i)}\right)\right)} \quad (4.28)$$

Figure 4.7 shows a simple example of incorporating an ambiguous goalpost observation into a robot belief represented by two hypotheses. The true pose of the robot is near the hypothesis on the right side of the field (shown in blue) and the observation comes from the left goalpost of the goal on the right side of the field. Four new hypotheses will be created for each of the two existing hypotheses yielding a total of eight hypotheses in the resulting belief. Only three hypotheses out of eight will receive high weights, namely the one that almost coincides with the true pose ($\pi \approx 0.713$), the one slightly below the true pose in the field ($\pi \approx 0.133$), and one of the hypotheses on the left side ($\pi \approx 0.122$). The weights of all the other hypotheses are negligible.

## 4.4 Merging Hypotheses

The main drawback of MHT is the exponential growth of the hypotheses. Many hypotheses acquire a very low weight and redundant ones are created. For this reason a mixture reduction algorithm was implemented [38]. This algorithm is based on the assumption

that the hypotheses with the largest weights carry the most important information in the pool of hypotheses. Thus, starting with the hypothesis having the largest weight, the algorithm initially gathers all surrounding hypotheses that are in some sense close to that dominant hypothesis to form a cluster around it. The distance measure chosen to represent the closeness of hypothesis $i$ to the center of the dominant hypothesis $c$ is defined as follows:

$$d(i,c) = \frac{\pi^{(i)}\pi^{(c)}}{\pi^{(i)} + \pi^{(c)}} \left(\mathbf{x}^{(i)} - \mathbf{x}^{(c)}\right)^{\top} \left(\mathbf{P}^{(c)}\right)^{-1} \left(\mathbf{x}^{(i)} - \mathbf{x}^{(c)}\right) \tag{4.29}$$

where $\pi^{(c)}$, $\mathbf{x}^{(c)}$, and $\mathbf{P}^{(c)}$ are the probability, mean, and covariance of the dominant hypothesis and $\pi^{(i)}$, $\mathbf{x}^{(i)}$ are the probability and mean of hypothesis $i$.

Any hypothesis $i$ for which $d < T$, where $T$ is a carefully-selected threshold, is taken as a cluster member. The second factor in Equation 4.29 implies that the clustering boundary would be a hyper-ellipsoid in the state space, which is a surface of constant probability density of the dominant hypothesis. The first factor in Equation 4.29 biases the distance measure so that hypotheses with small weights are more easily clustered.

The clustered hypotheses are then joined together to form a single hypothesis with mean and covariance matrix taken from the dominant hypotheses and weight equal to the sum of the weights of all hypotheses in the cluster. The newly-formed hypothesis replaces all hypotheses in the cluster. The decision to represent the cluster with the dominant hypothesis as opposed to some weighted mean was made to prevent small shifts in the parameter estimates of highly-weighted hypotheses that can be caused when merging with lowly-weighted ones which have different means.

The algorithm continues iteratively with the remaining hypotheses, always selecting the hypothesis with the largest weight as the dominant hypothesis of each new cluster, until all hypotheses have been examined. Clearly, when the algorithm terminates the number of resulting hypotheses will be less or equal to the initial one. The goal is to reduce the number of hypotheses below some threshold $\overline{N}$. If the algorithm terminates, but the desired amount of reduction has not been achieved, the hypotheses reduction algorithm is repeated with a larger threshold $T + \delta T$ on the set of hypotheses of the previous repeat.

Figrue 4.8 shows the resulting set of hypotheses from merging the eight hypotheses in the example shown in Figure 4.7 (right). Their total number exceeds the predefined

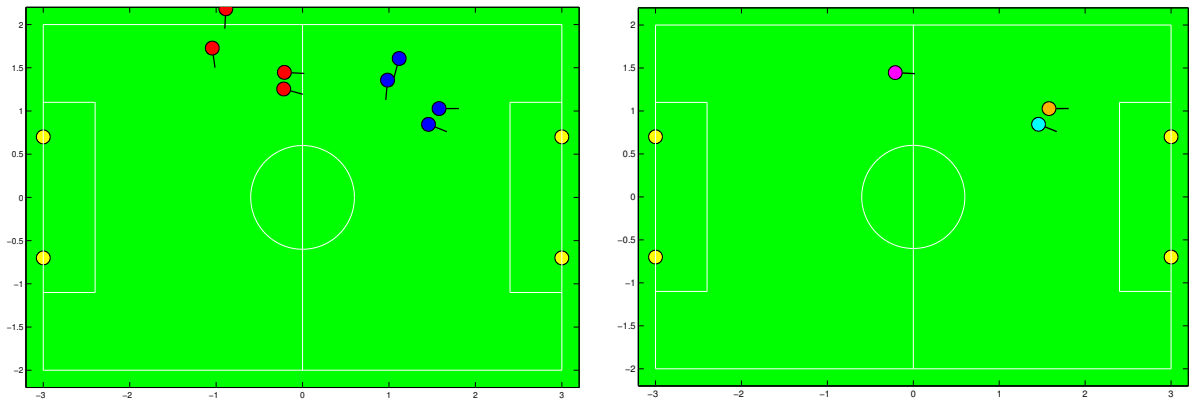


Figure 4.8: Hypotheses before (left) and after (right) merging

threshold $\overline{N} = 4$. After merging only three out of eight hypotheses remain in the pool, more specifically the ones in the pool with significant weights, which absorbed the hypotheses with negligible weights.

## 4.5 Odometry Calibration

Another issue that introduced significant errors in the robot localization belief was the erroneous odometer information that was not always consistent with the true locomotion actions. To overcome this problem and make the robot capable of maintaining a fairly correct belief for a long period of time only through prediction without observing any landmarks, the state was augmented to include three parameters related to the systematic error of the odometry system. These parameters are used to model errors, like drift and slippage, and are estimated online during filtering [39]. The importance of this decision can be understood better by considering that these odometry error parameters may differ from robot to robot, may be different for various surfaces, and may be slowly-changing over time due to actuators' characteristics.

Recall that the odometry displacement $\mathbf{u}_k$ was defined in Section 4.1 as:

$$\mathbf{u}_k = \begin{bmatrix} \delta_{trans,k} \\ \delta_{dir,k} \\ \delta_{rot,k} \end{bmatrix} \tag{4.30}$$

using the measured change in distance $\delta_{trans}$, direction $\delta_{dir}$, and rotation $\delta_{rot}$. Now, we assume that the true odometry displacement $\overline{\mathbf{u}}_k$ is affected by three parameters $e_1, e_2, e_3$ modeling systematic odometry errors:

$$\overline{\mathbf{u}}_k = \begin{bmatrix} e_1\delta_{trans,k} \\ \delta_{dir,k} \\ e_2\delta_{rot,k} + e_3\delta_{trans,k} \end{bmatrix} \tag{4.31}$$

Essentially, $e_1$ implies a multiplicative error in translation, $e_2$ a multiplicative error in rotation, and $e_3$ an additive error in rotation which depends multiplicatively on translation. Note that we assume no error in direction. The parameter values $(e_1, e_2, e_3) = (1, 1, 0)$ imply no error in odometry measurements. The (unknown) true values of $e_1, e_2, e_3$ will be estimated through filtering.

The augmented state $\overline{\mathbf{x}}_k$ contains the position and orientation of the robot as before and three additional state variables for $e_1, e_2, e_3$ and is described by the following six-dimensional vector

$$\overline{\mathbf{x}}_k = \begin{bmatrix} x & y & \theta & e_1 & e_2 & e_3 \end{bmatrix}^\top \tag{4.32}$$

The mean of the predicted state is now given by the following equations

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k-1} + \hat{e}_{1,k-1}\delta_{trans,k}\cos(\hat{\theta}_{k-1} + \delta_{dir,k}) \\ \hat{y}_k &= \hat{y}_{k-1} + \hat{e}_{1,k-1}\delta_{trans,k}\sin(\hat{\theta}_{k-1} + \delta_{dir,k}) \\ \hat{\theta}_k &= \hat{\theta}_{k-1} + \hat{e}_{2,k-1}\delta_{rot,k} + \hat{e}_{3,k-1}\delta_{trans,k} \\ \hat{e}_{1,k} &= \hat{e}_{1,k-1} \\ \hat{e}_{2,k} &= \hat{e}_{2,k-1} \\ \hat{e}_{3,k} &= \hat{e}_{3,k-1} \end{aligned} \tag{4.33}$$

The corresponding covariance is a $6 \times 6$ matrix, updated according to

$$\overline{\mathbf{P}}_{k|k-1} = \overline{\mathbf{F}}_x\overline{\mathbf{P}}_{k-1|k-1}\overline{\mathbf{F}}_x^\top + \overline{\mathbf{F}}_u\mathbf{M}_k\overline{\mathbf{F}}_u^\top + \overline{\mathbf{Q}} \tag{4.34}$$

The Jacobian $\overline{\mathbf{F}}_x$, the derivative of $f(\overline{\mathbf{x}}, \overline{\mathbf{u}})$ with respect to $\overline{\mathbf{x}}$, is now given by the following $6 \times 6$ matrix:

$$\overline{\mathbf{F}}_x = \begin{bmatrix} \mathbf{F}_x & \mathbf{G}_x \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.35}$$

where

$$\mathbf{F}_x = \begin{bmatrix} 1 & 0 & -\hat{e}_{1,k-1}\delta_{trans,k}\sin(\hat{\theta}_{k-1} + \delta_{dir,k}) \\ 0 & 1 & +\hat{e}_{1,k-1}\delta_{trans,k}\cos(\hat{\theta}_{k-1} + \delta_{dir,k}) \\ 0 & 0 & 1 \end{bmatrix} \tag{4.36}$$

$$\mathbf{G}_x = \begin{bmatrix} \delta_{trans,k}\cos(\hat{\theta}_{k-1} + \delta_{dir,k}) & 0 & 0 \\ \delta_{trans,k}\sin(\hat{\theta}_{k-1} + \delta_{dir,k}) & 0 & 0 \\ 0 & \delta_{trans,k} & \delta_{rot,k} \end{bmatrix} \tag{4.37}$$

The Jacobian $\overline{\mathbf{F}}_u$, the derivative of $f(\overline{\mathbf{x}}, \overline{\mathbf{u}})$ with respect to $\overline{\mathbf{u}}$, is now given by the following $6 \times 3$ matrix:

$$\overline{\mathbf{F}}_u = \begin{bmatrix} \mathbf{F}_u \\ \mathbf{0} \end{bmatrix} \tag{4.38}$$

$$\mathbf{F}_u = \begin{bmatrix} -\hat{e}_{1,k-1}\delta_{trans,k}\sin(\hat{\theta}_{k-1} + \delta_{dir,k}) & \hat{e}_{1,k-1}\cos(\hat{\theta}_{k-1} + \delta_{dir,k}) & 0 \\ +\hat{e}_{2,k-1}\delta_{trans,k}\cos(\hat{\theta}_{k-1} + \delta_{dir,k}) & \hat{e}_{2,k-1}\sin(\hat{\theta}_{k-1} + \delta_{dir,k}) & 0 \\ 0 & \hat{e}_{2,k-1} & \hat{e}_{3,k-1} \end{bmatrix} \tag{4.39}$$

The process noise matrix, $\overline{\mathbf{Q}}$, is the $6 \times 6$ matrix

$$\overline{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \tag{4.40}$$

where $\mathbf{S}$ is a diagonal $3 \times 3$ matrix which implies a fictitious noise injection on the additional state variables of the augmented state to prevent them from converging to false estimates:

$$\mathbf{S} = \begin{bmatrix} \sigma_{e_1}^2 & 0 & 0 \\ 0 & \sigma_{e_2}^2 & 0 \\ 0 & 0 & \sigma_{e_3}^2 \end{bmatrix} \tag{4.41}$$

Matrix $\mathbf{Q}$ which is the covariance matrix of the noise that corrupts the input remains the same.

The updated state and covariance are computed using the Kalman gain matrix $\overline{\mathbf{K}}_k$ as described in Section 4.2:

$$\hat{\overline{\mathbf{x}}}_{k|k} = \hat{\overline{\mathbf{x}}}_{k|k-1} + \overline{\mathbf{K}}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) \tag{4.42}$$

$$\overline{\mathbf{P}}_{k|k} = \overline{\mathbf{P}}_{k|k-1} - \overline{\mathbf{K}}_k\overline{\mathbf{H}}_k\overline{\mathbf{P}}_{k|k-1} \tag{4.43}$$

The Kalman gain matrix $\overline{\mathbf{K}}_k$ is a $6 \times 2$ matrix given by:

$$\overline{\mathbf{K}}_k = \overline{\mathbf{P}}_{k|k-1}\overline{\mathbf{H}}_k^\top\overline{\mathbf{S}}_k^{-1} \tag{4.44}$$

where

$$\overline{\mathbf{S}}_k = \overline{\mathbf{H}}_k\overline{\mathbf{P}}_{k|k-1}\overline{\mathbf{H}}_k^\top + \mathbf{R} \tag{4.45}$$

The observations aren't directly related to the additional state variables, therefore matrix $\overline{\mathbf{H}}$, which is the derivative of the sensor model $h$ with respect to the state $\overline{\mathbf{x}}$, is a $2 \times 6$ matrix given by:

$$\overline{\mathbf{H}}_k = \begin{bmatrix} \mathbf{H}_k & \mathbf{0} \end{bmatrix} \tag{4.46}$$

## 4.6 The Proposed Algorithm

In this section we summarize our approach and describe the proposed localization algorithm. Our localization approach is based on a multi-hypotheses extended Kalman filter (MHT-EKF) on a six-dimensional state space (Equation 4.32), which includes the three robot pose variables and the three odometry error parameters. Note that, during localization, each robot tracks not only its pose, but also its own odometry error parameters. Algorithm 5 outlines our belief update from time step $k-1$ to time step $k$, while Table 4.1 lists all algorithm parameters along with the values we have adopted.

**Algorithm 5** MHT-EKF robot localization algorithm

**for** $i = 1$ to $N_{k-1}$ **do**
  Predict new hypothesis $H_k^{(i)}$ from $H_{k-1}^{(i)}$ (Eqs 4.22, 4.23, 4.24)
**end for**
**for** $i = 1$ to $N_{k-1}$ **do**
  **for** $j = 1$ to $L_k$ **do**
    Generate $H_k^{(i,j)}$ from hypothesis $H_k^{(i)}$ and landmark $\mathbf{l}^{(j)}$ (Eqs 4.25, 4.26, 4.27)
  **end for**
**end for**
**while** $N_k > \overline{N}$ **do**
  Sort hypotheses in accordance to their weights
  Create a new empty set $S$ of hypotheses
  **while** current hypotheses pool is not empty **do**
    Select dominant hypothesis $H_k^{(c)}$ in current pool
    **for all** the remaining hypothesis $H_k^{(i)}$ in current pool **do**
      Compute similarity measure $d(c, i)$ (Eq 4.29)
      **if** $d(c, i) < T$ **then**
        Select hypothesis $H_k^{(i)}$ as a member of cluster $c$
      **end if**
    **end for**
    Merge clustered hypotheses into a single new hypothesis and add it to $S$
    Remove clustered hypotheses from the current pool
  **end while**
  Make $S$ the current pool of hypotheses
  Increment threshold $T$ by $\delta T$
**end while**
**return** hypothesis with the largest weight

Table 4.1: Parameter values of localization algorithm

| **Parameter** | **Proposed Value** | **Description** |
|---|---|---|
| $a_1$ | 0.3 | Control input noise |
| $a_2$ | 0.3 | Control input noise |
| $a_3$ | 0.09 | Control input noise |
| $a_4$ | 0.1 | Control input noise |
| $\sigma_x$ | 0.002 | Motion model noise |
| $\sigma_x$ | 0.002 | Motion model noise |
| $\sigma_\theta$ | 0.002 | Motion model noise |
| $\sigma_r$ | 1.5 | Observation model noise |
| $\sigma_\phi$ | 0.35 | Observation model noise |
| $\overline{N}$ | 4 | Maximum number of hypotheses |
| $T$ | 0.07 | Hypotheses merging threshold |
| $\delta T$ | 0.05 | Hypotheses merging threshold increment |
| $\sigma_{e_1}$ | 0.002 | Fictitious noise of odometry error variables |
| $\sigma_{e_2}$ | 0.002 | Fictitious noise of odometry error variables |
| $\sigma_{e_3}$ | 0.0002 | Fictitious noise of odometry error variables |

# Chapter 5

# Results

In this chapter we present the results of our work. First, we need to clarify that our approach gives accurate estimates about the robot's position. The standard way of showing that a localization method works well is to compare the estimated state to the true state of the robot obtained by a "ground truth" source. This way an objective evaluation of localization accuracy is obtained. Unfortunately, in lack of a ground truth source, this comparison cannot be performed.

Instead, an experiment was performed in which the robot was given a specific sequence of motion commands and the final poses were measured manually on the field and compared to the localization algorithm output. In addition, we needed to test whether our solution could be executed fast enough in real game scenarios along with the other activities of the Monas architecture of our team. Finally, to illustrate the effect that the augmentation of state and the MHT approach have in our implementation we performed two additional experiments.

The experiments reported in this chapter were conducted on the SPL 2012 field $4m \times 6m$, which is smaller than the 2013 one, because of the setup in our robot lab. Nevertheless, the two field are proportionally the same. In addition, our localization approach can be customized to any field of arbitrary dimensions by simply changing an `.xml` file.

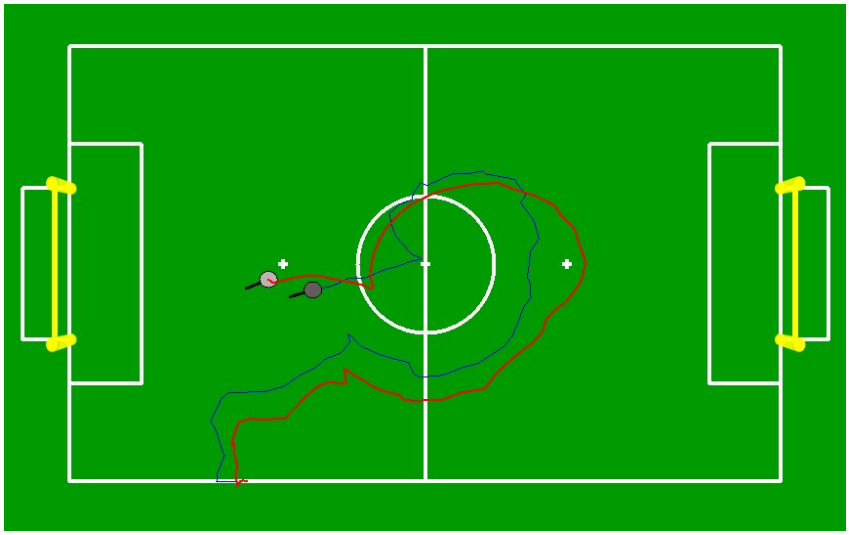


Figure 5.1: Estimated traces of MHT-EKF and MCL algorithm

## 5.1 Algorithm Accuracy

In the first experiment the robot was placed on the middle of the bottom-left side line of the field and a specific sequence of locomotion commands was given to it. As the robot moved, we executed simultaneously and independently two localization modules: the old one based on particle filters and the new one proposed in this thesis. This way both modules were given the exact same input in terms of initialization, controls, and observations. The number of particles in the MCL algorithm was set to 120, while the maximum number of hypothesis in our MHT-EKF approach was set to 6. Figure 5.1 shows an example of the estimated traces provided by the two algorithms during a single run.

The entire experiment was repeated fifteen times (to obtain average results) using the predetermined sequence of commands shown in Figure 5.2. The arguments represent velocities in the $x$, $y$, and $\theta$ axes respectively. The duration of each run was approximately 80 seconds. Despite the fact that the robot was given the same motion commands, the routes in different runs differ significantly. Figure 5.3 shows some of the final pose estimates reported by the two localization algorithms. The true state is represented by the yellow circle, the MHT-EKF estimate by the red circle, and the MCL estimate by the blue circle. The MHT-EKF algorithm gave an average error of 18cm in distance between

```
velocityWalk( 1.0,  0.0,  0.0 );   (10 seconds)
velocityWalk( 1.0,  0.0, -1.0 );   ( 3 seconds)
velocityWalk( 1.0,  0.0,  0.0 );   (10 seconds)
velocityWalk( 0.0,  0.0, -1.0 );   ( 3 seconds)
velocityWalk( 1.0,  0.0,  0.0 );   (19 seconds)
velocityWalk( 0.0, -1.0,  0.0 );   ( 3 seconds)
velocityWalk( 1.0,  0.0,  0.0 );   (19 seconds)
velocityWalk( 0.0, -1.0,  0.0 );   ( 6 seconds)
velocityWalk( 1.0,  0.0,  0.0 );   ( 7 seconds)
```

Figure 5.2: Sequence of walk commands for comparing MHT-EKF and MCL estimates

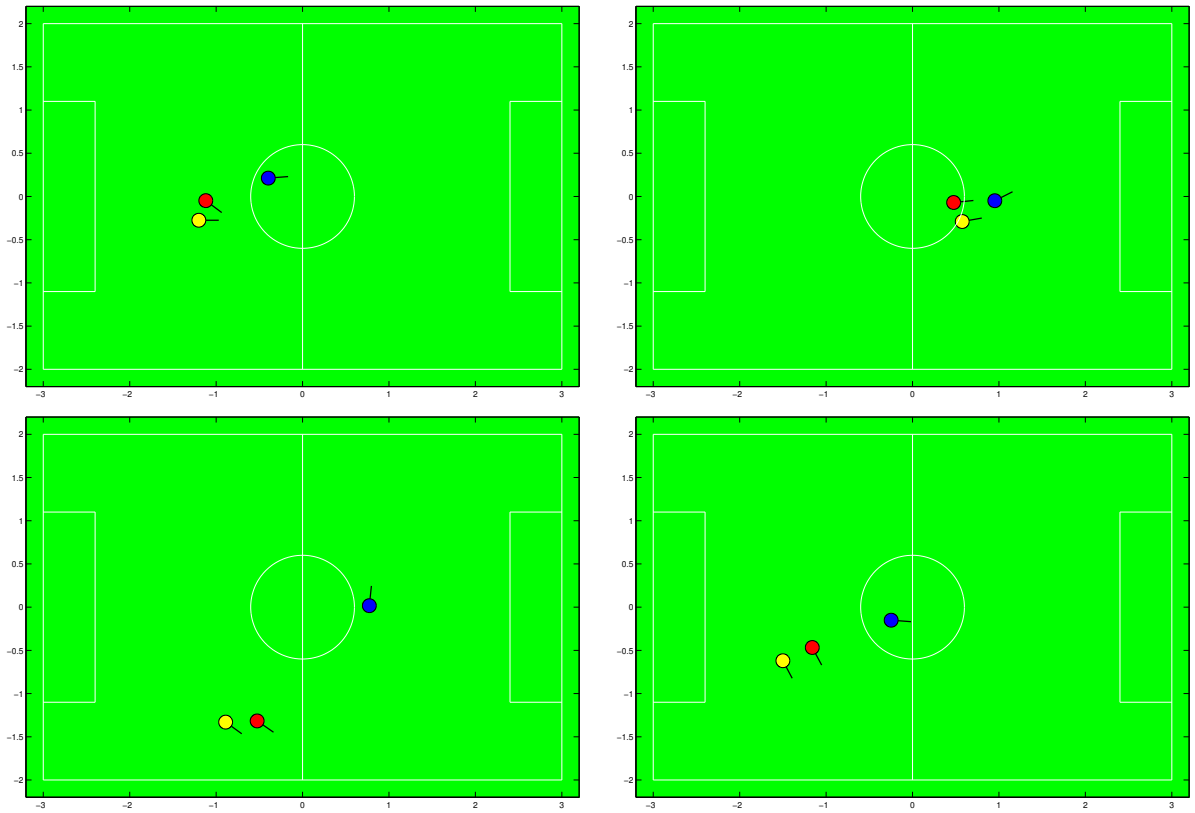


Figure 5.3: Final poses of four representative executions

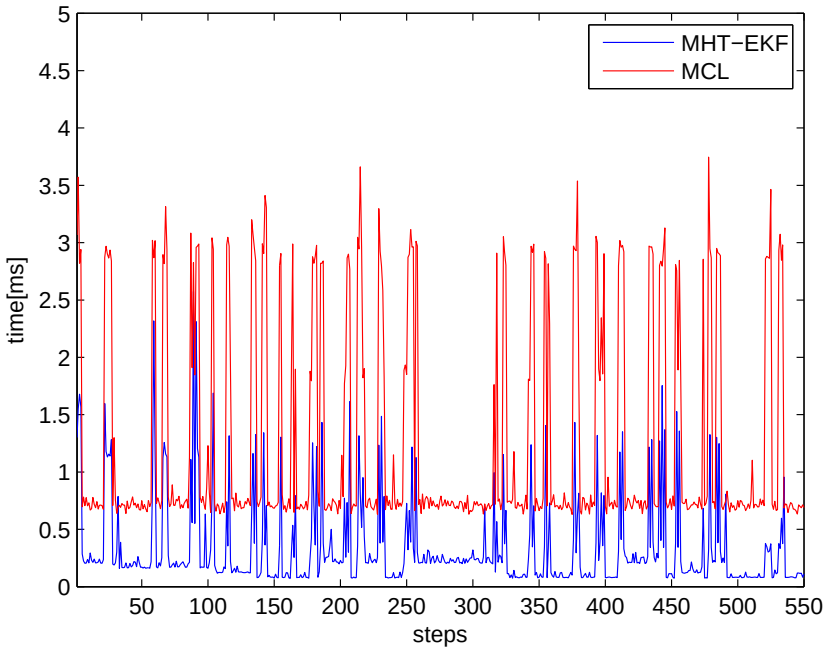



Figure 5.4: Execution times of MHT-EKF and MCL algorithms during a run

the final true and estimated poses, while the MCL algorithm game an average error of 41cm. The average error in orientation was 23 degrees for the MHT-EKF algorithm and 52 degrees for the MCL. Clearly, the estimate of MHT-EKF is much more accurate compared to the estimate of MCL.

## 5.2 Algorithm Performance

Figure 5.4 shows the execution time of our old MCL and the new MHT-EKF algorithms, as measured during a single experiment, where the robot moved around the field localizing itself. The graph shows the execution time of both algorithms over all execution steps of our Monas software architecture during this experiment. The average execution time over all steps was 0.3295ms for the MHT-EKF and 1.2425ms for the MCL algorithm. The spikes shown in the figure correspond to steps where an observation was reported and the update procedure took place. Note that the prediction procedure takes place at all steps.

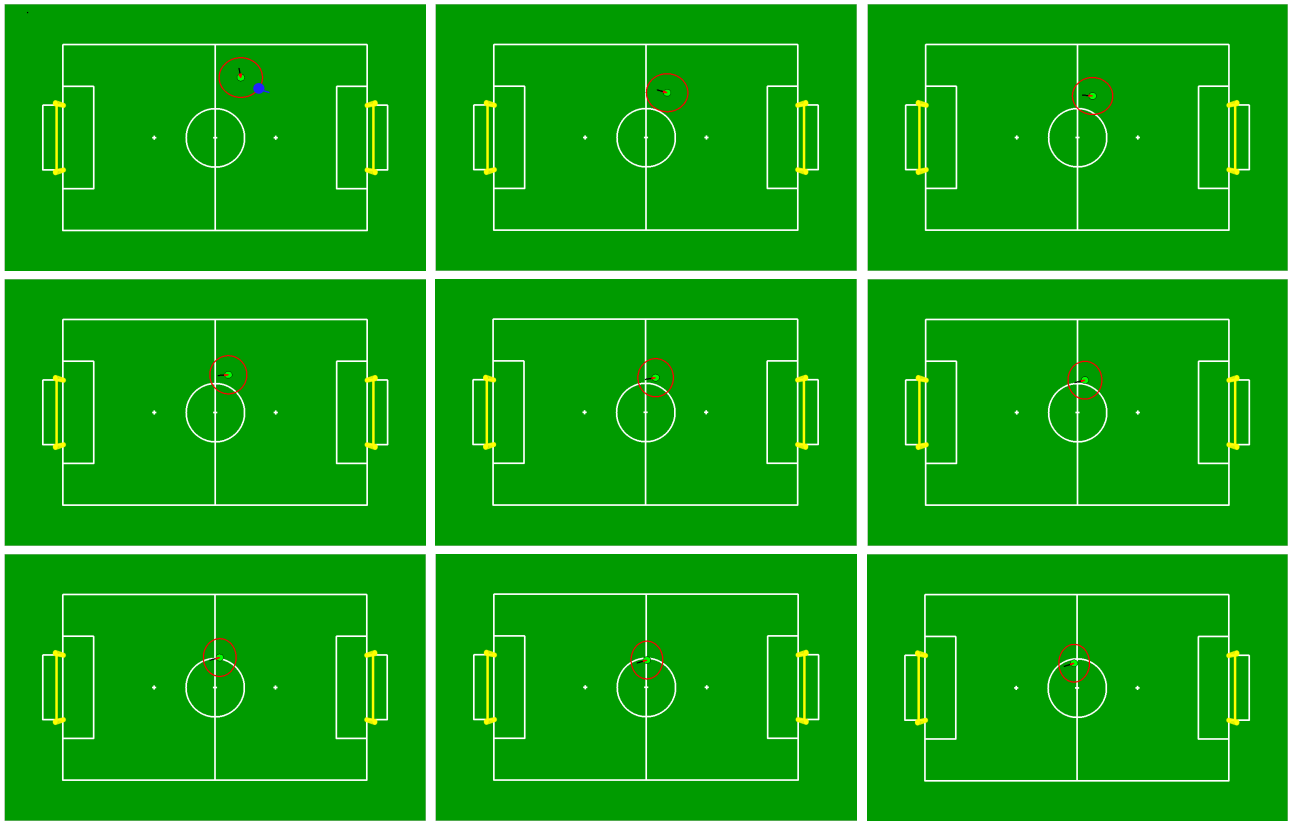


Figure 5.5: Estimated robot pose after 8 update steps with a single-hypothesis EKF

## 5.3 Multi-Hypothesis Tracking Results

To illustrate the effect of Multi-Hypothesis Tracking (MHT) on our implementation a specific experiment was carried out. The robot was placed statically (no move) at a specific pose $(0.8, 1.2, -0.09)$ inside the field (shown in blue in Figures 5.5 and 5.6), but its belief was erroneously initialized to a different pose $(0.5, 1.3, 1.75)$. Then, the robot performed a head scan, acquiring a total of eight observations, which were provided as input to two filters that run simultaneously: a multi-hypotheses EKF limited to four hypotheses and a single-hypothesis EKF, in which the most likely association of observation and landmark is selected (Maximum Likelihood).

Figure 5.5 shows the evolution of the estimated state of the single-hypothesis EKF algorithm, as it was updated with the eight observations. The robot associated the first observation (ambiguous goalpost) to the right post of the goal on left half of the field.

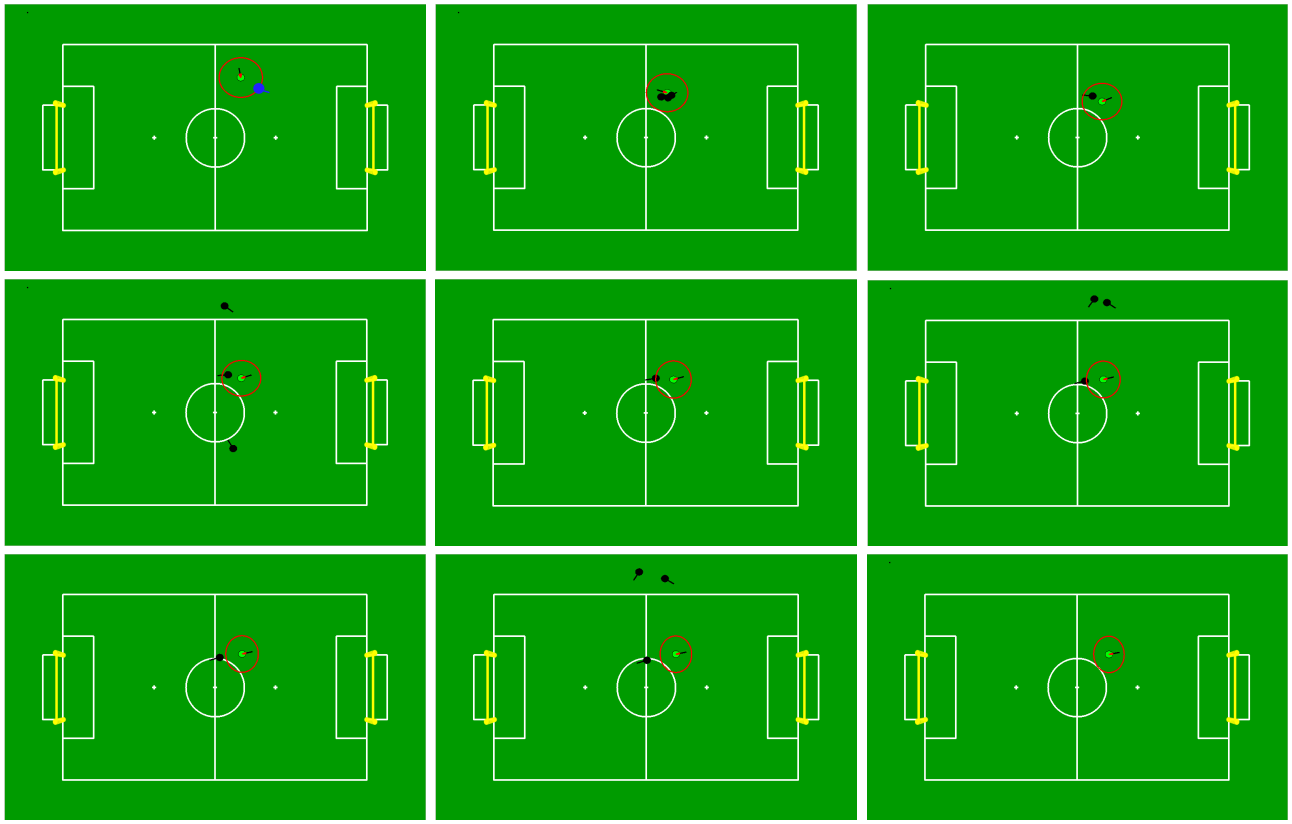


Figure 5.6: Estimated robot pose after 8 update steps with a multi-hypothesis EKF

As a consequence, the pose estimate drifted towards the symmetric pose, after taking into account the remaining seven observations. Figure 5.6 shows the evolution of the estimated state of the multi-hypotheses EKF algorithm, after updates with the eight observations. The circles denote the robot's pose hypotheses; green is the one with the largest weight reported by the localization module. After the wrong association of the first observation (ambiguous goalpost) to the right post of the goal on left half of the field, the algorithm is able to recover and finally estimate a pose close to the true one.

## 5.4 Odometry Calibration Results

To illustrate the effect of the additional state variables on our localization approach, the following experiment was performed. The robot was placed at the center of the field and was given a locomotion command corresponding to a straight-ahead walk towards the goal

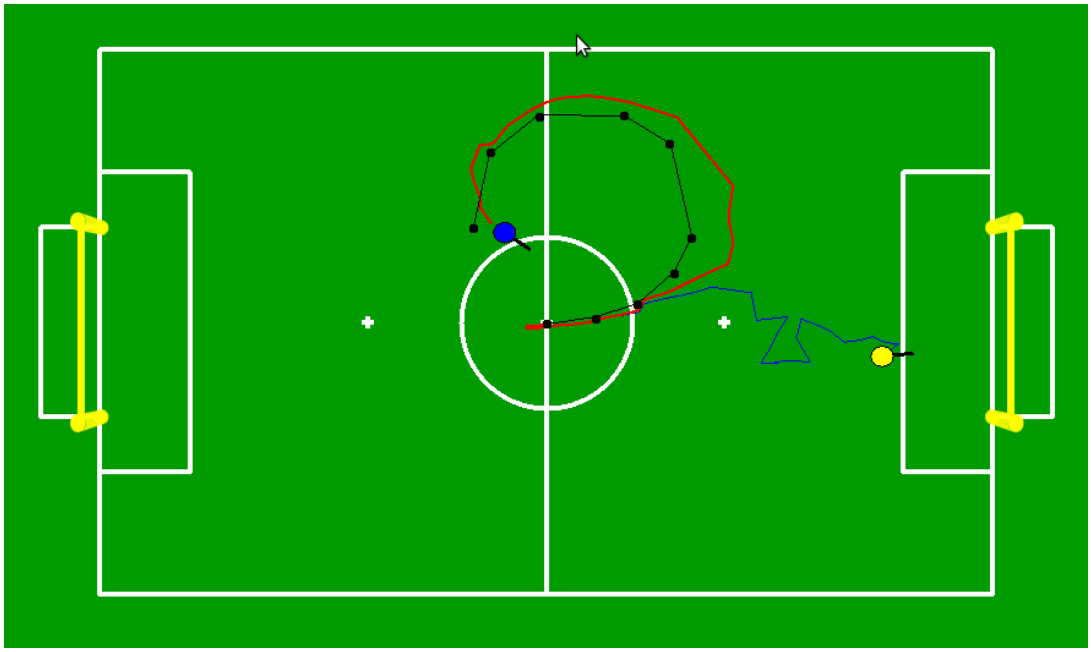


Figure 5.7: Estimated pose traces with and without odometry calibration

on the right side of the field (Figure 5.7). Two variations of our localization algorithm were run in parallel, receiving the same observations and odometry information. The first variation did not account for systematic odometry errors and represented the state of the robot with a three-dimensional vector, as described in Section 4.1. The state of the robot in the second variation was augmented with the three odometry error parameters, as described in Section 4.5. The three parameters $(e_1, e_2, e_3)$ were initialized to $(1, 1, 0)$, implying that initially no odometry error is assumed.

Figure 5.7 shows the estimated robot traces of both localization variations. The robot in reality performed a cyclic path, due to error related to drift, shown with a black dotted line in the figure. The first variation did not estimate the robot pose correctly but instead estimated a path that was consistent (more or less) with the command given, since the odometry is directly extracted from the motion commands and assumes no error. Without some feedback concerning the true trajectory of the robot, the algorithm failed to identify the true motion. Note that the goalpost observations coming from the goal on the left side of the field during the second half of the experiment could not correct the erroneous pose estimate, but only managed to push it back towards the center of the

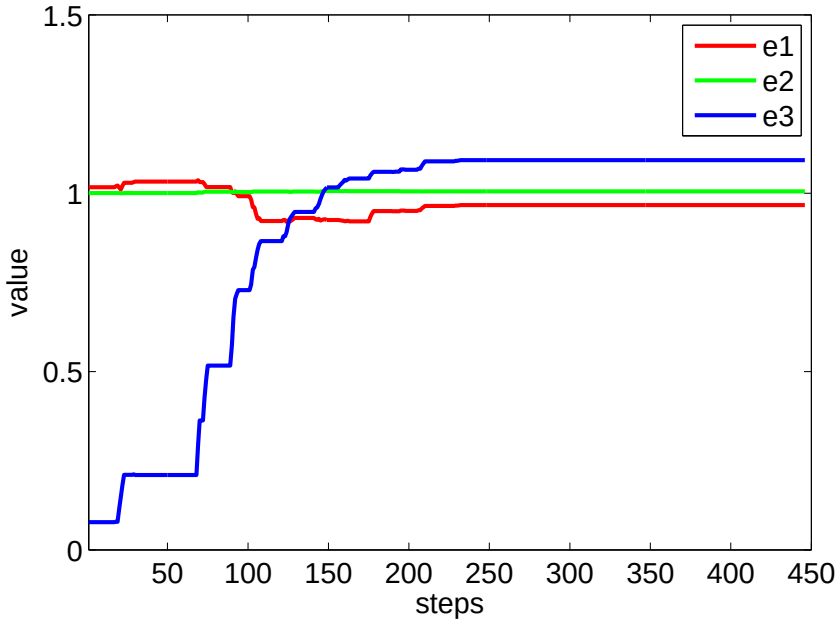



Figure 5.8: Evolution of estimated values of odometry error parameters

field due to the long observation distances. The second variation, in contrast, succeeded in maintaining a correct robot pose at least in the second half of the experiment, as it took some time to estimate the odometry error parameters in the first half. Figure 5.8 shows the evolution of the estimated values of $e_1$, $e_2$, $e_3$ during the entire experiment. Due to the command given to the robot (zero rotational speed) only the third parameter, which is related to drift, has changed significantly from its initial value. Yet this change was crucial in maintaining a correct robot pose estimate.

# Chapter 6

# Conclusion

## 6.1 Conclusion

This thesis describes a new localization approach for the RoboCup SPL field implemented and incorporated into the Monas software architecture of our RoboCup team Kouretes. The localization algorithm takes as input goalpost observations and odometry information and performs local state estimation using extended Kalman filtering. Landmark ambiguities are handled using multi-hypotheses tracking and odometry uncertainty is compensated by estimating the odometry error parameters using an augmented state space. The outcome of the algorithm is used as input to our shared world model activity to produce a global belief about the state of the game which in turn is used by our team coordination activity to determine the roles of the robots. The proposed localization approach gives more accurate results than our previous localization method based on particle filters and is computationally more efficient.

## 6.2 Future Work

### 6.2.1 Additional Landmarks

The use of additional landmarks, such as the center circle or field line corners and T-junctions, can aid significantly the localization task, considering that in the large SPL 2013 field the goals are not always visible. Our localization algorithm can easily be extended to incorporate these new (ambiguous) landmarks, as long as their distance

and bearing is reported when recognized. Note that the inherent ambiguity of these landmarks is automatically handled by our approach. Ongoing work in our team focuses on the visual recognition of such landmarks.

### 6.2.2 Vision System Calibration

In our approach we have used an augmented state space to handle the odometry systematic errors. This technique essentially assumes that observations are correct and therefore can be used to estimate correctly the odometry error parameters. In practice, however, there is significant systematic errors in the observations. It is quite common that the camera of our Nao robots gets displaced by some unknown horizontal and/or vertical angle due to a fall or minor hardware differences. Even small camera displacements have significant effect on the estimation of distance and bearing to visually recognized landmarks. A similar approach (augmented state) to that of calibrating robot odometry could be used to account for observation errors and calibrate the vision system. However, such an approach would require accurate odometry to be able to estimate correctly the vision error parameters. It remains to be seen if both odometry and vision error parameters can be included simultaneously in an augmented state and be estimated correctly when uncertainty is present on both sides.

# References


[1] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: Robocup: A challenge problem for AI. AI Magazine **18**(1) (1997) 73–85 5

[2] RoboCup SPL Technical Committee: RoboCup Standard Platform League (NAO) rule book (2013) Only available online: www.tzi.de/spl/pub/Website/Downloads/Rules2013.pdf. 6, 25

[3] Gouaillier, D., Blazevic, P.: A mechatronic platform, the Aldebaran Robotics humanoid robot. In: Proceedings of the 32nd IEEE Annual Conference on Industrial Electronics (IECON). (2006) 4049–4053 7

[4] Aldebaran Robotics: Nao documentation (2012) Only available online: www.aldebaran-robotics.com/documentation. 8

[5] Paraschos, A.: A flexible software architecture for robotic agents. Diploma thesis, Technical University of Crete, Greece (2010) 11

[6] Orfanoudakis, E.: Reliable object recognition for the RoboCup domain. Diploma thesis, Technical University of Crete, Greece (2011) 12

[7] Pavlakis, N.: Cooperative global game state estimation for the RoboCup standard platform league. Diploma thesis, Technical University of Crete, Greece (2013) 12

[8] Kyranou, I.: Path planning for NAO robots using an egocentric polar occupancy map. Diploma thesis, Technical University of Crete, Greece (2012) 12

[9] Michelioudakis, E.: Dynamic multi-robot coordination for the RoboCup standard platform league. Diploma thesis, Technical University of Crete, Greece (2013) 13

[10] Kofinas, N., Orfanoudakis, E., Lagoudakis, M.G.: Complete analytical inverse kinematics for NAO. In: Proceedings of the 13th International Conference on Autonomous Robot Systems and Competitions (ROBOTICA). (2013) 13

[11] Kofinas, N.: Forward and inverse kinematics for the NAO humanoid robot. Diploma thesis, Technical University of Crete, Greece (2012) 13

[12] Paraschos, A., Spanoudakis, N.I., Lagoudakis, M.G.: Model-driven behavior specification for robotic teams. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2012) 13

[13] Topalidou-Kyniazopoulou, A., Spanoudakis, N.I., Lagoudakis, M.G.: A CASE tool for robot behavior development. In: RoboCup 2012: Robot Soccer World Cup XVI. Volume 7500 of Lecture Notes in Computer Science. Springer (2013) 225–236 13

[14] Topalidou-Kyniazopoulou, A.: A CASE (computer-aided software engineering) tool for robot-team behavior-control development. Diploma thesis, Technical University of Crete, Greece (2012) 13

[15] Vazaios, E.: Narukom: A distributed, cross-platform, transparent communication framework for robotic teams. Diploma thesis, Technical University of Crete, Greece (2010) 14

[16] Karamitrou, M.: KMonitor: global and local state visualization and monitoring for the Robocup SPL league. Diploma thesis, Technical University of Crete, Greece (2012) 14

[17] Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust Monte Carlo localization for mobile robots. Artificial Intelligence **128**(1–2) (2001) 99–141 14, 19

[18] Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005) 17, 20

[19] Gordon, N.J., Salmond, D.J., Smith, A.F.M.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. Radar and Signal Processing, IEE Proceedings F **140**(2) (1993) 107–113 19

[20] Lenser, S., Veloso, M.: Sensor resetting localization for poorly modelled mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE (2000) 20

[21] Thrun, S., Fox, D., Burgard, W.: Monte Carlo localization with mixture proposal distribution. In: Proceedings of the 17th National Conference on Artificial Intelligence (AAAI), MIT Press (2000) 859–865 20

[22] Gutmann, J.S., Fox, D.: An experimental comparison of localization methods continued. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Volume 1. (2002) 454–459 20

[23] Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME – Journal of Basic Engineering **D**(82) (1960) 35–45 20

[24] Welch, G., Bishop, G.: An introduction to the Kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, USA (1995) 21

[25] Julier, S.J., Uhlmann, J.K.: A new extension of the Kalman filter to nonlinear systems. In: Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation, and Controls. (1997) 182–193 23

[26] Barrett, S., Genter, K., Hester, T., Khandelwal, P., Quinlan, M., Stone, P., Sridharan, M.: Austin Villa 2011: Sharing is caring: Better awareness through information sharing. Technical Report UT-AI-TR-12-01, The University of Texas at Austin (2012) 27

[27] Jochmann, G., Kerner, S., Tasse, S., Urbann, O.: Efficient multi-hypotheses unscented Kalman filtering for robust localization. In: RoboCup 2011: Robot Soccer World Cup XV. Volume 7416 of Lecture Notes in Computer Science. Springer (2012) 222–233 28

[28] Quinlan, M.J., Middleton, R.H.: Multiple model Kalman filters: A localization technique for RoboCup soccer. In: RoboCup 2009: Robot Soccer World Cup XIII. Volume 5949 of Lecture Notes in Computer Science. Springer (2010) 276–287 28

[29] Meissner, M., Friedrich, H., Fürtig, A., Weis, T., Siegl, J.M., Becker, C., Michalski, V., Ruscher, G., Kehlenbach, A., Reckers, A., Zacharias, K., Hanssen, E., Heun, A.: Bembelbots Frankfurt team description for RoboCup 2013. Technical report, Goethe University, Germany (2013) 28

[30] Chang, C.H., Wang, S.C., Wang, C.C.: Vision-based cooperative simultaneous localization and tracking. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2011) 5191–5197 28

[31] Anderson, P., Hunter, Y., Jeffrey, Hengst, B.: An ICP inspired inverse sensor model with unknown data association. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2013) 28

[32] Laue, T., Röfer, T.: Particle filter-based state estimation in a competitive and uncertain environment. In: Proceedings of the 6th International Workshop on Embedded Systems. (2007) 29

[33] Laue, T., Röfer, T.: Pose extraction from sample sets in robot self-localization – a comparison and a novel approach. In: Proceedings of the 4th European Conference on Mobile Robots. (2009) 283–288 29

[34] Burchardt, A., Laue, T., Röfer, T.: Optimizing particle filter parameters for self-localization. In: RoboCup 2010: Robot Soccer World Cup XIV. Volume 6556 of Lecture Notes in Computer Science. Springer (2011) 145–156 29

[35] Gohring, D., Mellmann, H., Burkhard, H.D.: Constraint based world modeling in mobile robotics. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2009) 2538–2543 29

[36] Martín, F., Agüero, C.E., Cañas, J.M.: Extended kalman filter populations for a reliable real-time robot self-localization. In: Proceedings of the IEEE Intelligent Vehicles Symposium - Workshop on Perception in Robotics. (2012) 29

[37] Roumeliotis, S., Bekey, G.A.: Bayesian estimation and Kalman filtering: a unified framework for mobile robot localization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Volume 3. (2000) 2985–2992 37

[38] Salmond, D.J.: Mixture reduction algorithms for target tracking in clutter. In: Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets. Volume 1305. (1990) 434–445 42

[39] Martinelli, A., Siegwart, R.: Estimating the odometry error of a mobile robot during navigation. In: Proceedings of the European Conference on Mobile Robots (ECMR). (2003) 44