Fakultät für Informatik
der Technischen Universität München
Lehrstuhl Prof. Dr. B. Radig

Vision-based Probabilistic State Estimation for Cooperating Autonomous Robots

Thorsten Schmitt

Fakultät für Informatik
der Technischen Universität München
Lehrstuhl Prof. Dr. B. Radig

Vision-based Probabilistic State Estimation for Cooperating Autonomous Robots

Thorsten Schmitt

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. rer. nat. R. Westermann
Prüfer der Dissertation:		
	1.	Univ.-Prof. Dr. rer. nat. B. Radig
	2.	Univ.-Prof. Dr. rer. nat. W. Burgard,
		Albert-Ludwigs-Universität Freiburg

Die Dissertation wurde am 27. 11. 2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 03. 05. 2004 angenommen.

to the memory of

Dr. Stefan Lanser

∗ 11.02.1967
† 21.01.1999

Abstract

With the services that autonomous robots are to provide becoming more demanding, the states that the robots have to estimate become more complex. In this thesis, a vision-based, probabilistic state estimation method for large and complex states is developed and applied to autonomous mobile robot applications.

The proposed method extends the state-of-the-art in state estimation in two important ways. First, it demonstrates how the estimation of states in complex and ill structured state spaces, spanned by more than 60 parameters, can be achieved. The state estimation problem is made feasible by decomposing it into several loosely coupled subproblems. Each subproblem is solved by a task specific state estimator. In particular, this method enables the mobile robots of a team to estimate their own positions in a known environment and to track the positions of independently moving objects at frame rate.

The second contribution is the investigation of the use of cooperation, i.e. the exchange of observations and state estimates between robots, for improved state estimation. The state estimators of the robots are extended to use the information provided by other robots as evidence. This information is shown to increase the accuracy, reliability and completeness of the state estimation process. In particular, it is demonstrated that cooperation enables robots to determine their poses and the positions of further dynamic objects more accurately, to track temporarily occluded objects successfully, and to obtain a complete view of the surrounding environment.

The method is empirically validated based on experiments with a team of autonomous robots equipped with off-the-shelf computing hardware and sensory equipment within the RoboCup scenario. It was applied during four RoboCup world championships. The collected experimental data, from two competitions covering more than four hours net operation time, is analysed.

Keywords: autonomous mobile robot, Bayes filter, computer vision, cooperative state estimation, Kalman filter, multiple hypothesis tracking, multi-robot system, obstacle tracking, pose tracking, RoboCup, robot soccer, sensor fusion, vision-based localisation.

Zusammenfassung

Damit autonome mobile Serviceroboter ihre immer komplexer werdenden Serviceaufgaben erfolgreich bearbeiten können, müssen sie den Zustand ihrer Umwelt immer detaillierter und genauer wahrnehmen. Diese Dissertation entwickelt ein bildbasiertes probabilistisches Verfahren zur Schätzung hochdimensionaler Zustände und setzt es auf realen autonomen mobilen Roboterplattformen ein.

Das vorgeschlagene Verfahren erweitert den Stand der Technik in zwei wichtigen Bereichen. Zum einen wird demonstriert wie Zustände in Zustandsräumen mit mehr als 60 Dimensionen geschätzt werden können. Die Komplexität des Schätzproblems wird dadurch reduziert, dass es in mehrere lose verbundene Teilprobleme zerlegt wird. Jedes Teilproblem wird dann durch einen spezifischen Zustandsschätzer gelöst. Insbesondere ermöglicht dieses Verfahren einem Team mobiler Roboter ihre eigenen Positionen und die anderer sich unabhängig bewegender Roboter zu bestimmen.

Zum anderen gelingt der Nachweis, dass kooperative Zustandsschätzung durch mehrerer Roboter die Qualität und Vollständigkeit der geschätzten Zustände erheblich verbessert. Die Zustandsschätzer der einzelnen Roboter werden dafür so erweitert, dass sie Informationen anderer Roboter mit berücksichtigen können. Weiterhin wird gezeigt, dass kooperative Zustandsschätzung dem Roboterteam genauere Positionsschätzungen sowie das Verfolgen temporär verdeckter Objekte ermöglicht.

Das Verfahren wurde erfolgreich an einem Team autonomer mobiler Roboter im Rahmen des Roboterfussballszenarios (RoboCup) getestet. Erschwert wurde dies durch die Ausstattung des Roboterteams, welches nur über handelsübliche Rechenleistung und Videosensorik verfügte. Das Verfahren wurde im Rahmen von vier RoboCup Weltmeisterschaften erfolgreich eingesetzt. Zur grundlegenden Evaluierung des Verfahren wurden experimentelle Daten aus zwei Weltmeisterschaften mit insgesamt mehr als vier Stunden autonomer Spielzeit ausgewertet.

Keywords: Autonome mobile Roboter, Bayes Filter, Bildverstehen, Kooperative Zustsandsschätzung, Kalman Filter, Multiples Hypothesen tracking, RoboCup, Roboterfussball, Sensor Fusion, Bildbasierte Lokalisation.

Acknowledgements

The venture of managing a middle size RoboCup team, successfully participating at several RoboCup world championships, and writing a PhD thesis at the same time would not have been possible without the help of many very enthusiastic and devoted students, colleagues, and friends. Together we endured times of hardship when we prepared our robots for a tournament and when we participated at a RoboCup tournament. But the fun of getting things to work, travelling to distant places on this planet, working for several days without sleep, meeting and getting to know people from all over the world compensated our efforts more than we could have expected.

It is a pleasure to acknowledge the efforts for our team of every single human team member. In alphabetical order the human team members of the years 1998 to 2002 were

RoboCup-Team 1998: Kagan Aksit, Thorsten Bandlow, Marc Grimme, Markus Hofbauer, Iganz Kellerer, Michael Klupsch, Stefan Lanser, Fabian Schwarzer and Christoph Zierl.

RoboCup-Team 1999: Thorsten Bandlow, Daniel Frey, Robert Hanek, Michael Klupsch, Petra Räbel, Michael Schilp and Mark Wenig.

RoboCup-Team 2000: Matthias Budil, Sebastian Buck, Max Fischer, Robert Hanek, Michael Klupsch, Arnim Kreutzer, Petra Räbel and Mark Wenig.

RoboCup-Team 2001: Michael Beetz, Sebastian Buck, Max Fischer, Andreas Hofhauser, Jacoppo Rubbia.

RoboCup-Team 2002: Michael Beetz, Andreas Bombe, Sebastian Buck, Lorenzo Capecchi, Max Fischer, Andreas Hofhauser, Dirk Neumann, Kathrin Scheidemann, Derik Schröter and Freek Stulp.

It is difficult to list the contributions of every single team member and you will have to forgive me if I miss out on one or the other.

First of all, I would like to emphasise my colleague and friend Robert Hanek. Robert, you have supported me in every single phase of this thesis. In endless discussions you have helped me to sort out and focus my thoughts. Many of your ideas have been coded for the robot system and found their way into this thesis. It is sad, that we have not yet succeeded in setting up our own company, but the future has yet to come!

Michael Beetz has had a great impact on my research and on this thesis. Michael, you understood very well how to guide me and to publish the achieved results in the scientific community. You have also read the draft of this thesis several times and suggested numerous improvements. It has been a great pleasure for me working together with you and I could not have had a better advisor than you!

When I entered the human RoboCup team in 1998, a well designed and perfectly working multi-robot research platform was handed over to me. I would like to credit the team leaders of the 1998 RoboCup team, Christoph Zierl, Michael Klupsch, Fabian Schwarzer and Thorsten Bandlow, with this achievement. You have laid the foundation stone for *The AGILO RoboCuppers*, their later achievements and my thesis. Michael, without your continuous support, even after you had left our research group, we would not have got that far!

Throughout the years the robots underwent a huge number of mechanical changes. This would not have been possible without the support of a great hardware crew. The help of the *Rechnerbetriebsgruppe (RBG)* of the Munich University of Technology is greatfully acknowledged. In particular, Richard Haensch, Thomas Engels and Franz Öttl have spent hours with great enthusiasm on the redesign of the robots' mechanics. The first kicking device for the robots was designed by Michael Schilp (IWB, Munich University of Technology). It has been improved and perfectionised by Richard Haensch and has been built several times by the company Scheidemann Klebetechnik GmbH. Maximilian Fischer had a great impact on the redesign of the robots and their electronical features in 2002. Our system administrator Oliver Bösl supported us by solving several difficult administration and network problems. Thanks a lot for your continuous support!

The AGILO RoboCuppers have been sponsored and supported by several innovative and leading edge high tech companies. The most generous include RadioLAN (1999), MVTec (1999 - 2004), 4Soft (2001) and Siemens mobile (2002 - 2004). The research reported in this thesis has been sponsored by the Deutsche Forschungsgemeinschaft (DFG) under contract Ra-359/71 ("Semi-automatic acquisition of visuomotoric plans"). All funds are gratefully acknowledged.

I would like to thank Ingemar J. Cox and Matthew Miller of the NEC Research Institute for providing their wonderful and well designed Multiple Hypothesis Tracking implementation and reprints of their journal papers. It is sad, that we have never met, but I am looking forward to getting to know you guys one day!

My gratitude is also devoted towards Prof. Dr. Bernd Radig, for giving me the opportunity to work in such an exciting and interesting field of research and to work on this thesis. I would like to thank Prof. Dr. Bernd Radig for the confidence he has put in me, the possibility to lead a RoboCup team for such a long time, and the enthusiasm with which he has supported Robert's and my ideas towards an entrepreneurship.

Furthermore, I would like to express my thanks to Prof. Dr. Wolfram Burgard for being the second reporter of this thesis. I am sorry, that we have not yet carried out the proposed comparisson of the SJPADF and CODT algorithms based on data sets for multiple people and robot tracking. Research in the RoboCup domain and the maintainance of a middle size robocup team are very demanding and time consumeing tasks. I hope we can still perform this research after this thesis has been handed in!

I own a great debt to my colleagues at the research group for image understanding and knowledge based systems of the Munich University of Technology. The presence and company of you all made the time we have spent together very enjoyable. All the best to your futures!

Bea Horvath has read the draft of this thesis several times and suggested numerous linguistic and punctuation improvements. Bea, thanks a lot for help and your persistent efforts!

Finally, I would like to express my gratitude to my wife Katrin. She has enjoyed the good times and endured the bad times with me that are associated with scientific work. I hope, I can compensate her for the many lost hours in the future.

Freek and Suat[1], you are now inheriting a working multi-robot platform and a huge chunk of legacy code. It will not be easy to familiarise yourselves with the whole lot and there will be times (hopefully not too often) where you are going to curse the robots and us (previous human RoboCuppers). But overall I am sure you will enjoy your time with the robots and all other opportunities associated with RoboCup, as much as I have. Good luck, I will keep my fingers crossed for you and will watch the performance of *The AGILO RoboCuppers* in the coming years!

Thank you all!

Munich, November 2003 Thorsten

[1]Freek Stulp and Suat Gedikli are now the team leaders of *The AGILO RoboCuppers*.

Contents

List of Figures

List of Tables

Glossary of Notation

Constants and Scalars

C_x, C_y	Principal point - coordinates of center of radial lens distortion
f	Focus - effective focal length of the pin hole camera model
p	Weight of a state
$r_\alpha, r_\beta, r_\gamma$	Rotation angles
S_x, S_y	Scale factors
κ	Kappa - 1st order radial lens distortion
ν	Normalising constant

Distributions

$Bel(.), Bel'(.)$	Belief state
$N(.)$	Normal distribution

Matrices

K	Kalman filter gain
$\mathcal{R}$	Rotation matrix
$\mathcal{R}_x$	Rotation matrix about x-axis
$\mathcal{R}_y$	Rotation matrix about y-axis
$\mathcal{R}_z$	Rotation matrix about z-axis
$\Sigma_\mathbf{x}$	Covariance of a Gaussian random variable x
$\Sigma_\mathbf{s}$	Kalman filter innovation

Operators

a	Action selection model
A	Linearised action selection model
e	Inverse measurement model
E	Linearised inverse measurement model
f, g	System model
F, G	Linearised system model
h	Measurement model
H	Linearised measurement model
V	Linearised process noise
W	Linearised measurement noise
∇	Jacobi matrix (linearisation) of a model
$\mathbf{H}(.)$	Hessian matrix
$det(.)$	Determinant of a matrix
$\lvert.\rvert$	Modulus of an expression, cardinality of a set
$\lVert.\rVert$	Norm or length of a vector

Sets

H	Set of hypotheses (MHT)
S	Set of samples (Monte Carlo Bayes Filter)
Z	Set of measurements (MHT)

Superscripts and Subscripts

$x^{(i)}$	Element i of a vector or set
x^{-1}	Inverse
x^T	Transpose
x_t	Discrete time index t
$x_{t_1 \dots t_2}$	Set with elements $x_{t_1}, x_{t_1+1}, \dots, x_{t_2-1}, x_{t_2}$
x_i	Element i of a vector or set
x_{rc}	Element of a matrix row r, column c
$\widetilde{x}$	Predicted state or object hypothesis
$\widehat{x}$	Estimate of a state or object hypothesis
$\bar{\mathbf{x}}$	Mean of Gaussian random variable x
$\Sigma_{\mathbf{x}}$	Covariance of a Gaussian random variable x

Vectors

$data$	Dynamic system data (action or measurement)
h	Object hypothesis (MHT)
u	Dynamic system action
v	Dynamic system process noise
x	State
z	Dynamic system measurement
w	Dynamic system measurement noise
$\mathcal{T}_c$	Translation vector

Chapter 1

Introduction

In many applications, autonomous robots have to know and assess the states of themselves and their environments in order to choose their actions competently. Contrary to these need, the information what robots receive through their sensors is inherently uncertain: typically, the robot's sensors can only perceive parts of their environments, and their sensor measurements are inaccurate and noisy. In addition, control over their actuators is also inaccurate and unreliable. Finally, the dynamics of many environments cannot be accurately modelled and sometimes environments change too fast or even nondeterministically.

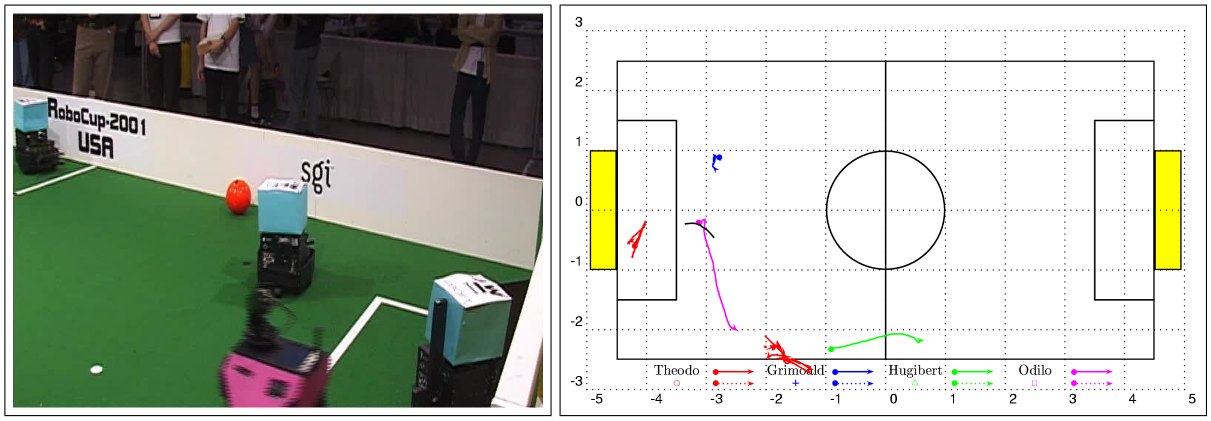
This dissertation explores fundamental issues in estimating the states of complex and dynamic environments for a team of autonomous mobile robots. In particular, the problem of vision-based state estimation, i.e. self-localisation as well as the detection and tracking of dynamic objects, is investigated. A successful realisation of this process has to address several sub-problems.

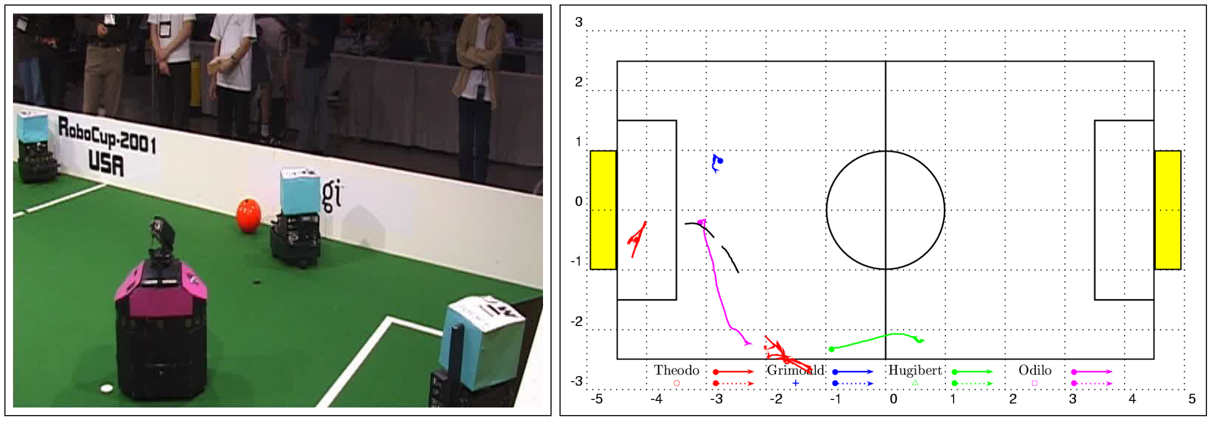
1.1 Background and Motivation

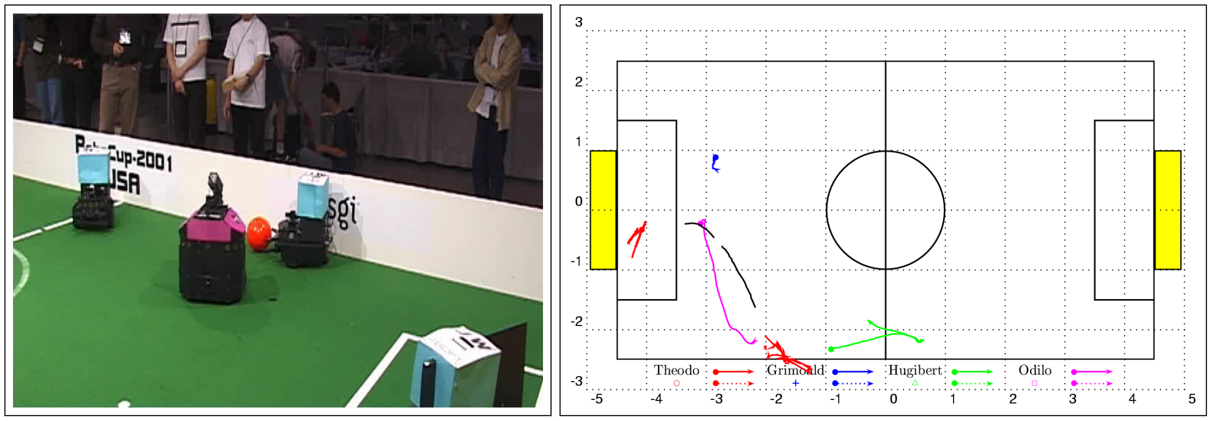
Autonomous robot soccer is considered to be the primary application domain for the techniques developed in this thesis. This scenario is very challenging in the respect, that the states of several dynamic objects have to be estimated simultaneously.

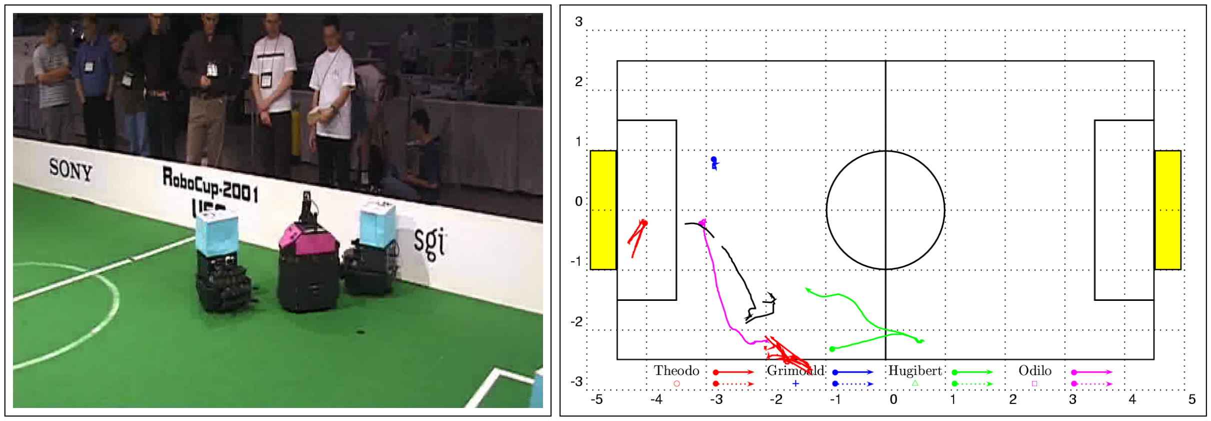
In the robot soccer middle size league two teams of four autonomous robots play soccer against each other. Both teams have the intention to win the game and, consequently, try to get possession of the ball and score goals. A state estimator for competent robotic soccer players should provide the action selection routines with estimates of the positions and this may even be the dynamic states of all players (teammates and opponents) and the ball.

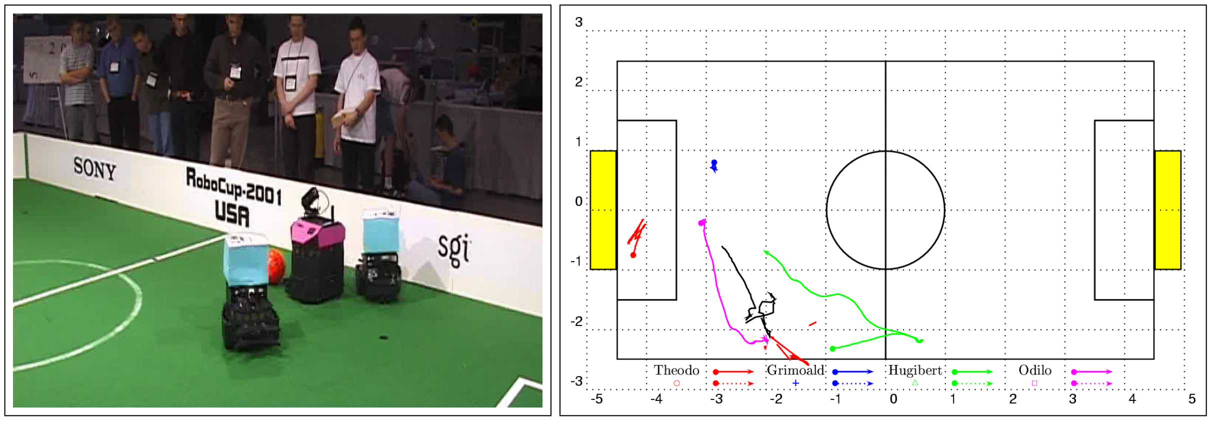
Figures 1.1(a) to (m) (on the next page) depict a scene taken form a match at the RoboCup World Championship 2001 in Seattle between *The Ulm Sparrows* (magenta marker) and *The AGILO RoboCuppers* (cyan marker). The goals of *The Ulm Sparrows* and *The AGILO RoboCuppers* are to the left and to the right of the images, respectively. The column to the left displays the current match situation. The column to the right visualises the estimated game state of *The AGILO RoboCuppers*. The trajectories of the AGILO robots are displayed with different colours. Trajectories of opponent robots and the ball are black and red, respectively. The time interval between two consecutive rows is approximately two seconds.

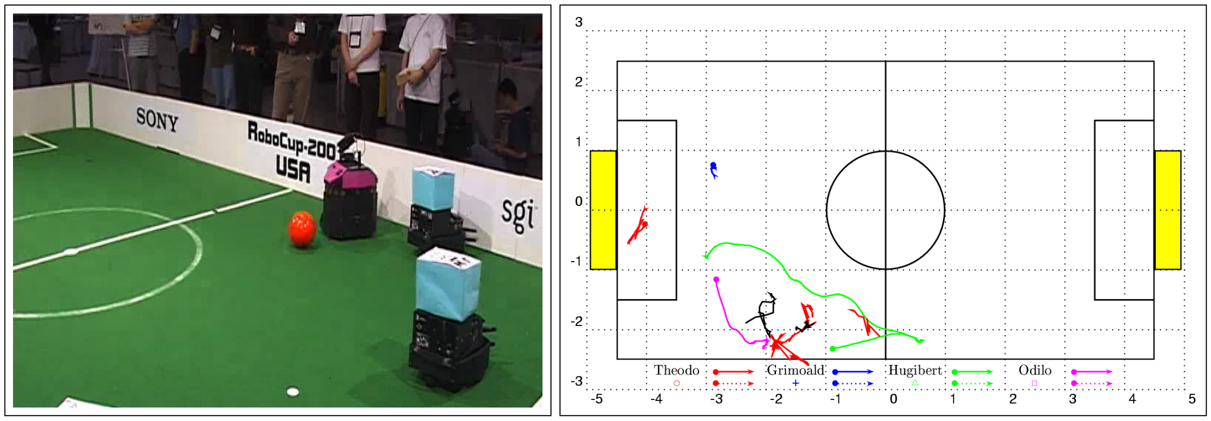
(a)

RoboCup-2001
USA
sgi
Theodo
Grimoald
Hugibert
Odilo

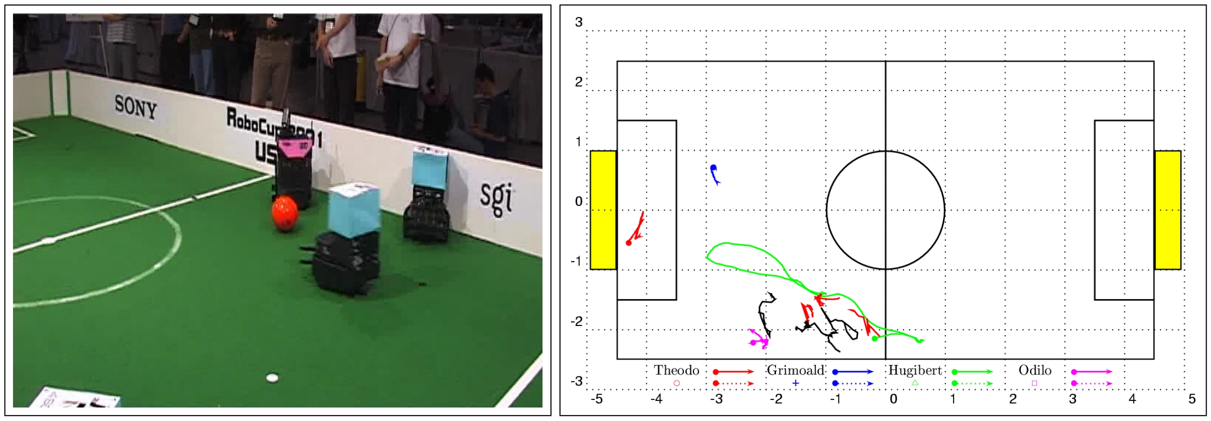
(b)

RoboCup-2001
USA
sgi
Theodo
Grimoald
Hugibert
Odilo

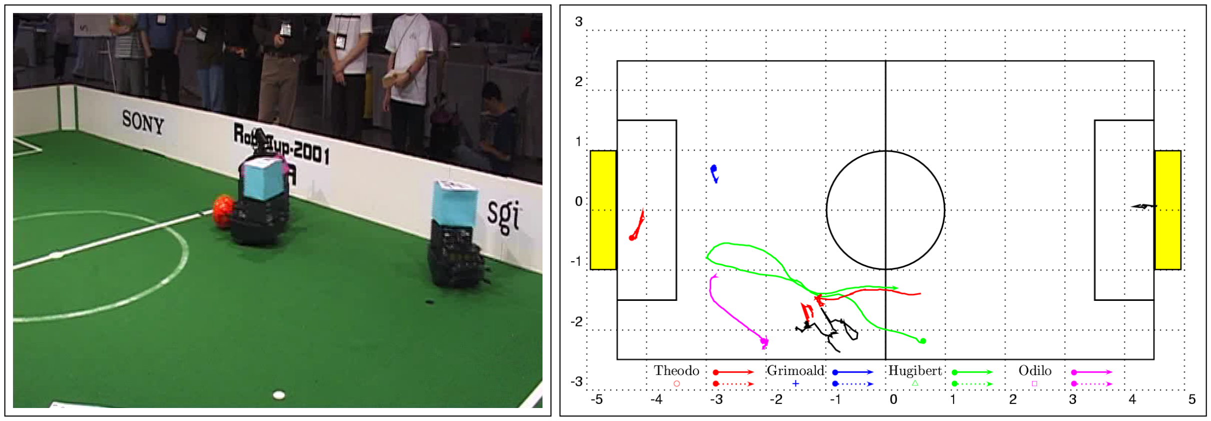
(c)

RoboCup-2001
USA
sgi
Theodo
Grimoald
Hugibert
Odilo

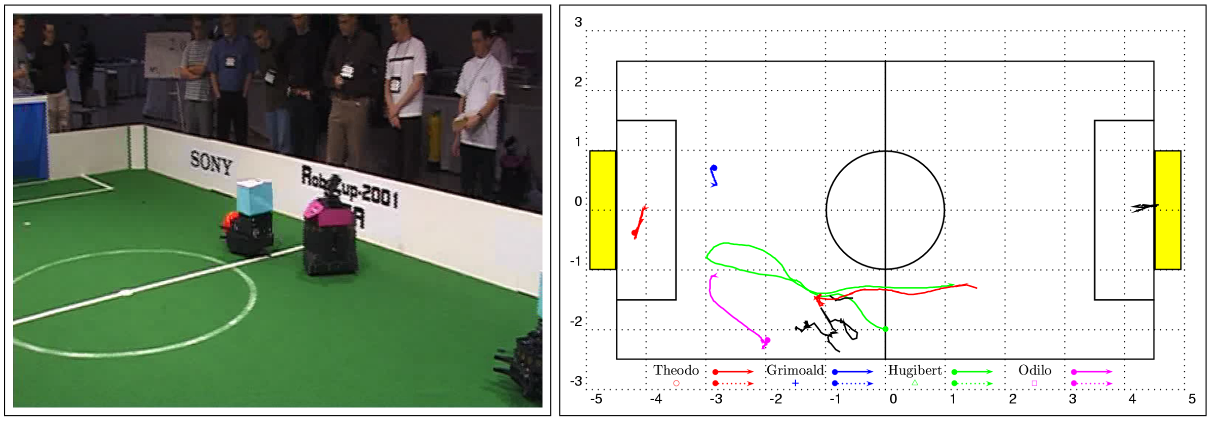
(d)

SONY
RoboCup-2001
sgi
Theodo
Grimoald
Hugibert
Odilo


SONY
RoboCup-2001
USA
sgi
Theodo
Grimoald
Hugibert
Odilo
(e)

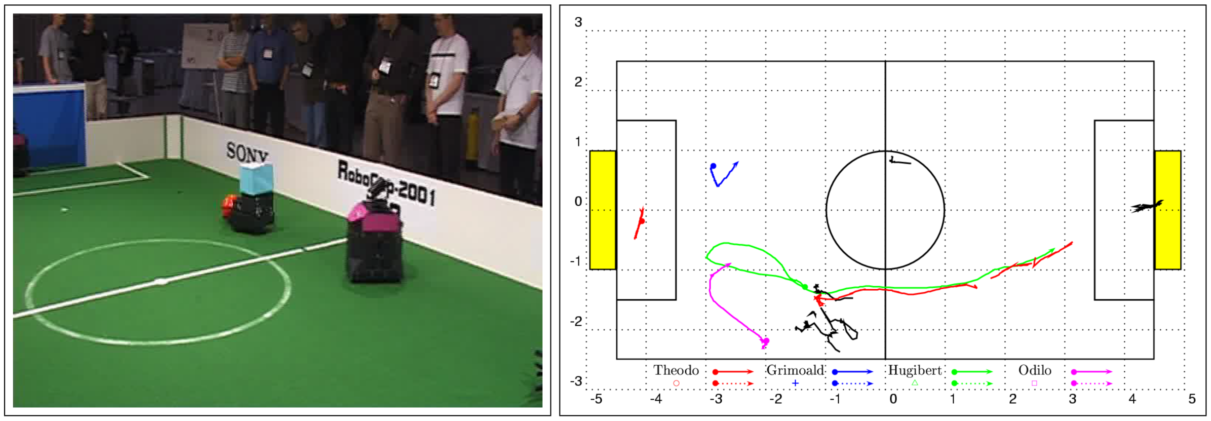

SONY
RoboCup-2001
USA
sgi
Theodo
Grimoald
Hugibert
Odilo
(f)


SONY
sgi
Theodo
Grimoald
Hugibert
Odilo
(g)

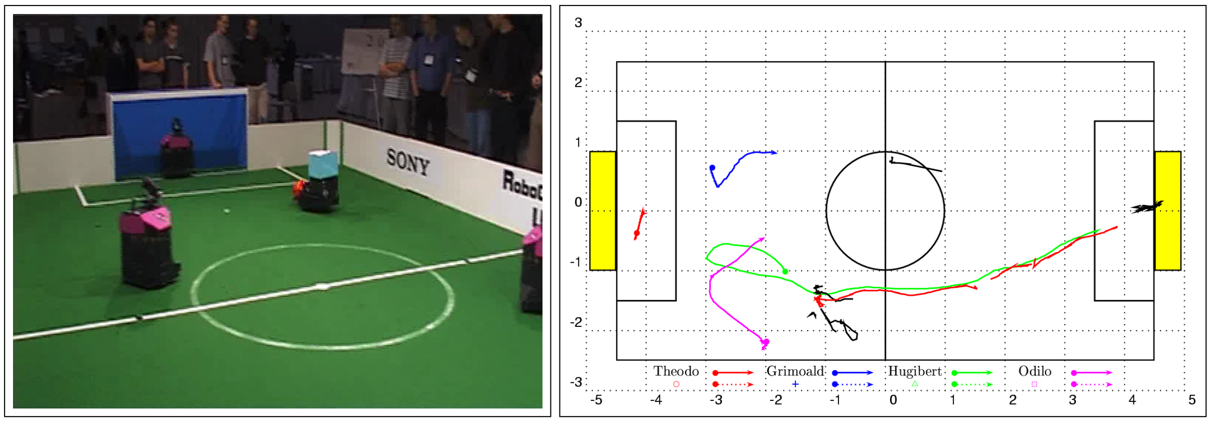

SONY
RoboCup-2001
sgi
Theodo
Grimoald
Hugibert
Odilo
(h)


SONY
RoboCup-2001
Theodo
Grimoald
Hugibert
Odilo

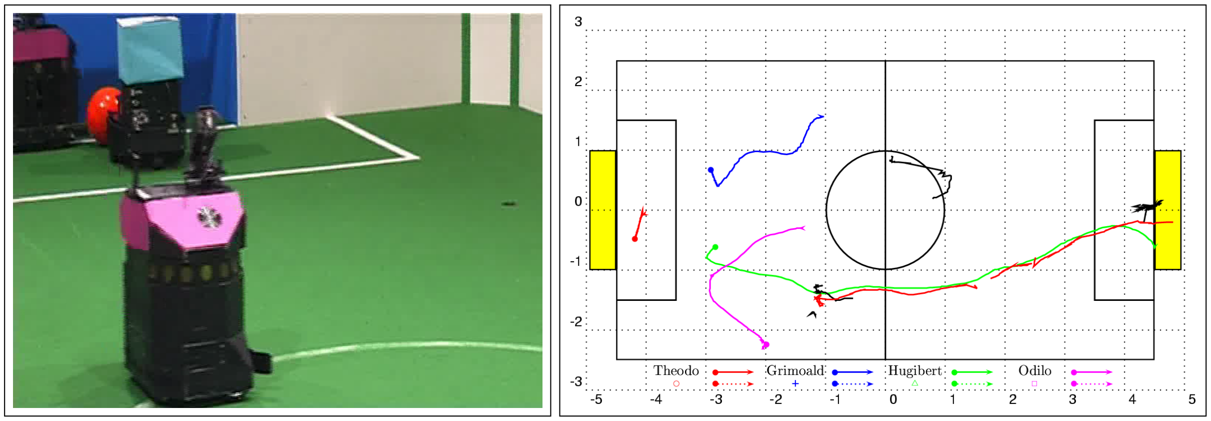
(i)


SONY
RoboCup-2001
Theodo
Grimoald
Hugibert
Odilo

(k)


SONY
Theodo
Grimoald
Hugibert
Odilo

(l)


Theodo
Grimoald
Hugibert
Odilo

(m)

In Figure 1.1(a), two robots of the *The AGILO RoboCuppers* are facing the ball. They have to decide which one of them will go for it. If both robots approached the ball, it would be very likely that they got entangled and a robot from *The Ulm Sparrows* would get possession of the ball. As a consequence, the robot that can reach the ball faster[1] decides to go for it, whereas the second robot decides to move to a defending position. Figures 1.1(b) to (e) illustrate this behaviour. The first robot moves towards the ball and tries to shuffle it away from the wall. This requires precise knowledge about the ball's position. By the time the first robot has achieved this goal, a striker from *The Ulm Sparrows* blocks its path. The robot then decides to avoid this obstacle and to push back. In the meantime, the second robot has also successfully detected the opponent player and navigates around it. In order to accomplish this task successfully, both robots must have the capability to detect the opponent player and generate an estimate of its position.

As a result of its obstacle avoiding manoeuvre, the first robot gets stuck at the wall. Figures 1.1(f) and (g) depict how it is trying to get away from the wall. Even though it does not see the ball, it knows its exact position which is estimated by the second player and communicated via a wireless connection link to all teammates. The second AGILO player recognises that its teammate cannot actively intervene and decides to go for the ball. It moves towards the ball and successfully dribbles it around the opponent player (see Figures 1.1(h) to (j)) towards the goal of *The Ulm Sparrows*. Meanwhile, the other player moves towards a defending position.

Finally, the robot dribbling the ball towards the opponent's goal has to make the crucial decision whether it is trying to score past the left or the right side of the goalkeeper (see Figures 1.1(k) to (l)). In order to preserve a maximum chance of scoring, the AGILO player is aiming for the right goal post. This allows it to kick the ball into both corners. If the goalkeeper drives towards the right corner, a short turn and a kick into the left corner is sufficient to score. On the other hand, if the goalkeeper decides to stay in the middle of the goal or to move to the left, then a shot into the right corner is adequate. Eventually, the goalkeeper stays in the middle of the goal and the striker scores into the right corner.

In order to exhibit the football playing behaviour described in the example above, the perception mechanisms of the robots must provide them with several kinds of information: (1) the position and orientation of the robot itself, (2) the position and orientation of the teammates, (3) the position of the ball, and (4) the positions of the opponent robots. Determining the positions of opponent robots is particularly difficult, since the robots look similar and exhibit only very few features that allow for an exclusive identification. Furthermore, the robots and the ball may be taken out of the field and may be put in again later.

A system for state estimation in autonomous robot soccer should address the following aspects.

- *State Estimation and Environment Modelling* - Autonomous robots must have sufficient and accurate information about themselves and their environments to complete their tasks competently. Contrary to these needs, the information that robots receive through their sensors is inherently uncertain: typically the robots' sensors can only access parts

[1]Faster, in this context, refers to the time that a robot requires to approach the ball in order to effectively being able to dribble the ball towards the opponents goal. In certain situations a robot has to drive around the ball and approach it from behind.

of their environments and their sensor measurements are inaccurate and noisy.

- *Cooperative Perception* - Teams of robots, equipped with communication devices, can estimate the same state cooperatively, i.e. robots can provide their own state estimates and observations as evidence for the teammates. Through the fusion of information from different observers, the uncertainty of the state estimates can be reduced. Furthermore, information observed by only one member of the team can be used by others for better decision making, action selection, and path planning. In particular, cooperative perception enables a team of robots to track temporarily occluded objects and to faster recover their position after they have lost track of it.

In Section 1.2, the problem addressed in this thesis is described in the context of the constraints imposed on any proposed solution. In Section 1.3, the proposed approach and the components of the solution are described. In Section 1.4, the principal contributions of the thesis are summarised, and in Section 1.5, an outline of the thesis is presented.

1.2 Problem Summary

The general problem addressed in this thesis is the development of a vision-based cooperative game state estimation system for a team of autonomous soccer robots (see Figure 1.2). Given four video streams, a team of autonomous robots has to estimate a joint game state. This game state contains all the estimates necessary for role and action selection, such as the pose estimates of the teams robots, the position of the ball and the positions of the opponents.

This system must be able to cope with dynamic and fast changing environments. The robots might travel at speeds up to 2 m per second and must be able to recognise static landmarks and dynamic objects, determine their own poses relatively to a given model of the playing field, track dynamic objects and determine their movement directions and velocities. All of this information is derived from (1) a video stream, captured by the robot's onboard video camera, with 30 frames per second and (2) all the observations made by members of the team which are broadcast through a wireless communication channel. Off-the-shelf computing hardware is used for communication, I/O and processing of the sensor data. This requires new and carefully chosen algorithmic techniques in order to achieve the required accuracy, robustness towards noise and real time performance. The basic functional breakdown of the system under development is the following:

1. A robot starts at an unknown position in a predefined global coordinate frame. The system will refer all position measurements from the sensor relative to this global coordinate frame.

2. The robot must estimate it's own state, i.e. localise itself, and the states of the ball and the opponent players. The estimated states are then broadcast to all members of the team. State estimates received from other team members are fused with the own states and the robot's model of the world is updated.

3. On the basis of this world model, path planning is performed and a new action is chosen. Finally, a motor command is selected and sent to the motor controller.

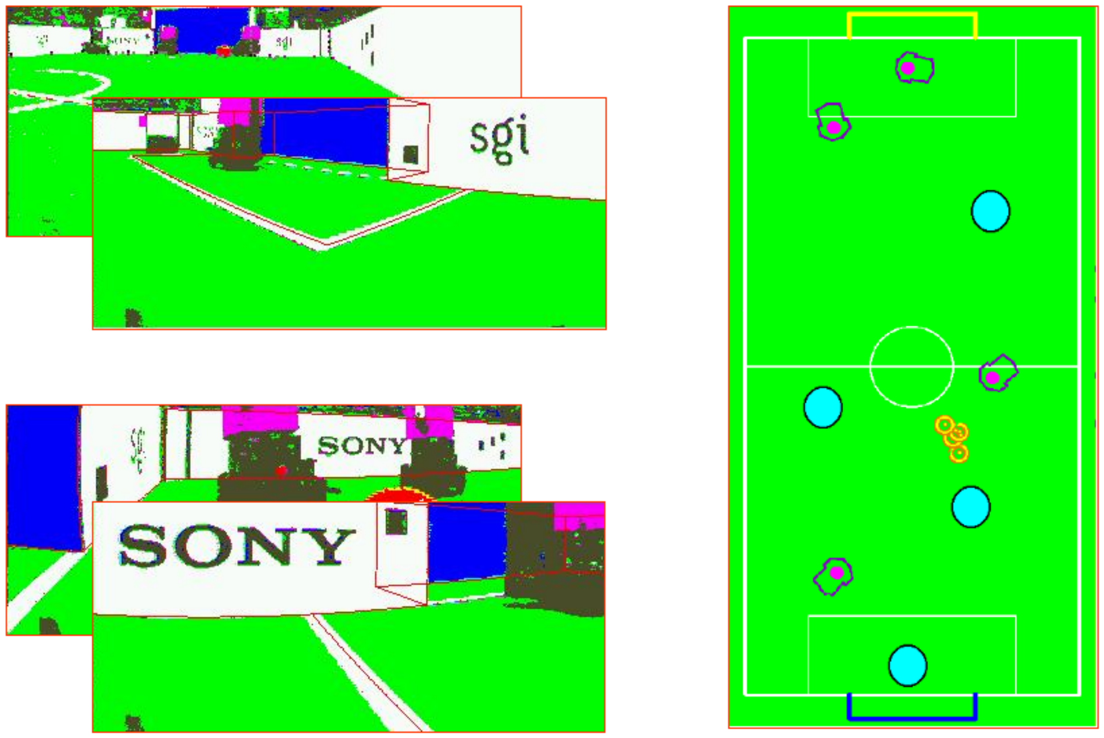


Figure 1.2: Given four different views of the field a team of autonomous robots has to estimate a joint game state.

The apparent simplicity of these three tasks is deceiving. For example, any measurement taken by a sensor has some degrees of imprecision that must be represented. Applying a non-linear transformation to such a measurement, e.g. transform from sensor centred coordinates to global coordinates, this will affect the precision or accuracy of the position estimate in a way that may be difficult to compute. Once the measurement has been transformed to global coordinates, determining its origin from possibly millions of features may be very difficult, ambiguous and computationally intensive. Finally, a mechanism must be applied to use the imprecise measurement and to update the pose estimates of the robot and the game state. The system also has to deal with several sensors, performing observations at various points in time and with different observation frequencies, e.g. odometric and image data are received at 10 and 30 Hz, respectively. Furthermore, communication channels between robots may be unreliable and packets may be delayed, lost, or incomplete. Some principal constraints that must be satisfied by the system include:

- *Statistically meaningful representation of uncertainty.* - Observations and pose estimates are always associated with a degree of uncertainty. Particularly, the uncertainty of observations may influence the uncertainty introduced into the robot's position estimates. Furthermore, during the process of global self-localisation, ambiguous situations must be represented. Usually, they are resolved by a sequence of observations at different locations.

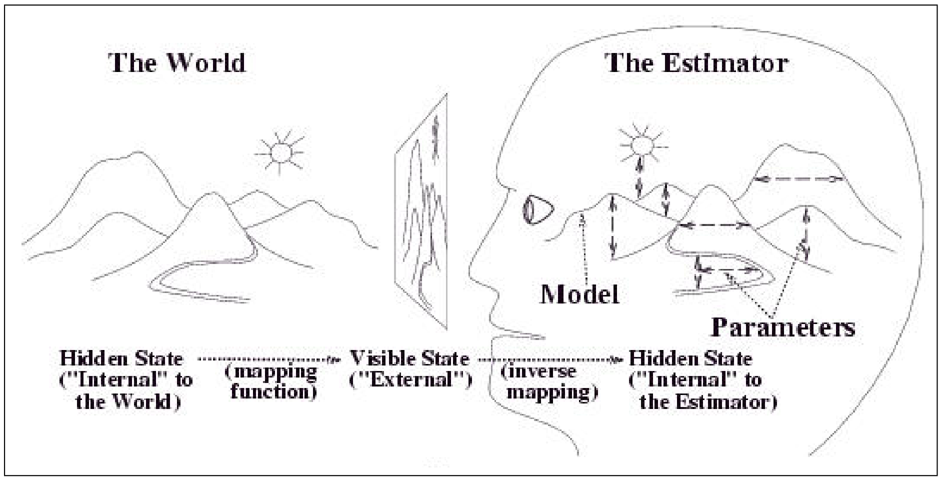

The World
The Estimator
Model
Parameters
Hidden State ("Internal" to the World)
(mapping function)
Visible State ("External")
(inverse mapping)
Hidden State ("Internal" to the Estimator)

Figure 1.3: The general state estimation problem, taken from Rao (1997).

- *Real time processing of sensor data.* - Robots operating in highly dynamic environments require fast sensor data processing algorithms in order to react to environment changes in an appropriate way. Disproportionately long processing times may result in unsatisfactory damage to both the environment and the robot. Moreover, a robot may be equipped with several sensors in order to perceive more of its environment.

- *Efficiently scalable.* - The system must be scalable in the number of robots and sensors belonging to the team, and in the number of dynamic objects that can efficiently be tracked.

In the next section, the approach and the key techniques are identified which must be addressed in order to create a system that satisfies the above constraints.

1.3 The Approach

Successful state estimation is a key requirement for autonomous robot deployment. The general state estimation problem is depicted in Figure 1.3. The world is observed by an observer. For simplicity, this observer is called estimator in the following. The sensing device used by the estimator is limited. It has to obey several physical constraints, such as maximum resolution, lens distortions and a limited field of view, and thus can only perceive a partial and incomplete view of the world. The virtual function, mapping the world to sensor data, is defined by the type of sensor used. This mapping function is commonly referred to as sensor model. The estimator is now confronted with the problem of perceiving the external world with the help of an inverse sensor model and an internal model of the world. The estimator does not have access to the hidden internal states of the world that are causing its sensory experiences. Instead, in order to correctly interpret and understand the external world, it must solve the "inverse" problem of estimating these hidden state parameters using only the sensory data obtained from its various sensing devices.

Figure 1.4 adapts the general state estimation problem to an autonomous robot and illustrates this process with the help of the dynamic system model[2]. The state of a continuously changing world is observed by the robot through a sensor. Observations are acquired in the form of sensor data according to an unknown sensor model. The state estimation process uses an approximation of the inverse sensor model to estimate the world's state, which is also called the belief state (Aström, 1965) at times. This state usually includes the robot's pose[3] and the states of all other objects of interest to the robot. For example, a game state[4], s_t, in the robot soccer domain comprises the following variables:

$$\begin{array}{rcl}
\mathit{Robot}_t^1 & = & (x_t^1, y_t^1, \phi_t^1, \dot{x}_t^1, \dot{y}_t^1, \dot{\phi}_t^1) \\
\mathit{Robot}_t^2 & = & (x_t^2, y_t^2, \phi_t^2, \dot{x}_t^2, \dot{y}_t^2, \dot{\phi}_t^2) \\
\mathit{Robot}_t^3 & = & (x_t^3, y_t^3, \phi_t^3, \dot{x}_t^3, \dot{y}_t^3, \dot{\phi}_t^3) \\
\mathit{Robot}_t^4 & = & (x_t^4, y_t^4, \phi_t^4, \dot{x}_t^4, \dot{y}_t^4, \dot{\phi}_t^4) \\
\mathit{Ball}_t & = & (x_t, y_t, \dot{x}_t, \dot{y}_t) \\
\mathit{Opponent}_t^1 & = & (x_t^1, y_t^1, \dot{x}_t^1, \dot{y}_t^1) \\
\vdots & = & \vdots \\
\mathit{Opponent}_t^n & = & (x_t^n, y_t^n, \dot{x}_t^n, \dot{y}_t^n)
\end{array}$$

$$s_t = (\mathit{Robot}_t^1, \ \ldots \ , \mathit{Robot}_t^4, \ \ldots \ , \mathit{Ball}_t, \mathit{Opponent}_t^1, \ \ldots \ , \mathit{Opponent}_t^n) \tag{1.1}$$

$\dot{x}, \dot{y}$ and $\dot{\phi}$ denote the translational and rotational velocities, respectively. This game state is used as input to the action selection and path planning routines. These procedures decide which action and path the robot is selecting. As a result, these routines determine a motor control command which is executed by the robot's motors. The execution of motor commands change the state of the world and the whole process is repeated.

The main difficulty with state estimation is that in general the estimated states are inherently uncertain. Uncertainty arises from sensor limitations, noise, approximated and simplified sensor and internal world models and the fact that most interesting environments are to "a certain degree" uncertain and unpredictable. Consequently, a good state estimation technique must not only provide estimates of the world's state, but also an estimate of the uncertainty associated with this state.

One particularly promising method for accomplishing this task is *probabilistic state estimation* (Thrun, 2000). Recent longterm experiments with autonomous robots (Thrun et al., 2000; Burgard et al., 2000; Simmons et al., 1997) have shown that an impressively high level of reliability and autonomy can be reached by explicitly representing and maintaining the uncertainty inherent in the available information.

Probabilistic state estimators maintain the probability densities for the states of objects over time. The probability density of an object's state, conditioned on the sensor measurements received so far, contains all the information which is available about an object for a

[2]An introduction to the dynamic system model and a more formal definition can be found in Section 3.2

[3]The term pose, in this thesis, refers to a robot's $x - y$-coordinates together with its heading direction ϕ.

[4]This state description is somewhat idealistic. In practice, the translational and rotational velocities of the ball and the opponent robots are quite difficult to determine.

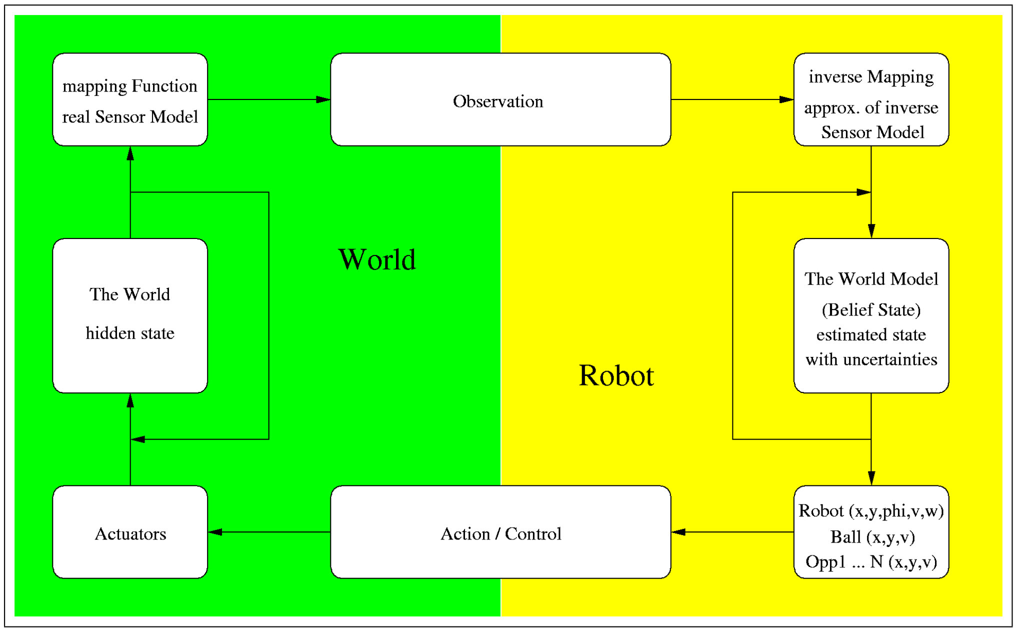


Figure 1.4: State estimation with an autonomous robot.

robot. Based on these densities, robots are not only able to determine their own and other objects most likely state, but can also derive even more meaningful statistics, such as the variance of the current estimate, and derive an idea about the quality (e.g. accuracy) of the estimate. As a result, a probabilistic robot can gracefully recover from errors, can handle ambiguities, and can integrate noisy sensor data in a consistent way. Moreover, a probabilistic robot knows about its own ignorance, a key prerequisite of truly autonomous robots.

Approached probabilistically, the state estimation problem is a density estimation problem, where a robot seeks to estimate a posterior distribution over the space of its poses and the poses of other objects conditioned on the available data. Denoting the game state at time t by s_t, and the data received up to time t by $data_t$, the posterior is conveniently written as

$$p(s_t|data_{0...t}) \tag{1.2}$$

For brevity, this posterior will be denoted by $Bel(s_t)$, and referred to as the robot's belief state (Aström, 1965) at time t.

Building a single competitive state estimator for robot soccer is not feasible. Thus, two simplifications assumptions about the probability distributions are made in order to reduce the complexity of the estimation problem:

1. To estimate the desired belief state, $Bel(s_t)$, probabilistic approaches frequently resort to a Markov assumption, which states that the past is independent of the future given knowledge of the current state. The Markov assumption is often referred to as the static world assumption, since it assumes that the robot's pose is the only state in the world that would impact more than just one isolated sensor reading. Practical experience suggests, however, that probabilistic algorithms are robust to mild violations of the

Markov assumption, and extensions exist that go beyond this assumption (Fox et al., 1998).

2. The game state estimator is decomposed into several different estimation modules which use different estimation techniques. Consequently, the estimation problem is reduced to estimating the following belief states:

$$Bel(Robot_t^1),\ \ldots\ , Bel(Robot_t^4)$$
$$Bel(Ball_t)$$
$$Bel(Opponent_t^1),\ \ldots\ , Bel(Opponent_t^n)$$

In robotic soccer the four core components of a probabilistic state estimation system for game state estimation and robot control are the following:

1. *A framework for the integration of probabilistic state estimation modules.* - Several different modules that perform different tasks, work together in the proposed probabilistic state estimation system. They resolve feature ambiguities and contradictions in order to estimate one global state of the environment. The framework is able to handle, load, unload, start, stop, and reparameterise several different kinds of modules. It assures that data among modules are exchanged in time and that data may be broadcast over a wireless communication link.

2. *A global self-localisation algorithm.* - The static features, extracted by the feature extraction algorithms, are used to determine a rough pose estimate. During this process of global self-localisation, the algorithm must be capable of representing ambiguous situations. These ambiguities are then resolved in a fast manner by a sequence of observations at different poses.

3. *An iterative self-localisation algorithm.* - Once a rough pose of the robot is uniquely determined, refining and tracking this pose with a local self-localisation algorithm can be achieved more efficiently. Localisation also includes the ability to detect failures of the pose estimation and to relocalise the robot, when necessary. This requires to run the global self-localisation algorithm again.

4. *An object detection and tracking technique.* - When a tracking process involves multiple targets, it is necessary to identify the subset of tracked objects - or tracks - with which each observation (dynamic feature) is feasibly associated. It is usually impractical to enumerate all pairs of observations and tracked objects. Therefore, an efficient gating mechanism is required, which avoids most infeasible associations without discarding any potentially feasible ones. From the set of tracks which are feasibly associated with a given observation, it is necessary to determine which observation/track pair should be maintained as updated track. It is assumed that subsequent observations resolve the ambiguity before the proliferation of hypotheses becomes excessive.

Although previous researchers have examined the described approach and some of these components before, no system exists that incorporates all these techniques into one robust vision-based system. Furthermore, the techniques are extended in such a way, that they can be applied to a multi-robot system, allowing for cooperative perception and cooperative state estimation.

1.4 Principal Contributions

In this thesis, theory and techniques are used from several diverse areas of study, including computer vision, statistics, state estimation and multiple target tracking, to study the basic technical challenges in cooperative large scale state estimation. Several important contributions are made towards both the theoretical foundations and the practical solution of cooperative vision-based state estimation, and more generally, to the areas of localisation and object tracking. In particular the following:

1. *Estimation of Large Scale and Complex States* - The approach to state estimation that is presented in this thesis is able to solve large scale and complex state estimation problems in the robot soccer domain. The estimated state includes, the pose of the own robots, the position of the ball, and the positions of the opponent players. The overall state consists of a vector with more than 60 parameters. In order to reduce the overall complexity of the estimation problem, it is decomposed into several loosely coupled subproblems. Each subproblem is solved by a task specific state estimator employing a specific state estimation technique. The state estimators interact in order to derive more accurate and complete game state estimates. This is made possible by a compact and uniform representation of uncertainties, consisting of a mean value and a covariance matrix, that can concisely be communicated between and easily be processed by the state estimation algorithms. This approach allows a team of robots to make use of all available information and to perform vision-based self-localisation, to track the ball, and to track the opponent robots at a frame rate of 30 Hz with off-the-shelf computing hardware and sensory equipment.

2. *Vision-based Self-Localisation, Ball- and, Opponent-Tracking* - A fast and accurate algorithm for model driven vision-based self- and ball-localisation is presented. The proposed algorithm is capable of estimating the pose of a robot in 3D (2D position and 1D orientation) or 6D (3D position and 3D orientation) and can be employed on various robotic platforms. Fast response times are achieved through the use of a known environment model consisting of curve features, an accurate and universal model of an image sensor, and an extremely efficient feature projection and extraction process. A particular feature of this algorithm is its capability to uniquely solve ambiguous localisation problems through the use of teammate observations of dynamic objects with known shapes. Furthermore, an efficient algorithm for vision-based object detection and cooperative tracking is presented. This algorithm is especially suited for mobile sensors with uncertain position. The observations of dynamic objects with unknown shape performed by all members of the robot team are integrated into one single view of the world.

3. *Cooperative Perception Enhances the State Estimates* - It is demonstrated that cooperative perception enhances the accuracy, completeness and reliability of the estimated states. This evidence is derived from a series of real world experiments, with a physical team of robots in real match situations. Using ground truth data, provided by an overhead camera system it is shown that cooperative state estimation increases the accuracy as well as the coverage of ball and opponent tracks both substantially and significantly.

1.5 Outline

The following is a synopsis of the chapters of this thesis:

Chapter 2 introduces notation as well as some fundamental preliminaries. Particularly, rotations in 3D Cartesian space, Gaussian normal distributions, distance measures for Gaussian random variables, variants for the linear and nonlinear transformation of a Gaussian probability density function are discussed. The chapter concludes with the presentation of the pinhole camera model which is a fundamental building block for this thesis.

Chapter 3 presents probabilistic state estimation as a formal framework for state estimation in dynamical systems. The Bayes filter is introduced and several variants along with their properties, strengths, weaknesses and applicability for the purpose of self-localisation and multiple object tracking are examined and discussed.

Chapter 4 introduces the autonomous robot soccer benchmark and presents the research platform, *The AGILO RoboCuppers*, used for the experimental evaluation of the algorithms proposed in this thesis. The hardware setup, as well as the software architecture of the robot controller, its components and the interactions among them are outlined.

Chapter 5 describes the extremely fast and accurate vision-based self- and ball-localisation procedure, called the Cooperative Incremental Iterative Self-localisation (CIIL) algorithm. The data structures of the algorithm, its computational principles and extensions for cooperative perception are presented.

Chapter 6 presents the Cooperative Object Detection and Tracking (CODT) algorithm used to track the opponent players. The data structures of the algorithm, its computational principles and extensions for cooperative perception are developed.

Chapter 7 evaluates the proposed states estimation algorithms CIIL and CODT in several real world experiments. Collected experimental data from two RoboCup world championships, covering more than four hours net operation time, is presented and analysed.

Chapter 8 summarises the contributions of this thesis and sketches further directions for research.

Chapter 2

Preliminaries

2.1 Introduction

This chapter introduces notation as well as some fundamental preliminaries. Particularly, rotations in 3D Cartesian space, Gaussian normally distributed random variables, distance measures for Gaussian random variables, i.e. the Mahalanobis and Bhattacharyya distance, variants for the linear and non-linear transformation of a Gaussian random variables are discussed. The chapter concludes with the presentation of the pinhole camera model which constitutes is a fundamental building block for this thesis. Readers familiar with these concepts may skip this chapter and should proceed with the next chapter.

2.2 Representation of Rotations in 3D Space

A relatively intuitive representation of a rotation is the encoding of the rotation with three angles. $(r_\alpha, r_\beta, r_\gamma)$, each of them describing a rotation about a given axis,

$$\begin{aligned}
\mathcal{R}_x(r_\alpha) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(r_\alpha) & -\sin(r_\alpha) \\ 0 & \sin(r_\alpha) & \cos(r_\alpha) \end{pmatrix} \\
\mathcal{R}_y(r_\beta) &= \begin{pmatrix} \cos(r_\beta) & 0 & \sin(r_\beta) \\ 0 & 1 & 0 \\ -\sin(r_\beta) & 0 & \cos(r_\beta) \end{pmatrix} \\
\mathcal{R}_z(r_\gamma) &= \begin{pmatrix} \cos(r_\gamma) & -\sin(r_\gamma) & 0 \\ \sin(r_\gamma) & \cos(r_\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
\mathcal{R} &= \mathcal{R}_x(r_\alpha) \cdot \mathcal{R}_y(r_\beta) \cdot \mathcal{R}_z(r_\gamma)
\end{aligned}$$

i.e. the rotation is expressed through a rotation about the Z-axis (angle r_γ), followed by a rotation about the new Y-axis (angle r_β) and a rotation about the new X-axis (angle r_α). The rotation matrix $\mathcal{R}$ is composed as follows:

$$
\begin{aligned}
\mathcal{R} &= \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \\
r_{11} &= \cos(r_\beta)\cos(r_\gamma) \\
r_{12} &= -\cos(r_\beta)\sin(r_\gamma) \\
r_{13} &= \sin(r_\beta) \\
r_{21} &= \sin(r_\alpha)\sin(r_\beta)\cos(r_\gamma) + \cos(r_\alpha)\sin(r_\gamma) \\
r_{22} &= -\sin(r_\alpha)\sin(r_\beta)\sin(r_\gamma) + \cos(r_\alpha)\cos(r_\gamma) \\
r_{23} &= -\sin(r_\alpha)\cos(r_\beta) \\
r_{31} &= -\cos(r_\alpha)\sin(r_\beta)\cos(r_\gamma) + \sin(r_\alpha)\sin(r_\gamma) \\
r_{32} &= \cos(r_\alpha)\sin(r_\beta)\sin(r_\gamma) + \sin(r_\alpha)\cos(r_\gamma) \\
r_{33} &= \cos(r_\alpha)\cos(r_\beta)
\end{aligned} \tag{2.1}
$$

This is only one of several possibilities to describe a rotation with three rotation angles. A commonly used representation is also the application of the rotation angles in reverse order:

$$
\begin{aligned}
\mathcal{R} &= \mathcal{R}_z(r_\theta)\cdot\mathcal{R}_y(r_\vartheta)\cdot\mathcal{R}_x(r_\phi) \\
r_{11} &= \cos(r_\theta)\cos(r_\vartheta) \\
r_{12} &= \cos(r_\theta)\sin(r_\vartheta)\sin(r_\phi) - \sin(r_\theta)\cos(r_\phi) \\
r_{13} &= \cos(r_\theta)\sin(r_\vartheta)\cos(r_\phi) + \sin(r_\theta)\sin(r_\phi) \\
r_{21} &= \sin(r_\theta)\cos(r_\vartheta) \\
r_{22} &= \sin(r_\theta)\sin(r_\vartheta)\sin(r_\phi) + \cos(r_\theta)\cos(r_\phi) \\
r_{23} &= \sin(r_\theta)\sin(r_\vartheta)\cos(r_\phi) - \cos(r_\theta)\sin(r_\phi) \\
r_{31} &= -\sin(r_\vartheta) \\
r_{32} &= \cos(r_\vartheta)\sin(r_\phi) \\
r_{33} &= \cos(r_\vartheta)\cos(r_\phi)
\end{aligned} \tag{2.2}
$$

The rotation angles $(r_\alpha, r_\beta, r_\gamma)$ and $(r_\theta, r_\vartheta, r_\phi)$ can be derived from $\mathcal{R}$ as follows:

$$
\begin{aligned}
\cos(r_\beta) &= \sqrt{r_{11}^2 + r_{12}^2} \\
r_\alpha &= atan2\left(-\frac{r_{23}}{\cos(r_\beta)}, \frac{r_{33}}{\cos(r_\beta)}\right) \\
r_\beta &= atan2\left(r_{13}, \cos(r_\beta)\right) \\
r_\gamma &= atan2\left(-\frac{r_{12}}{\cos(r_\beta)}, \frac{r_{11}}{\cos(r_\beta)}\right)
\end{aligned} \tag{2.3}
$$

respectively,

$$\begin{aligned}
\cos(r_\vartheta) &= \sqrt{r_{11}^2 + r_{21}^2} \\
r_\theta &= atan2\left(\frac{r_{21}}{\cos(r_\vartheta)}, \frac{r_{11}}{\cos(r_\vartheta)}\right) \\
r_\vartheta &= atan2\left(-r_{31}, \cos(r_\vartheta)\right) \\
r_\phi &= atan2\left(\frac{r_{32}}{\cos(r_\vartheta)}, \frac{r_{33}}{\cos(r_\vartheta)}\right)
\end{aligned} \tag{2.4}$$

with $atan2(y, x) = arctan\left(\frac{y}{x}\right)$. Furthermore, $atan2(x, y)$ determines the quadrant of the result according to the signs of both arguments.

This problem cannot be solved uniquely[1] in the pathologic cases of $\cos(r_\beta) = 0$ and $\cos(r_\vartheta) = 0$. A possible solution is $r_\alpha = atan2(r_{21}, r_{22})$, $r_\beta = 90°$ and $r_\gamma = 0°$. It is feasible to apply the single rotations in reverse order if $r_\beta \approx 90°$ or $r_\vartheta \approx 90°$. In many cases a minimal disturbance of the rotation matrix is tolerable in oder to avoid this singularity.

2.3 Gaussian Normal Distribution

Many sensor data processing algorithms assume that the probability of measuring a certain quantity is distributed according to a Gaussian normal distribution. This assumption is valid since usually several disturbances (sources of error) are incorporated in a measurement. Every disturbance is distributed according to a unique but unknown distribution. According to the stochastic *central limit theorem*, the resulting measurement can be assumed to be Gaussian distributed. The advantage of a normally distributed random variable is that it can completely be characterised by its mean and variance.

The notation $x \sim N(\bar{\mathbf{x}}, \sigma_{\mathbf{x}}^2)$ describes a Gaussian distributed one dimensional (scalar or single-variate) random variable, x, with mean $\bar{\mathbf{x}}$ and variance $\sigma_{\mathbf{x}}^2$. The probability density function (pdf), $p(x)$, of this variable is defined as follows:

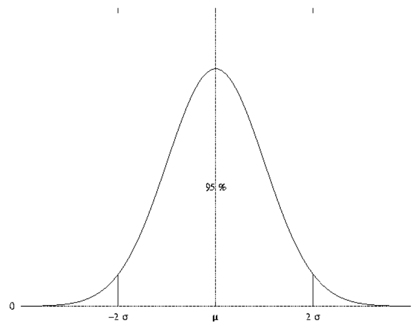
$$p(x) = N(x; \bar{\mathbf{x}}, \sigma_{\mathbf{x}}^2) = \frac{1}{\sqrt{2\pi\sigma_{\mathbf{x}}^2}} e^{-\frac{1}{2}(\frac{x-\bar{\mathbf{x}}}{\sigma_{\mathbf{x}}^2})^2} \tag{2.5}$$

The notation $x \sim N(\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$ describes a normally distributed n-dimensional (vector or multi-variate) random variable, x, with a n-dimensional mean vector, $\bar{\mathbf{x}}$, and a $n \times n$-dimensional covariance matrix, $\Sigma_{\mathbf{x}}$. The pdf, $p(x)$, of this variable is defined as follows:

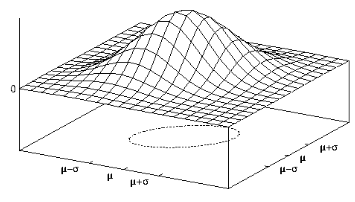
$$p(x) = N(x; \bar{\mathbf{x}}, \Sigma_{\mathbf{x}}) = \frac{1}{\sqrt{(2\pi)^n det(\Sigma_{\mathbf{x}})}} e^{-\frac{1}{2}(x-\bar{\mathbf{x}})^T \Sigma_{\mathbf{x}}^{-1}(x-\bar{\mathbf{x}})} \tag{2.6}$$

The current state of a robot can efficiently be characterised by a Gaussian random variable. The mean of this variable represents the robot's state and the covariance matrix the associated uncertainty. The same holds for measurements or observations performed with a sensor. The

[1] In fact every representation of a rotation, with three parameters, features such a singularity, e.g.(Melen, 1994).



(a)



(b)

Figure 2.1: Sigma contours (a) $H_x(2)$ for a one-dimensional and (b) $H_x(1)$ for a two-dimensional random variable, taken from Lanser (1997).

mean of this random variable represents the actual measurement and the covariance matrix the associated measurement uncertainty.

In order to display a Gaussian distribution, the concept of the Mahalanobis distance will be introduced next. The Mahalanobis distance can also be used as a measure of similarity between two Gaussian distributions. A more elaborate concept is the Bhattacharyya distance.

2.3.1 Mahalanobis Distance of a Gaussian Random Variable

Multi-dimensional Gaussian distributed random variables can efficiently be represented with the use of hyperellipsoids. Given a confidence level α, e.g. $\alpha = 0.95$, the hyperellipsoid defines a volume around the mean that contains 95 % of the distributions volume. This hyperellipsoid can also be regarded as a multi-dimensional confidence interval. Given a n-dimensional random variable, $x \sim N(\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$, the associated hyperellipsoid

$$H_x(d) = \{x \in \mathbb{R}^n | d_m(x; \bar{\mathbf{x}}, \Sigma_{\mathbf{x}}) \leq d\} \tag{2.7}$$

can be defined with the *Mahalanobis distance*

$$d_m(x; \bar{\mathbf{x}}, \Sigma_{\mathbf{x}}) = \sqrt{(x - \bar{\mathbf{x}})^T \Sigma_{\mathbf{x}}^{-1} (x - \bar{\mathbf{x}})} \tag{2.8}$$

The hyperellipsoid, $H_x(d)$, contains all vectors x, of which the distance to the mean $\bar{\mathbf{x}}$ of the random variable x is smaller than d (see Figure 2.1). The parameter d defines the confidence level, i.e. the probability $p(d)$ with which a possible value of the random variable x is contained in the set $H_x(d)$:

$$p(d) = 2\Phi(d) - 1$$

Here, $\Phi(z)$ denotes the standard normal distribution. A confidence level of 95% is achieved for $d = 2$. If the random variable describes the estimated state of a robot or an observation, the real state or the actual observation is contained with probability $p(d)$ in $H_x(d)$.

In the scalar case, the computation of the Mahalanobis distance Eq. (2.8) for the random variable $x \sim (\bar{\mathbf{x}}, \sigma^2)$ simplifies to $d_m(x; \bar{\mathbf{x}}, \sigma^2) = \frac{|x - \bar{\mathbf{x}}|}{\sigma}$. Set $H_x(d)$ contains all possible values form the confidence interval $[\bar{\mathbf{x}} - d\sigma, \bar{\mathbf{x}} + d\sigma]$, see Figure 2.1(a). In the two-dimensional case $H_x(d)$ is an ellipsoid with mean $\bar{\mathbf{x}}$, see Figure 2.1(b). Intuitively, the value of d defines the contour of an ellipsoid, the size of which is defined by $d * \Sigma$. Consequently, this contour is commonly called the d-sigma (d-σ) contour of a Gaussian distribution. The orientation and the length of an ellipsoid's (d-σ contour) axes can easily be determined by the *singular value decomposition* of $(d * \Sigma_{\mathbf{x}})^{-1}$. In the multi-dimensional case $H_x(d)$ can be thought of as an hyperellipsoid around the mean $\bar{\mathbf{x}}$.

2.3.2 Mahalanobis Distance of two Gaussian Random Variables

In the context of sensor data interpretation algorithms, it is often desirable to have a criterion that decides whether an object's estimate and an arbitrary observation, both represented as Gaussian random variables, describe the same physical object.

The Mahalanobis distance can also be used as a measure of similarity between two Gaussian distributed random variables. The squared Mahalanobis distance of the two Gaussian random variables $x \sim N(\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$ and $y \sim N(\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$ is defined as follows:

$$d_m^2(x,y) = (\bar{\mathbf{x}} - \bar{\mathbf{y}})^T (\Sigma_{\mathbf{x}} + \Sigma_{\mathbf{y}})^{-1} (\bar{\mathbf{x}} - \bar{\mathbf{y}}) \tag{2.9}$$

The squared Mahalanobis distance is distributed according to a χ^2 distribution with n degrees of freedom (Bar-Shalom and Fortmann, 1988). This can be used to define a test procedure that decides whether two Gaussian Random Variables are similar to a certain level of confidence α, e.g. $\alpha = 0.95$. Two Gaussian distributions are assumed to be similar when

$$d_m^2(x,y) \leq \chi^2_{n;\alpha} \tag{2.10}$$

The value of $\chi^2_{n;\alpha}$ can be found in the χ^2 distribution tables. Geometrically, this can be interpreted as follows. Given the σ contours of two random variables that correspond to the confidence level of α (see Figure 2.2(a)), the test criterion is successfully passed if the mean $\bar{\mathbf{y}}$ falls within the validation region of x. The validation region of y is defined by the corresponding σ contour of the sum of the covariances, $\Sigma_{\mathbf{x}} + \Sigma_{\mathbf{y}}$ (see Figure 2.2(b)). This test criterion is also commonly called a validation gate.

In the context of this thesis, the test criterion will be used to solve the data association problem, i.e. to decide whether an estimate of an object's position corresponds to a current observation. Both entities are represented as Gaussian random variables. If the test is successfully passed, it is assumed that both variables originated from the same object, and thus the measurement can be used to refine the position estimate of that object.

2.3.3 Bhattacharyya Distance of two Gaussian Random Variables

The validation gate procedure based on the Mahalanobis distance described in the previous section is commonly used to solve the data association problem. However, one significant drawback of this procedure is that it only determines, if the mean of one random variable falls within the validation region of another. The procedure does not take into account how well two Gaussian random variables overlap and thus match. A more adequate measure is the Bhattacharyya distance that computes the joint integral of the pdfs of both variables:

$$d_b^2(x,y) = \int\int p(x; \bar{\mathbf{x}}, \Sigma_{\mathbf{x}})\, p(y; \bar{\mathbf{y}}, \Sigma_{\mathbf{y}})\, dxdy \tag{2.11}$$

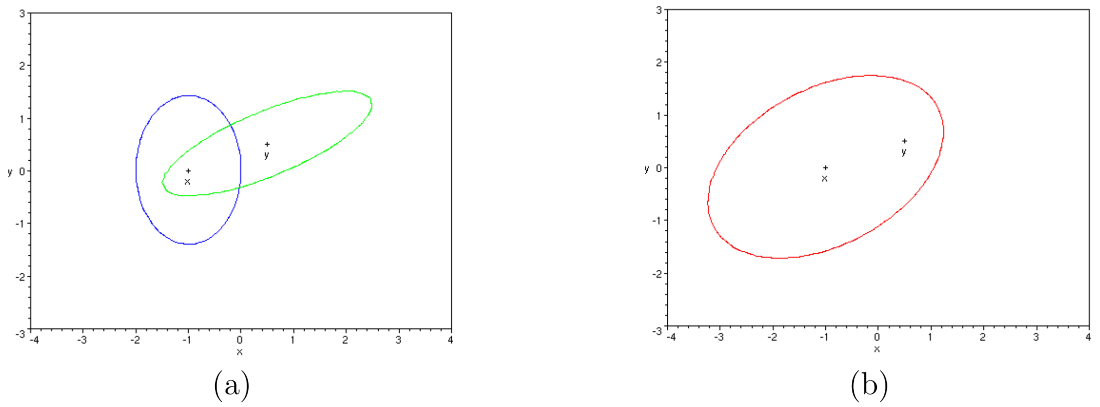


Figure 2.2: The validation region of the Mahalanobis distance spanned by two random variables. (a) Given σ contours of two random variables, x and y, representing the estimate of an object's position and a current observation, respectively. (b) The observation y can successfully be associated with the estimated position x if the mean $\bar{\mathbf{y}}$ of y falls within the validation region of x. The validation region is defined by σ contour of the sum of the covariances of x and y.

Bhattacharyya (1943) evaluated the above expression for Gaussian random variables and found an expression which is similar to the Mahalanobis distance.

$$\begin{aligned} d_b^2(x,y) &= \frac{1}{8} d_m^2(x,y) + \frac{1}{2} \ln \frac{det(\Sigma_{\mathbf{x}} + \Sigma_{\mathbf{y}})}{2\sqrt{det(\Sigma_{\mathbf{x}}) det(\Sigma_{\mathbf{y}})}} \\ &= \frac{1}{8} (\bar{\mathbf{x}} - \bar{\mathbf{y}})^T (\Sigma_{\mathbf{x}} + \Sigma_{\mathbf{y}})^{-1} (\bar{\mathbf{x}} - \bar{\mathbf{y}}) + \frac{1}{2} \ln \frac{det(\Sigma_{\mathbf{x}} + \Sigma_{\mathbf{y}})}{2\sqrt{det(\Sigma_{\mathbf{x}}) det(\Sigma_{\mathbf{y}})}} \end{aligned} \tag{2.12}$$

In fact, the Bhattacharyya distance consists of a weighted sum of the Mahalanobis distance and a second term. Consequently, the Bhattacharyya distance is computationally more expensive than the Mahalanobis distance. To the author's knowledge, the Bhattacharyya distance has not been used to solve the data association problem before.

2.3.4 Transformation of Gaussian PDFs

Sensor data processing algorithms frequently have to transform measurements and observations, performed by different sensors, into a common frame of reference or coordinate system. Measurements are generally represented by a pdf, e.g. a Gaussian normal distribution, with mean and covariance. The task is to transform the pdf of the measurement from the sensor's coordinate system to a new pdf in the common frame of reference.

A common example is the transformation of polar coordinates into Cartesian coordinates. Polar coordinates, such as bearing and range measurements, are obtained from a camera or 2D laser range finder. Bearing and range measurements are assumed to be normally distributed. The associated mean and covariance can usually be determined through a series of measurements. The transformation of this measurement into the Cartesian frame of reference determines a new mean and covariance and, as a result, a new pdf.

Formally, the transformation of pdfs can be described as follows. Given a n-dimensional random variable $x \sim N(\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$ and a transformation function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ the m-dimensional random variable $y \sim N(\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$ with $y = f(x)$ is to be obtained. Consequently, not only the value of y but also its pdfs and the associated parameters $\bar{\mathbf{y}}$ and $\Sigma_{\mathbf{y}}$ have to be determined. In order to determine the parameters of the transformed normal distribution, two cases can be distinguished, i.e. (1) f is linear and (2) f is nonlinear.

Linear Transformation

In the linear case f is defined as $f(x) = Ax + b$, with A being a $n \times m$-dimensional matrix and b being a m-dimensional vector. The transformed mean $\bar{\mathbf{y}}$ and covariance $\Sigma_{\mathbf{y}}$ can be determined as follows:

$$\begin{aligned}\bar{\mathbf{y}} &= E(y)\\ &= E(Ax+b)\\ &= AE(x)+b\\ &= A\bar{\mathbf{x}}+b\end{aligned} \tag{2.13}$$

$$\begin{aligned}\Sigma_{\mathbf{y}} &= E((y-E(y))(y-E(y))^T)\\ &= E((Ax+b-AE(x)-b)(Ax+b-AE(x)-b)^T)\\ &= E((A(x-E(x)))(A(x-E(x)))^T)\\ &= E((A(x-E(x)))((x-E(x))^T A^T))\\ &= AE((x-E(x))(x-E(x))^T)A^T\\ &= A\Sigma_{\mathbf{x}}A^T\end{aligned} \tag{2.14}$$

Nonlinear Transformation

If f is nonlinear two variants can be distinguished: (1) f is linearised and approximated by a first order Taylor series, (2) the unscented transformation is used.

Linearisation and Approximation by a first order Taylor series If f is nonlinear one possibility is to linearise it, i.e. approximate f by a first order Taylor series about an appropriate support point x_0:

$$f(x) \approx f(x_0) + \nabla F(x_0)(x - x_0) \tag{2.15}$$

$\nabla F(x_0) = \frac{df}{dx}(x_0)$ is a $n \times m$-dimensional matrix, containing the partial derivatives of f at the support point x_0. ∇F is also called the Jacobi matrix of f.

Usually, the mean of x is chosen as support point $x_0 = \bar{\mathbf{x}}$. If f now is transformed into an equivalent representation as in the linear case, then matrix A and vector b are defined by

$$A = \nabla F(\bar{\mathbf{x}}) \tag{2.16}$$

$$b = f(\bar{\mathbf{x}}) - \nabla F(\bar{\mathbf{x}})\bar{\mathbf{x}} \tag{2.17}$$

Substitution of A and b in Eq. 2.13 and Eq. 2.14 yields the transformed mean $\bar{\mathbf{y}}$ and covariance $\Sigma_{\mathbf{y}}$ as follows:

$$\begin{aligned} \bar{\mathbf{y}} &= A\bar{\mathbf{x}} + b \\ &= \nabla F(\bar{\mathbf{x}})\bar{\mathbf{x}} + f(\bar{\mathbf{x}}) - \nabla F(\bar{\mathbf{x}})\bar{\mathbf{x}} \\ &= f(\bar{\mathbf{x}}) \end{aligned} \tag{2.18}$$

$$\begin{aligned} \Sigma_{\mathbf{y}} &= A\Sigma_{\mathbf{x}}A^T \\ &= \nabla F(\bar{\mathbf{x}})\Sigma_{\mathbf{x}}\nabla F(\bar{\mathbf{x}})^T \end{aligned} \tag{2.19}$$

Unscented Transformation An alternative approach for the nonlinear transformation of Gaussian pdfs, called the *unscented transformation*, was developed by Julier and Uhlmann (1997). They followed the intuition that *it should be easier to approximate a given distribution with a fixed number of parameters than it is to approximate an arbitrary nonlinear mapping or transformation.* Following this intuition, they have found a parameterisation that captures the mean and covariance information while at the same time the unscented transformation permits the direct propagation of the information through an arbitrary set of nonlinear equations.

This can be accomplished by generating a discrete distribution having the same first and second (and possibly higher) moments, where each point in the discrete approximation can directly be transformed. The mean and covariance of the transformed ensemble can then be computed as the estimate of the nonlinear transformation of the original distribution. More generally, the application of a given nonlinear transformation to a discrete distribution of points, which are computed so as to capture a set of known statistics of an unknown distribution, is referred to as an unscented transformation (Uhlmann, 1995).

Intuitively the unscented transformation works as follows: The 1σ contour of a Gaussian random variable is sampled at predefined points. In total, $2n$ sample points are generated. Each of these sample points is propagated through the nonlinear transformation function f. Finally, the transformed set of sample points is used to determine the mean and the covariance of the transformed pdf. This procedure is illustrated in Figure 2.3.

More formally, the unscented transformation is defined as follows. Given a n-dimensional Gaussian random variable $x \sim N(\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$ and a transformation function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, the m-dimensional random variable $y \sim N(\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$ with $y = f(x)$ is determined by the following four steps:

1. Compute the set Z of $2n$ points from the rows or columns of the matrices $\pm\sqrt{n\Sigma_{\mathbf{x}}}$. This set is zero mean with covariance $\Sigma_{\mathbf{x}}$. The matrix square root can efficiently be computed by the Cholesky decomposition.

2. Compute a set of points X with the same covariance, but with mean $\bar{\mathbf{x}}$, by translating each of the points as $x_i = z_i + \bar{\mathbf{x}}$.

3. Compute a set of points Y by transforming each element of X according to the transformation function f, $y_i = f(x_i)$.

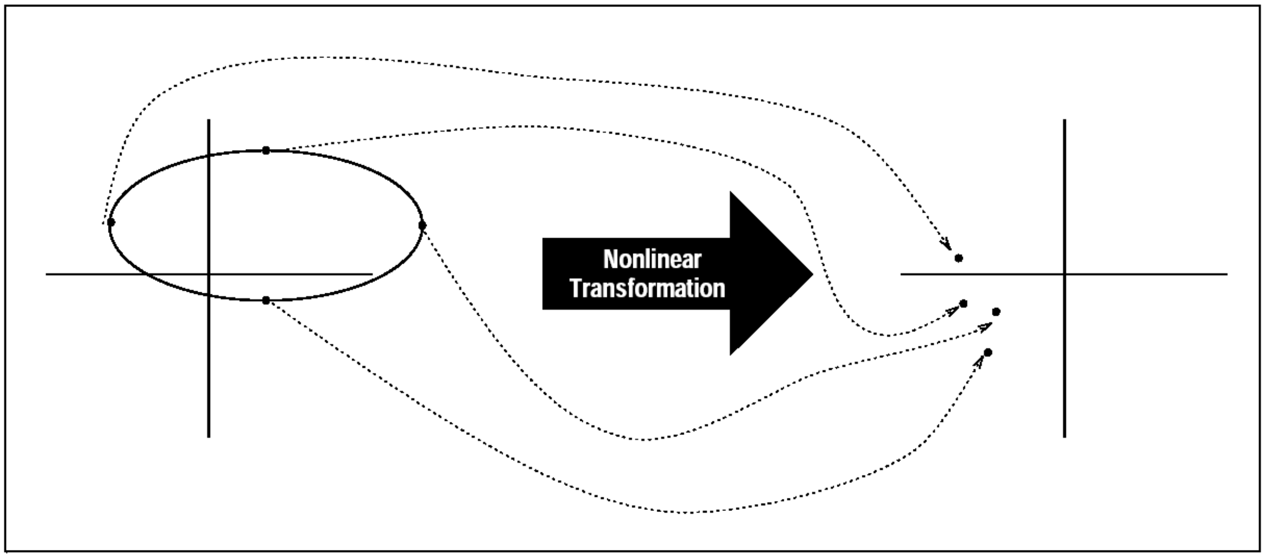


Figure 2.3: The concept of the Unscented Transformation. A set of sigma points, capturing the mean and covariance information of the distribution, is transformed according to a nonlinear transformation function and is used to approximate the mean and covariance of the transformed pdf, taken from Uhlmann (1995).

4. Compute $\bar{\mathbf{y}}$ and $\Sigma_{\mathbf{y}}$ by computing the mean and covariance of the $2n$ points in the set Y.

The unscented transformation is summarised in Figure 2.4. An analysis of this approach to transformation of Gaussian pdfs reveals that:

1. The unscented transformation is demonstrably superior to the results achieved by linearisation for all absolutely continuous nonlinear transformations. Specifically, the unscented transformation achieves second order (or better) accuracy in determining the mean as compared to the first order accuracy achieved by linearisation. Although both approaches transform the covariance correctly up to the second order, the absolute errors in the forth and higher order terms of the unscented transformation are smaller.

2. The unscented transformation can be applied with non-differentiable functions, in which linearisation by a first order Taylor series is not defined.

3. The unscented transformation avoids the derivation of Jacobian (and Hessian) matrices for linearising nonlinear kinematic and observation models. This makes the unscented transformation conducive to the creation of efficient, general purpose black box code libraries.

4. For several nonlinear transformations that are commonly required in robotic applications empirical results clearly demonstrate that linearised estimators yield very poor approximations, compared to the unscented transformation.

While the linearisation of nonlinear transformation functions has been used in a variety of applications (Lanser, 1997; Gutmann, 2000), the unscented transformation is used in this

```
algorithm UNSCENTED TRANSFORMATION ($< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >, f$)
let
    $n$                                  % dimension of $\omega$
    $< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >$                % previous <mean,covariance>
    $f()$                                % transformation function
    $X = \{x_0, \ldots, x_{2n}\}$        % set of points
    $Y = \{y_0, \ldots, y_{2n}\}$        % set of transformed points
    $< \bar{\mathbf{y}}, \Sigma_{\mathbf{y}} >$                % transformed <mean,covariance>

do
    % compute the set $X$ of $2n$ points
    for $i \leftarrow 1$ to $n$ do
        $x_i \leftarrow \bar{\mathbf{x}} + \text{COLUMN}\ (i, \sqrt{n\Sigma_{\mathbf{x}}})$;
        $x_{n+i} \leftarrow \bar{\mathbf{x}} - \text{COLUMN}\ (i, \sqrt{n\Sigma_{\mathbf{x}}})$;

    % transform each $x_i \in X$ to the set $Y$
    for $i \leftarrow 1$ to $2n$ do
        $y_i \leftarrow f(x_i)$;

    % compute transformed mean and covariance
    $\bar{\mathbf{y}} \leftarrow \frac{1}{2n} \sum_{i=1}^{2n} y_i$;
    $\Sigma_{\mathbf{y}} \leftarrow \frac{1}{2n-1} \sum_{i=1}^{2n} (y_i - \bar{\mathbf{y}})^2$;

return $(\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$;
```

Figure 2.4: The Unscented Transformation.

thesis to propagate uncertainty information through a series of nonlinear transformations, e.g. system and observation models.

2.4 Camera Model

A camera model describes the mapping of the continuous world into the discrete pixel of a video image. Knowledge about this mapping is of elementary importance for vision based localisation and state estimation. In the following section the camera model of a pinhole camera with radial distortions will be described. This model is applied throughout this thesis and constitutes a fundamental building block for the algorithms presented.

2.4.1 Pinhole Camera with Radial Distortions

The model of a pinhole camera with radial distortions is a widely used cameramodel. It was described in (Lenz, 1987). The mapping of a 3D world point $\mathbf{w}$ into a 2D pixel $\mathbf{p}$ can be

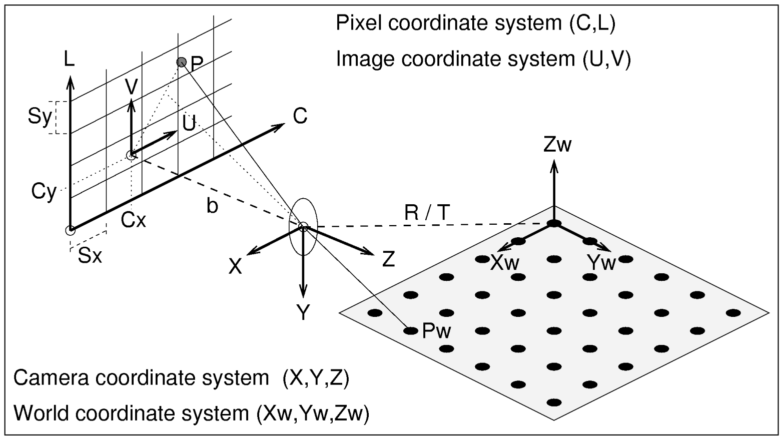

Pixel coordinate system (C,L)
Image coordinate system (U,V)
L
P
V
Sy
U
C
Cy
Cx
b
Sx
X
Z
Y
R / T
Zw
Xw
Yw
Pw
Camera coordinate system (X,Y,Z)
World coordinate system (Xw,Yw,Zw)

Figure 2.5: Model of a pinhole camera with radial distortions (Lanser, 1997): The mapping of a 3D world point **w** into a 2D pixel **p**.

decomposed into four separate steps, see Figure 2.5:

1. A 3D point **w** in world coordinates is transformed according to a rotation and a translation into a 3D point **c** in camera coordinates.
2. In the next step, **c** is projected into the 2D point **u** in the image plane according to the perspective projection[2].
3. Lens aberrations, i.e. radial distortions, are modelled in the third step. The 2D point **u** is transformed into **v**.
4. Finally, **v** is discretised into the pixel **p**.

The following subsections will look at these four steps in greater detail. More accurate and more complex camera models are described in (Weng, Cohen, and Herniou, 1992). They are distinguished from the above camera model in the third, sometimes also in the fourth step. However, the applications and algorithms presented in this thesis would not have gained any benefits from the application of another camera model than the pinhole camera model.

The pinhole camera model consists of 12 *camera parameters* : six *internal* (also called *intrinsic* or *interior*) parameters

- f - effective *focal length (focus)* of the pinhole camera,
- κ - 1st order *radial lens distortion*,
- S_x, S_y - *scale factors* to account for any uncertainty due to framegrabber horizontal/vertical scan line resampling,
- $[C_x, C_y]^T$ - coordinates of centre of radial lens distortion *(principal point)* and the intersecting point of the camera coordinate frame's Z axis with the camera's sensor plane,

[2]Earlier publications applied affine mappings as approximations of the perspective projection. Under certain circumstances, this reduces the achievable localisation and distance measurement accuracy significantly.

and six *external* (also called *extrinsic* or *exterior*) parameters.

- $r_\alpha, r_\beta, r_\gamma$ - *rotation angles* for the transform between the world and camera coordinate frames, and
- t_x, t_y, t_z - *translational components* for the transform between the world and camera coordinate frames.

The internal parameters describe how the camera forms an image, while the external parameters describe the camera's pose (i.e. position and orientation) in the world coordinate frame. The parameters are determined through a process called camera calibration. Calibration data for the model consists of 3D world coordinates $\mathbf{w}$ of a feature point (in mm for example) and corresponding 2D pixel coordinates $\mathbf{p}$ (typically in pixels) of the feature point in the image. Several different variants for camera calibration procedures are described in (Lenz and Tsai, 1986; Tsai, 1986; Lenz and Tsai, 1988; Wang, 1992).

2.4.2 3D-Transformation to Camera Coordinates

The first step is the transformation of a point $\mathbf{w} = [w_x, w_y, w_z]^T$ from the world coordinate system (WCS) into a point $\mathbf{c} = [c_x, c_y, c_z]^T$ of the camera coordinate system (CCS):

$$\mathcal{R}_c = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \ \mathcal{T}_c = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$\mathbf{c} = \mathcal{R}_c (\mathbf{w} - \mathcal{T}_c) \tag{2.20}$$

The 3D vector $\mathcal{T}_c$ and the rotation matrix $\mathcal{R}_c$ describe the position and the orientation of the camera (the camera coordinate system) relative to the surrounding world (the world coordinate system). If a camera is applied as part of a mobile robot or a manipulator, further coordinate systems such as the robot (RCS) or the manipulator coordinate system (MCS), come into play.

$$\mathbf{c} = \mathcal{R}_c (\mathcal{R}_v (\mathbf{w} - \mathcal{T}_v) - \mathcal{T}_c) \tag{2.21}$$

For many aspects of localisation and state estimation Eq. (2.21) can be transformed with

$$\begin{aligned} \mathcal{R}_c^{'} &= \mathcal{R}_c \cdot \mathcal{R}_v \\ \mathcal{T}_c^{'} &= \mathcal{R}_v^{-1} \cdot \mathcal{T}_c + \mathcal{T}_v \end{aligned}$$

into a mapping similar to Eq. (2.20). For simplicity, a single step transformation will be used in the following. For the application of multiple cameras, the propagation of uncertainties (see Section 2.3.4), and the localisation of objects (see Section 6.3.1) additional coordinate transformations according to Eq.(2.21) have to be applied.

Coordinate rotations in 3D space can be represented in several different ways. Two common possibilities, are (1) the specification of a rotation angles for every axis and (2) the Rodrigues (see Section 2.2). Every rotation in 3D space possesses three degrees of freedom. For each degree of freedom a rotation angle $(r_\alpha, r_\beta, r_\gamma)$ is specified:

$$\mathcal{R}_C = \mathcal{R}_x(r_\alpha) \cdot \mathcal{R}_y(r_\beta) \cdot \mathcal{R}_z(r_\gamma)$$

i.e. the rotation of a 3D point is expressed by a rotation around the Z-axis (angle r_γ), followed by a rotation around the new Y-axis (angle r_β) and a rotation around the new X-axis (angle r_α)[3].

2.4.3 Perspective Projection into the Image Plane

The transformation of a 3D point $\mathbf{c} = [c_x, c_y, c_z]^T$ from the CCS into a two dimensional point $\mathbf{u} = [u_x, u_y]^T$ in the image plane (image coordinate system ICS) is performed by the perspective projection

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} f\frac{c_x}{c_z} \\ f\frac{c_y}{c_z} \end{bmatrix} \tag{2.22}$$

Here, f denotes the focal length (focus) of the camera.

Annotation 2.4.1 *Equations (2.20) and (2.22) define a mapping from $\mathbb{R}^3$ to $\mathbb{R}^2$. If the Euclidean space $\mathbb{R}^3$ is embedded into the the projective space $\mathbb{P}^3$, a corresponding linear projective transformation is given through:*

$$\mathbf{u} = \mathcal{P} \cdot \mathcal{D} \cdot \mathbf{w}$$

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} & \mathcal{R} & & -\mathcal{R}\mathcal{T} \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} w_x \\ w_y \\ w_z \\ 1 \end{pmatrix} \tag{2.23}$$

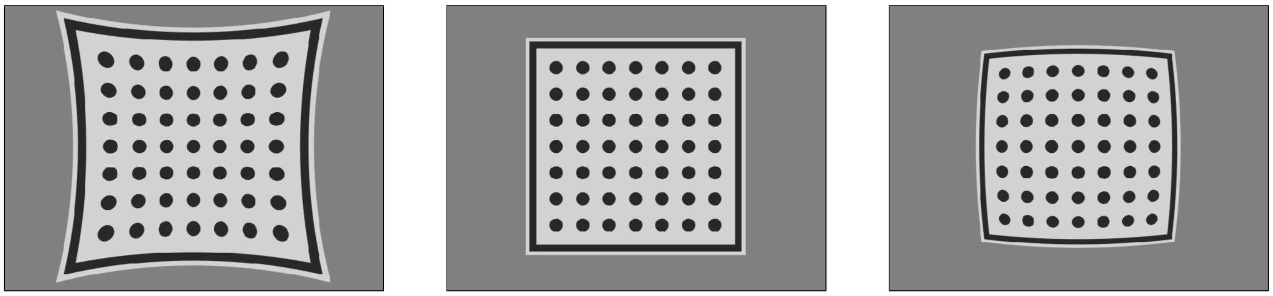
This alternative representation makes use of the so called *homogeneous coordinates* and is found throughout literature. It simplifies some geometrical considerations, as the nonlinearities of the perspective projection Eq. (2.22) are eliminated or deferred. [4]

2.4.4 Radial Distortions

The radial distortions caused by lens aberrations can be approximated by:

[3]This is one form of the so called *Euler angles*. Other representations, with reverse order of the rotation angles, are commonly used, see also Eq. (2.1) in Section 2.2.

[4] Linking the point $\mathbf{u} \in \mathbb{P}^2$ with the actual observable point, $\mathbf{u} \in \mathbb{R}^2$, reintroduces the nonlinearity again: $u_x = u_1/u_3$ respectively $u_y = u_2/u_3$.



(a) (b) (c)

Figure 2.6: Radial distortions through lens aberrations: pincushion distortions ($\kappa = 0.078\frac{1}{mm^2}$) (a), ideal mapping ($\kappa = 0$) (b) and barrel distortions ($\kappa = -0.078\frac{1}{mm^2}$) (c). Additional camera parameters (RWI Pioneer camera system): 640×480 Pixel (NTSC format), $f = 2.2\,mm$, $S_x = 0.0046\,mm$, $S_y = 0.0055\,mm$, $C_x = 318.2$ Pixel, $C_y = 236.7$ Pixel.

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \frac{2u_x}{1+\sqrt{1-4\kappa(u_x^2+u_y^2)}} \\ \frac{2u_y}{1+\sqrt{1-4\kappa(u_x^2+u_y^2)}} \end{bmatrix} \tag{2.24}$$

Here, κ represents the so called *distortion coefficient*. Eq. (2.24) allows for pincushion- ($\kappa > 0$) and barrel ($\kappa < 0$) distortions, see also Figure 2.6.

2.4.5 Transformation to discrete Pixel

Finally, the observable (sub-)pixel $\mathbf{p} = [p_x, p_y]^T$ in the pixel coordinate system (PCS) is obtained from the distorted image point $\mathbf{v} = [v_x, v_y]^T$ in ICS according to:

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{v_x}{S_x} + C_x \\ \frac{v_y}{S_y} + C_y \end{bmatrix} \tag{2.25}$$

S_x and S_y constitute *scale factors*, which represent the distance between the centres of the CCD elements on the CCD chip.[5] The coordinates, $[C_x, C_y]^T$, of the centre of the radial lens distortion and the intersection point of the camera coordinate frame's Z axis with the camera's sensor plane is called the *principal point*.

2.4.6 Inverse Camera Model

The inverse of a camera model describes the mapping of a discrete pixel of a video image into the continuous world. This mapping can be used to measure range and bearing information between camera and objects of interest, simply by back projecting the pixel coordinates of a pixel belonging to an object of interest.

[5] If pixel synchron sampling is used, S_x corresponds to the horizontal distance between two sensor centres. Otherwise, S_x is only a virtual magnitude which is highly influenced by the distance of sampling points used by the framegrabber. S_x has to be determined though a calibration procedure.

Due to the surjective nature of a camera model, the inverse cannot uniquely be defined and certain assumptions and simplifications have to be made. In the following, the inverse camera model and the underlying assumptions will be described.

Given an observable (sub-)pixel $\mathbf{p} = [p_x, p_y]^T$ in the pixel coordinate system (PCS) belonging to an object of interest, it can be transformed into a distorted image point $\mathbf{v} = [v_x, v_y]^T$ in ICS simply by the application of the inverse of Eq. (2.25):

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} S_x(p_x - C_x) \\ S_y(p_y - C_y) \end{bmatrix} \tag{2.26}$$

Again, S_x and S_y constitute *scale factors*, which represent the distance between the centres of the CCD elements on the CCD chip. The coordinates of the centre of the radial lens distortion and the intersection point of the camera coordinate frame's Z axis with the camera's sensor plane is called *principal point* and denoted by $[C_x, C_y]^T$.

Eq. (2.24) possesses a simple analytical inversion, which allows for the elimination of radial distortions during the image preprocessing:

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \frac{v_x}{1+\kappa(v_x^2+v_y^2)} \\ \frac{v_y}{1+\kappa(v_x^2+v_y^2)} \end{bmatrix} \tag{2.27}$$

with κ being the distortion coefficient of the lens aberrations.

The inverse of the perspective projections (see Eq. (2.22)) cannot uniquely be defined. Thus, the range between the optical centre and the object of interest is assumed to be the focal length, f, of the camera. By doing this the inverse of the perspective projection degrades to:

$$\mathbf{c} = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = \begin{bmatrix} \frac{u_x c_z}{f} \\ \frac{u_y c_z}{f} \\ c_z \end{bmatrix} = \begin{bmatrix} \frac{u_x f}{f} \\ \frac{u_y f}{f} \\ f \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ f \end{bmatrix} \tag{2.28}$$

Finally, the 3D point $\mathbf{c} = [c_x, c_y, c_z]^T$ is mapped from the CCS back into the WCS by inverting Eq. (2.20) and Eq. (2.21):

$$\mathbf{w}' = \begin{bmatrix} w'_x \\ w'_y \\ w'_z \end{bmatrix} = \mathcal{R}_c^{-1}\mathbf{c} \tag{2.29}$$

$$\mathbf{w} = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} t_x - t_z \frac{w'_x}{w'_z} \\ t_y - t_z \frac{w'_y}{w'_z} \\ 0 \end{bmatrix} \tag{2.30}$$

$\mathcal{R}_c$ and $\mathcal{T}_c = [t_x, t_y, t_z]^T$ are the rotation matrix and translation vector specifying the relative displacement between the CCS and the RCS or the CCS and the WCS, respectively. As before, several coordinate system transformations can be specified with one rotation matrix and one translation vector.

The underlying assumptions made in these equations is that the observed object touches the ground. Consequently, only the relative displacement between the observing camera and the object of interest can be derived. In any case, it is possible to convert this relative displacement into range and bearing information through the use of a polar coordinate system.

2.5 Conclusions

This chapter introduced notation as well as some fundamental preliminaries for sensor processing algorithms. Among the presented concepts are rotations in 3D Cartesian space, Gaussian normal distributed random variables, distance measures based on them, i.e. the Mahalanobis and Bhattacharyya distance, variants for the linear and non-linear transformation of a Gaussian random variables. The chapter is concluded with the formal model of a pinhole camera and its inverse. The presented concepts have been compiled to meet the requirements of this thesis. More detailed representations can be found in (Lanser, 1997; Gutmann, 2000; Uhlmann, 1995).

Chapter 3

State Estimation in Structured Dynamical Systems

3.1 Introduction

The problem of estimating large and complex states is the main subject addressed in this thesis. This chapter defines the formal problem in the context of a dynamic system and introduces a probabilistic framework for state estimation. This framework allows for the explicit representation of uncertainty and ambiguities associated with the estimated states. A particular and commonly used technique for probabilistic state estimation is the Bayes Filter. The general Bayes filter is introduced and several specialisations are discussed, along with their assumptions, strengths and weaknesses. Furthermore, the different implementations are analysed with respect to their properties and their applicability to self-localisation, object localisation and multiple object tracking.

3.2 Dynamical Systems

In control theory, a *dynamic system* (Dean and Wellmann, 1991) is a model that describes the interactions between a machine (called the *controller*) and its *environment*. Both entities are tightly coupled and they interact with each other through signals and actions. Usually, the controller is represented as a deterministic automaton that takes as input a signal (also called a measurement, an observation, or simply sensor data)[1] and outputs some actions (also called an input or control)[1]. The environment can be viewed as an automaton that takes the controller's action as input and generates a signal to serve as the controller's next input. Figure 3.1 illustrates this cycle. It is noteworthy that the terms *input* and *output* take the perspective of the environment rather than the controller.

In the structured dynamical system model the environment consists of two processes: (1) the environment process and, (2) the sensor process. The *environment process* changes and updates the state of the environment, according to the action generated by the controller. The *sensor process* maps the current state of the environment into a signal which can then be observed by the controller.

[1]All these terms are considered interchangeably and are used in alternating order throughout this thesis.

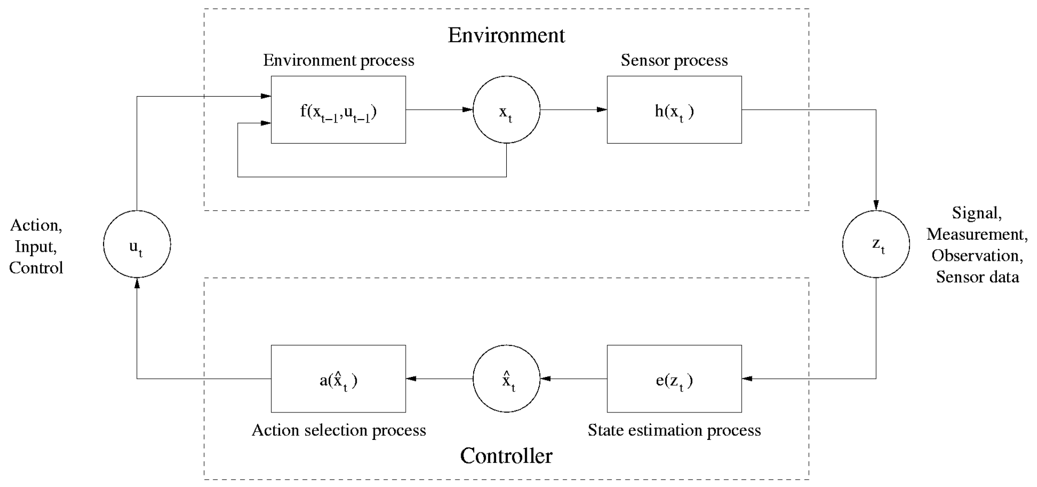


Figure 3.1: The structured dynamical system is a model that describes the interactions between a machine (controller) and its environment.

More formally, at time t, the state of the environment is represented by x_t, the output signal of the environment is z_t and the action performed by the controller is u_t. The state of the environment, x_t, evolves with time according to

$$x_t = f_t(x_{t-1}, u_{t-1}, v_{t-1}), \tag{3.1}$$

where $f_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{n_x}$ is a possibly nonlinear function of the state x_t, called the system, process or motion model[1]. v_{t-1} is the process noise, n_x, n_u and n_v are the dimensions of the state, the action and the process noise vectors, respectively. The updated state of the environment is mapped to the next output signal, z_t, by,

$$z_t = h_t(x_t, w_t), \tag{3.2}$$

where $h_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \rightarrow \mathbb{R}^{n_z}$ is a possibly nonlinear function called the observation or measurement model[1], w_t is the observation noise, and n_z, n_w are the dimensions of the observation and observation noise vectors, respectively.

The controller consists also of two processes: (1) the state estimation process, and (2) the action selection process. Given the signal of the environment, the *state estimation process* produces an estimate of the environment's state. This estimate serves as input to the *action selection process*, which in turn generates an action. Formally, the state estimation process can be described through the mapping

$$\widehat{x}_t = e_t(z_t) \tag{3.3}$$

where $e_t : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_{\widehat{x}}}$ is a possibly nonlinear function called the inverse observation or measurement model[2], $\widehat{x}_t$ is the estimate of the environments state, and $n_{\widehat{x}}$ is the dimension of the state estimate vector. It should be noted that x_t and $\widehat{x}_t$ in general have different

[2]In the context of probabilistic state estimation there is no need to distinguish between an observation model and its inverse. As such the term *inverse* is always omitted.

dimensions, thus $n_x \neq n_{\widehat{x}}$. holds. Obviously, the ideal mapping for e_t would simply be the inverse of h_t, $e_t = h_t^{-1}$. Practically, this is not possible. The application to the mapping h_t in the sensor process only performs a partial observation of the environment depending on the sensor's characteristics. Consequently, this results in a loss of information. It is impossible to accurately restore this lost information during the state estimation process. Thus, the controller's action selection process has to consider that the input is only an estimate of the environment's state and as such, may be erroneous. The action, which serves as input to the environment, is selected by

$$u_t = a_t(\widehat{x}_t) \tag{3.4}$$

where $a_t : \mathbb{R}^{n_{\widehat{x}}} \rightarrow \mathbb{R}^{n_u}$ is a possibly nonlinear function called the action selection model[3], u_t is the action selected, and n_u is the dimension of the action vector.

The correct implementation of the state estimation and the action selection processes is a fundamental prerequisite for a good controller performance. The work of this thesis focuses on the state estimation problem and aims at the generation of the best state estimates, based on the set of all available signals and actions. However, even with good state estimates a second quantity, describing the uncertainty of an estimate, is always useful and desirable.

3.3 Probabilistic State Estimation

Probabilistic state estimation (Thrun et al., 2000; Fox et al., 2000) addresses the state estimation problem within a probabilistic framework that allows for the explicit representation of uncertainties of the estimated states. The key idea is to estimate the posterior probability density over the state space conditioned on input data. In the robotics and AI literature, this posterior is typically called the *belief state* (Aström, 1965). Throughout this thesis, the following notation will be used:

$$Bel(\widehat{x}_t) = p(\widehat{x}_t | data_{0\ldots t}) \tag{3.5}$$

As before, $\widehat{x}_t$ denotes the state estimate at time t, and $data_{0\ldots t}$ denotes the data starting at time 0 up to time t. For mobile robots, two types of data can be distinguished: (1) *signals*, i.e. *observational data* such as features extracted from images or laser range measurements and (2) *actions*, i.e. *controls* or *odometry data* containing information about robot motion.[4] Denoting the former by z_t and the latter by u_t, the following expression is achieved:

$$Bel(\widehat{x}_t) = p(\widehat{x}_t | z_t, u_{t-1}, z_{t-1}, u_{t-2}, \ldots, u_0, z_0) \tag{3.6}$$

[3]The correct control theoretic term for this mapping is input regulation model. However, the controller of an autonomous robot is not only concerned with the control of its actuators, on a more abstract level, it has to deal with plans and actions. Hence, the term action selection model was chosen.

[4]It is noteworthy that in many successful autonomous robot systems the odometry readings are regarded as actions u, and thus are used to predict the next state. This is somehow contradictory to the dynamic system model, since the odometry of a robot can also be regarded as an additional sensor. However, this approach can be justified, as in general the odometry readings are a result of the actions determined by the action selection process, Eq. (3.4), and consequently are strongly correlated.

Without loss of generality, it is assumed that observations and actions occur in an alternating sequence. It is worth noting that the most recent perception in $Bel(\widehat{x}_t)$ is z_t, whereas the most recent control/odometry reading is u_{t-1}. Several different techniques for probabilistic state estimation exist. The most renowned and commonly used technique is the Bayes filter.

3.4 Bayes Filtering

Bayes filters estimate the belief (Aström, 1965) recursively. The *initial* belief characterises the *initial* knowledge about the system state. In the absence of such knowledge (e.g. global localisation), it is typically initialised by a *uniform distribution* over the state space. To derive a recursive update equation, Eq. (3.6) can be transformed by Bayes rule to

$$\begin{aligned} Bel(\widehat{x}_t) &= \frac{p(z_t|\widehat{x}_t, u_{t-1}, \ldots, z_0)\; p(\widehat{x}_t|u_{t-1}, \ldots, z_0)}{p(z_t|u_{t-1}, \ldots, z_0)} \\ &= \frac{p(z_t|\widehat{x}_t, u_{t-1}, \ldots, z_0)\; p(\widehat{x}_t|u_{t-1}, \ldots, z_0)}{p(z_t|u_{t-1}, data_{0\ldots t-1})} \end{aligned} \tag{3.7}$$

The Markov assumption states that measurements z_t are conditionally independent of the past measurements and odometry readings, given knowledge of the state $\widehat{x}_t$:

$$p(z_t|\widehat{x}_t, u_{t-1}, \ldots, z_0) = p(z_t|\widehat{x}_t) \tag{3.8}$$

Eq. (3.7) can conveniently be simplified:

$$Bel(\widehat{x}_t) = \frac{p(z_t|\widehat{x}_t)\; p(\widehat{x}_t|u_{t-1}, \ldots, z_0)}{p(z_t|u_{t-1}, data_{0\ldots t-1})} \tag{3.9}$$

To obtain the final recursive form, state $\widehat{x}_{t-1}$ at time $t-1$, has to be integrated out, which results in

$$Bel(\widehat{x}_t) = \frac{p(z_t|\widehat{x}_t)}{p(z_t|u_{t-1}, data_{0\ldots t-1})} \int p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1}, \ldots, z_0)\; p(\widehat{x}_{t-1}|u_{t-1}, \ldots, z_0)\; d\,\widehat{x}_{t-1} \tag{3.10}$$

The Markov assumption also implies that, given knowledge of $\widehat{x}_{t-1}$ and u_{t-1}, the state $\widehat{x}_t$ is conditionally independent of past measurements $z_0, \ldots, z_{t-1}$ and of odometry readings $u_0, \ldots, u_{t-2}$ up to time $t-2$, that is:

$$p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1}, \ldots, z_0) = p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1}) \tag{3.11}$$

Using the definition of the belief Bel, the recursive estimator known as Bayes filter is obtained:

$$\begin{aligned} Bel(\widehat{x}_t) &= \frac{p(z_t|\widehat{x}_t)}{p(z_t|u_{t-1}, data_{0\ldots t-1})} \int p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1})\; Bel(\widehat{x}_{t-1})\; d\,\widehat{x}_{t-1} \\ &= \nu\, p(z_t|\widehat{x}_t) \int p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1})\; Bel(\widehat{x}_{t-1})\; d\,\widehat{x}_{t-1} \end{aligned} \tag{3.12}$$

where ν is a normalising constant. This equation is of central importance, as it is the basis for various state estimation algorithms.

In order to conclude this derivation, Figure 3.2 displays the Bayes filtering algorithm. With every iteration of the algorithm, the belief state $Bel(\widehat{x}_t)$ is updated according to the supplied data. Depending on the type of data (action or signal), the algorithm can be divided into two different stages: (1) prediction, and (2) update. During the *prediction stage*, the system model Eq. (3.1) is used to obtain the prior pdf of the state at time t, via the Chapman-Kolomogorov equation:

$$Bel(\widehat{x}_t) = \int p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1})\ Bel(\widehat{x}_{t-1})\ d\,\widehat{x}_{t-1} \tag{3.13}$$

The probabilistic model of the state evolution, $p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1})$, is defined by the system equation Eq. (3.1) and the known statistics of v_{t-1}. In analogy to Eq. (3.1) and in the context of a Bayes filter $p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1})$, it is also referred to as the *system, process or motion model.*

During the *update stage*, a measurement z_t becomes available and is used to update the prior density to obtain the required posterior density of the current state:

$$Bel(\widehat{x}_t) = \nu\ p(z_t|\widehat{x}_t)\ Bel(\widehat{x}_{t-1}) \tag{3.14}$$

where the likelihood function $p(z_t|\widehat{x}_t)$ is defined by the measurement model Eq. (3.2) and the known statistics of w_t. In analogy to Eq. (3.2) $p(z_t|\widehat{x}_t)$ is also referred to (in the context of a Bayes filter) as the *observation or measurement model.* Since the measurement z_t plays the key role in this stage, this stage is also sometimes called the measurement stage.

For the successful implementation of this algorithm three distributions have to be known: the initial belief $Bel(\widehat{x}_0)$ (e.g. uniform), the next state probability $p(\widehat{x}_t|\widehat{x}_{t-1}, u_{t-1})$, and the observational likelihood $p(z_t|\widehat{x}_0)$. Assuming that these pdfs are given, three different Bayes filters, i.e. Gaussian, discrete, and Monte Carlo Bayes filter are presented in the following sections along with their individual features and properties.

3.4.1 Gaussian Bayes Filters

Gaussian Bayes filters assume that the belief, the system, and the observation model are normally distributed and, hence, can be represented by a mean and a covariance.

$$\begin{aligned} Bel(\widehat{x}_t) &= N_t(\widehat{x}_t; \bar{\mathbf{x}}_t, \Sigma_{\mathbf{x}_t}) \\ &= \frac{1}{\sqrt{(2\pi)^{n_x} det(\Sigma_{\mathbf{x}_t})}} e^{-\frac{1}{2}\{(\widehat{x}_t-\bar{\mathbf{x}}_t)^T \Sigma_{\mathbf{x}_t}^{-1} (\widehat{x}_t-\bar{\mathbf{x}}_t)\}} && (3.15) \\ & && (3.16) \end{aligned}$$

The well known Kalman filter belongs to the class of Gaussian Bayes filters. In general, it is possible to distinguish five different variants of the Kalman filter: (1) the linear Kalman filter, (2) the extended (nonlinear) Kalman filter, (3) the iterated (nonlinear) Kalman filter, (4) the unscented (nonlinear) Kalman filter, and (5) the multiple hypothesis Kalman filter. The next sections describe the different types of Kalman filters in this order.

```
algorithm BAYES FILTER ($Bel(\widehat{x}), data$)
let
    $Bel(\widehat{x})$        % previous belief state
    $Bel'(\widehat{x})$       % updated belief state
    $data$           % data item (action or signal)
    $\nu$              % normalising constant

do
    $\nu \leftarrow 0$;
    switch ($data$)

       case ($data$ is an action data item $u$) :
            % prediction stage
            for each $\widehat{x}$ do
               $Bel'(\widehat{x}) \leftarrow \int p(\widehat{x}|\widehat{x}', u)\ Bel(\widehat{x})\ d\,\widehat{x}'$;

       case ($data$ is a signal (perceptual) data item $z$) :
            % update or measurement stage
            for each $\widehat{x}$ do
               $Bel'(\widehat{x}) \leftarrow p(z|\widehat{x})\ Bel(\widehat{x})$;
               $\nu \leftarrow \nu + Bel'(\widehat{x})$;
            for each $\widehat{x}$ do
               $Bel'(\widehat{x}) \leftarrow \nu^{-1}\ Bel'(\widehat{x})$;

return ($Bel'(\widehat{x})$);
```

Figure 3.2: The Bayes Filtering Algorithm.

3.4.2 Linear Kalman Filter

The linear Kalman filter (KF) (Kalman, 1960) assumes that the system and the observation model are linear. That is, Eq. (3.1) and Eq. (3.2) can be re-written as:

$$x_t = F_t x_{t-1} + G_t u_{t-1} + v_{t-1} \tag{3.17}$$

$$z_t = H_t x_t + w_t \tag{3.18}$$

where F_t, G_t and H_t are known matrices defining the linear system and observation model. The Gaussian random variables $u_{t-1} = N(\bar{\mathbf{u}}_{t-1}, \Sigma_{\mathbf{v}_{t-1}})$ and $z_t = N(\bar{\mathbf{z}}_t, \Sigma_{\mathbf{w}_t})$ represent the action and the signal along with the process noise v_{t-1} and the measurement noise w_t, respectively. The prediction stage of the linear Kalman filter is defined by:

$$\widetilde{\mathbf{x}}_t = F_t \bar{\mathbf{x}}_{t-1} + G_t \bar{\mathbf{u}}_{t-1} \tag{3.19}$$

$$\Sigma_{\widetilde{\mathbf{x}}_t} = F_t \Sigma_{\mathbf{x}_{t-1}} F_t^T + \Sigma_{\mathbf{v}_{t-1}} \tag{3.20}$$

```
algorithm KALMAN FILTER ($< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >, data$)
let
    $< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >$      % previous belief state <mean,covariance>
    $< \bar{\mathbf{x}}', \Sigma_{\mathbf{x}'} >$    % update belief state <mean,covariance>
    $data$            % data item (action or signal)
    $F, G$            % system model
    $H$               % observation model
    $\Sigma_{\mathbf{s}}, K$          % innovation, Kalman gain

do
    switch ($data$)

        case ($data$ is an action data item $< \bar{\mathbf{u}}, \Sigma_{\mathbf{v}} >$) :
            % prediction stage
            $\bar{\mathbf{x}}' \leftarrow F\bar{\mathbf{x}} + G\bar{\mathbf{u}}$;
            $\Sigma_{\mathbf{x}'} \leftarrow F\Sigma_{\mathbf{x}}F^T + \Sigma_{\mathbf{v}}$;

        case ($data$ is a signal (perceptual) data item $< \bar{\mathbf{z}}, \Sigma_{\mathbf{w}} >$) :
            % update or measurement stage
            $\Sigma_{\mathbf{s}} = H\Sigma H^T + \Sigma_{\mathbf{w}}$;
            $K = \Sigma_{\mathbf{x}} H^T \Sigma_{\mathbf{s}}^{-1}$;
            $\bar{\mathbf{x}}' = \bar{\mathbf{x}} + K(\bar{\mathbf{z}} - H\bar{\mathbf{x}})$;
            $\Sigma_{\mathbf{x}'} = (I - KH)\Sigma_{\mathbf{x}}$;

return ($< \bar{\mathbf{x}}', \Sigma_{\mathbf{x}'} >$);
```

Figure 3.3: The Kalman Filtering Algorithm.

where $\widetilde{\mathbf{x}}_t$ and $\Sigma_{\widetilde{\mathbf{x}}_t}$ are the predicted state estimate and the associated covariance, respectively. The update stage is defined through

$$\Sigma_{\mathbf{s}_t} = H_t \Sigma_{\widetilde{\mathbf{x}}_t} H_t^T + \Sigma_{\mathbf{w}_t} \tag{3.21}$$

$$K_t = \Sigma_{\widetilde{\mathbf{x}}_t} H_t^T \Sigma_{\mathbf{s}_t}^{-1} \tag{3.22}$$

$$\bar{\mathbf{x}}_t = \widetilde{\mathbf{x}}_t + K_t(\bar{\mathbf{z}}_t - H_t\widetilde{\mathbf{x}}_t) \tag{3.23}$$

$$\Sigma_{\mathbf{x}_t} = (I - K_t H_t)\Sigma_{\widetilde{\mathbf{x}}_t} \tag{3.24}$$

where $\Sigma_{\mathbf{s}_t}$ and K_t are the covariances of the innovation term $(\bar{\mathbf{z}}_t - H_t\widetilde{\mathbf{x}}_t)$ and the Kalman gain, respectively. The standard Kalman filter algorithm is summarised in Figure 3.3. This framework has been used by numerous researchers for pose tracking and has also proven to be a good solution for sensor fusion (Leonard and Durrant-Whyte, 1991; Leonard and Durrant-Whyte, 1992; Forsberg et al., 1993; Rencken, 1994; Crowley, 1989; Crowley, Wallner, and Schiele, 1998; Gutmann, Weigel, and Nebel, 2001; Vestli and Tschichold-Gürman, 1996).

```
algorithm EXTENDED KALMAN FILTER ($< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >, data$)
let
    $< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >$       % previous belief state <mean,covariance>
    $< \bar{\mathbf{x}}', \Sigma'_{\mathbf{x}} >$     % update belief state <mean,covariance>
    $data$             % data item (action or signal)
    $\nabla F, \nabla G, \nabla V$   % Jacobians of system model and system noise
    $\nabla H, \nabla W$       % Jacobians of observation model and observation noise
    $\Sigma_{\mathbf{s}}, K$           % innovation, Kalman gain

do
    switch ($data$)

        case ($data$ is an action data item $< \bar{\mathbf{u}}, \Sigma_{\mathbf{v}} >$) :
            % prediction stage
            $\bar{\mathbf{x}}' \leftarrow f(\bar{\mathbf{x}}, \bar{\mathbf{u}});$
            $\Sigma_{\mathbf{x}'} \leftarrow \nabla F \Sigma_{\mathbf{x}} \nabla F^T + \nabla V \Sigma_{\mathbf{v}} \nabla V^T;$

        case ($data$ is a signal (perceptual) data item $< \bar{\mathbf{z}}, \Sigma_{\mathbf{w}} >$) :
            % update or measurement stage
            $\Sigma_{\mathbf{s}} \leftarrow \nabla H \Sigma_{\mathbf{x}} \nabla H^T + \nabla W \Sigma_{\mathbf{w}} \nabla W^T;$
            $K \leftarrow \Sigma_{\mathbf{x}} \nabla H^T \Sigma_{\mathbf{s}}^{-1};$
            $\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}} + K(\bar{\mathbf{z}} - h(\bar{\mathbf{x}}));$
            $\Sigma_{\mathbf{x}'} \leftarrow (I - K \nabla H) \Sigma_{\mathbf{x}};$

return ($< \bar{\mathbf{x}}', \Sigma_{\mathbf{x}'} >$);
```

Figure 3.4: The Extended Kalman Filtering Algorithm.

3.4.3 Extended Kalman Filter

A fundamental prerequisite of the linear Kalman filter is that the system, Eq. (3.1), and observation models, Eq. (3.2), are linear. However, in most applications the system model and the observation model perform coordinate transformations, which can only be accurately described by nonlinear mappings. For example, if the coordinate systems of the state estimate and the measurement are different, then the state estimate must be transformed to measurement coordinates, or vice versa. To overcome this problem, the Extended Kalman filter (EKF) was proposed (Jazwinski, 1970) and has been commonly used since. The EKF approximates the nonlinear mappings of the system and observation models by a Taylor series expansion about the current estimate $\widehat{x}_t$, which is usually truncated after the first term. This process is also sometimes called the linearisation of a mapping or a model. The success of the EKF depends on how well the system is approximated by the linearisation. The prediction stage of the EKF is defined by:

$$\begin{aligned}
\widetilde{\mathbf{x}}_t &= f_t(\bar{\mathbf{x}}_{t-1}, \bar{\mathbf{u}}_{t-1}) && (3.25)\\
\Sigma_{\widetilde{\mathbf{x}}_t} &= \nabla F_t \Sigma_{\mathbf{x}_{t-1}} \nabla F_t^T + \nabla V_t \Sigma_{\mathbf{v}_{t-1}} \nabla V_t^T && (3.26)
\end{aligned}$$

where again $\widetilde{\mathbf{x}}_t$ and $\Sigma_{\widetilde{\mathbf{x}}_t}$ are the predicted state estimate and the associated covariance, respectively. The partial derivative operator (∇) is used to indicate the approximation of the system model and the corresponding process noise by the truncated Taylor series expansion, i.e. the Jacobians ∇F_t and ∇V_t of F_t and V_t, respectively. The update stage is defined through

$$\begin{aligned}
\Sigma_{\mathbf{s}_t} &= \nabla H_t \Sigma_{\widetilde{\mathbf{x}}_t} \nabla H_t^T + \nabla W_t \Sigma_{\mathbf{w}_t} \nabla W_t^T && (3.27)\\
K_t &= \Sigma_{\widetilde{\mathbf{x}}_t} \nabla H_t^T \Sigma_{\mathbf{s}_t}^{-1} && (3.28)\\
\bar{\mathbf{x}}_t &= \widetilde{\mathbf{x}}_t + K_t(\bar{\mathbf{z}}_t - h_t(\widetilde{\mathbf{x}}_t)) && (3.29)\\
\Sigma_{\mathbf{x}_t} &= (I - K_t \nabla H_t)\Sigma_{\widetilde{\mathbf{x}}_t} && (3.30)
\end{aligned}$$

where $\Sigma_{\mathbf{s}_t}$ and K_t are the covariances of the innovation term $(\bar{\mathbf{z}}_t - h_t(\widetilde{\mathbf{x}}_t))$ and the Kalman gain, respectively. Here, ∇H_t and ∇W_t represent the Jacobians of H_t and W_t respectively. The first order EKF algorithm is summarised in Figure 3.4.

It is possible to use second and higher order terms of the Taylor series expansion. This type of EKF is called second or higher order extended Kalman filter (HKF). It is more accurate than the standard EKF, but the implementation and computational costs are substantially increased by the use of second and higher order terms. These facts prevented the wide spread of HKFs. In fact, the application of HKFs is questionable. Two alternatives exist of which the implementation and computational costs are approximate equivalent to a first order EKF. At the same time these alternatives achieve more accurate state estimates.

3.4.4 Iterated Kalman Filter and Gaussian MAP Estimation

One possibility to overcome the errors introduced by the linearisation of the observation model in the first order EKF is to use the iterated extended Kalman filter (IKF). The IKF iterates the update stage several times. With every iteration, the estimate of the previous iteration is used as predicted input and fused with the observation. With every iteration the measurement model is re-linearised about the previous estimate. The whole process is repeated a fixed number of times or until it reaches convergence, i.e. the change between two consecutive estimates becomes sufficiently small. The IKF algorithm is summarised in Figure 3.5.

In computer vision, model fitting, and tracking applications are usually formulated as a maximum a posteriori (MAP) estimation problem. The task of a MAP estimator is very similar to the one of a Bayes filter. It tries to find the most likely estimate:

$$\begin{aligned}
\widehat{x}_t &= \arg\max_x \; p(x|z_t)\\
&= \arg\max_x \; \frac{p(x) \cdot p(z_t|x)}{p(z_t)} && (3.31)
\end{aligned}$$

```
algorithm ITERATED EXTENDED KALMAN FILTER ($< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >, data$)
let
    $< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >$        % previous belief state <mean,covariance>
    $< \bar{\mathbf{x}}', \Sigma_{\mathbf{x}'} >$      % update belief state <mean,covariance>
    $data$                 % data item (action or signal)
    $\nabla F, \nabla G, \nabla V$   % Jacobians of system model and system noise
    $\nabla H, \nabla W$         % Jacobians of observation model and observation noise
    $\Sigma_{\mathbf{s}}, K$             % innovation, Kalman gain
    $i$                    % loop counter

do
    switch ($data$)

        case ($data$ is an action data item $< \bar{\mathbf{u}}, \Sigma_{\mathbf{v}} >$) :
            % prediction stage
            $\bar{\mathbf{x}}' \leftarrow f(\bar{\mathbf{x}}, \bar{\mathbf{u}});$
            $\Sigma_{\mathbf{x}'} \leftarrow \nabla F \Sigma_{\mathbf{x}} \nabla F^T + \nabla V \Sigma_{\mathbf{v}} \nabla V^T;$

        case ($data$ is a signal (perceptual) data item $< \bar{\mathbf{z}}, \Sigma_{\mathbf{w}} >$) :
            % update or measurement stage
            $i \leftarrow 0;$
            loop
                $\Sigma_{\mathbf{s}} \leftarrow \nabla H \Sigma_{\mathbf{x}} \nabla H^T + \nabla W \Sigma_{\mathbf{w}} \nabla W^T;$
                $K \leftarrow \Sigma_{\mathbf{x}} \nabla H^T \Sigma_{\mathbf{s}}^{-1};$
                $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + K(\bar{\mathbf{z}} - h(\bar{\mathbf{x}}));$
                $\Sigma_{\mathbf{x}} \leftarrow (I - K \nabla H) \Sigma_{\mathbf{x}};$
                $i \leftarrow i + 1;$
            until ($\|K(\bar{\mathbf{z}} - h(\bar{\mathbf{x}}))\| < \epsilon_{min}$ or $i > i_{max}$);
            $\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}};$
            $\Sigma_{\mathbf{x}'} \leftarrow \Sigma_{\mathbf{x}};$

return ($< \bar{\mathbf{x}}', \Sigma_{\mathbf{x}'} >$);
```

Figure 3.5: The Iterated Kalman Filtering Algorithm.

given a prior $p(x_t)$ pdf of the estimate x_t summarising all evidence gathered in the past, an observational model $p(z_t|x_t)$ and the prior of the observation $p(z_t)$. In most cases, these densities are assumed to be Gaussian distributed. Bar-Shalom and Fortmann (1988) show in their book Tracking and Data Association, pages 12f and 64f, that a MAP estimate of Gaussian pdfs is identical to the estimate of a Kalman Filter. As long as measurement and estimate are related to each other by a linear mapping, a closed form solution for the above maximisation exists.

However, in most applications this is not the case and they are related to each other by a nonlinear mapping. In this case the MAP estimate cannot be determined directly. Moreover, it has to be determined by an iterative process. For this, the MAP estimation is transformed into a numerically more favourable optimisation of a sum, by taking the negative logarithm:

$$\widehat{x}_t = \arg\min_x -2\ln(p(x)) - 2\ln(p(z_t|x)) \tag{3.32}$$

This optimisation problem can iteratively be solved by Newton minimisation. With every step, the nonlinear observation model, Eq. (3.2), is linearised about the latest estimate and a closed form solution is achieved in order to approximate a better estimate. This procedure is repeated several times or until convergence is achieved.

It can be shown that this iterative process of determining a MAP estimate with a nonlinear observation model, Eq. (3.2), is equivalent to an iterated Kalman filter. For this thesis, the notation of MAP estimation was adopted because of its notational analogy to Bayes estimation, Bayes filtering, and because of its common use throughout the computer vision research community (Lowe, 1991).

3.4.5 Unscented Kalman Filter

Another alternative to the EKF is the unscented Kalman filter (UKF) (Julier and Uhlmann, 1997; Wan and van der Merwe, 2000; Wan and van der Merwe, 2001). The UKF is based on the unscented transformation presented in Section 2.3.4. Instead of approximating the nonlinear system and observation model by their Jacobians and using these approximations to propagate the Gaussians, the UKF samples the Gaussians at several sigma points in space, transforms these points according to the nonlinear models and uses them to reconstruct the transformed Gaussian distribution. The UKF has been shown to achieve better estimation performance (smaller errors in the estimates and tighter covariances) than a first order EKF, while the computational complexity is equivalent. In fact, the state estimates can be proven to be as good as achieved by a third order EKF. Furthermore, the UKF abolishes the need to derive Jacobians and Hessians of the system and of the observation model.

In the UKF the unscented transformation is applied twice. First, in the prediction stage, in order to predict the next state and its covariance,

$$< \widetilde{\mathbf{x}}_t, \Sigma_{\widetilde{\mathbf{x}}_t} >= \textsc{Unscented Transformation}\ ((\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_{t-1}), (\Sigma_{\mathbf{x}_t}, \Sigma_{\mathbf{v}_{t-1}}), f_t) \tag{3.33}$$

and then in the update stage, in order to predict the next observation and its associated covariance.

$$\begin{aligned} < \bar{\mathbf{z}}'_t, \Sigma_{\mathbf{w'}_t} > &= \textsc{Unscented Transformation}\ ((\widetilde{\mathbf{x}}_t, \bar{\mathbf{z}}_t), (\Sigma_{\widetilde{\mathbf{x}}_t}, \Sigma_{\mathbf{w}_t}), h_t) && (3.34) \\ K_t &= \Sigma_{\widetilde{\mathbf{x}}_t} \Sigma_{\mathbf{w'}_t}^{-1} && (3.35) \\ \bar{\mathbf{x}}_t &= \widetilde{\mathbf{x}}_t + K_t(\bar{\mathbf{z}}_t - \bar{\mathbf{z}}'_t) && (3.36) \\ \Sigma_{\mathbf{x}_t} &= \Sigma_{\widetilde{\mathbf{x}}_t} - K_t \Sigma_{\mathbf{w'}_t} K_t^T && (3.37) \end{aligned}$$

where K_t is the Kalman gain. The summary of this algorithm can be found in Figure 3.6. An improvement of the UKF is the square root UKF developed by Wan and van der Merwe (2000).

```
algorithm UNSCENTED KALMAN FILTER ($< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >, data$)
let
    $< \bar{\mathbf{x}}, \Sigma_{\mathbf{x}} >$      % previous belief state <mean,covariance>
    $< \bar{\mathbf{x}}', \Sigma_{\mathbf{x}'} >$    % update belief state <mean,covariance>
    $< \bar{\mathbf{z}}', \Sigma_{\mathbf{w}'} >$    % predicted observation <mean,covariance>
    $data$          % data item (action or signal)
    $K$             % Kalman gain

do
    switch ($data$)

        case ($data$ is an action data item $< \bar{\mathbf{u}}, \Sigma_{\mathbf{v}} >$) :
            % prediction stage
            $< \bar{\mathbf{x}}', \Sigma_{\mathbf{x}'} > \leftarrow$ UNSCENTED TRANSFORMATION $((\bar{\mathbf{x}}, \bar{\mathbf{u}}), (\Sigma_{\mathbf{x}}, \Sigma_{\mathbf{v}}), f)$;

        case ($data$ is a signal (perceptual) data item $< \bar{\mathbf{z}}, \Sigma_{\mathbf{w}} >$) :
            % update or measurement stage
            $< \bar{\mathbf{z}}', \Sigma_{\mathbf{w}'} > \leftarrow$ UNSCENTED TRANSFORMATION $((\bar{\mathbf{x}}, \bar{\mathbf{z}}), (\Sigma_{\mathbf{x}}, \Sigma_{\mathbf{w}}), h)$;
            $K \leftarrow \Sigma_{\mathbf{x}} \Sigma_{\mathbf{w}'}^{-1}$;
            $\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}} + K(\bar{\mathbf{z}} - \bar{\mathbf{z}}')$;
            $\Sigma_{\mathbf{x}'} \leftarrow \Sigma_{\mathbf{x}} - K \Sigma_{\mathbf{w}'} K^T$

return ($< \bar{\mathbf{x}}', \Sigma_{\mathbf{x}'} >$);
```

Figure 3.6: The Unscented Kalman Filtering Algorithm.

The main advantage of the square root algorithm is to provide a better numerical stability and to ensure that the state covariance matrices are positive, which was not necessarily the case in the standard UKF. The square root version of the filter performs as well as the standard filter. Further improvements for nonlinear Gaussian state estimation can also be achieved through the combination of the IKF and the UKF.

3.4.6 Multiple Hypothesis Kalman Filters

One fundamental drawback of the KF arises from the fact that its state estimates can only be represented by unimodal Gaussian pdfs. As long as the KF is only used for pose tracking of a single object, a Gaussian pdf is a sufficient model for a state estimate. However, more complex tasks such as the initial localisation problem, the kidnapped robot problem or the multiple object tracking problem require a multimodal pdf that can represent several pose hypotheses[5] and association hypotheses[6] simultaneously.

[5]Throughout this thesis the term hypothesis referes to a possible object state or object pose.

[6]An association hyotheses refers to a collection of disjoint tracks.

To overcome this problem an extension to the KF, the Multiple Hypothesis Kalman Filter (MHKF), was proposed by Reid (1979). There are two basic approaches to MHKF. The first is Reid's algorithm where the association hypotheses are continually maintained and updated as measurements are received. This is the measurement oriented approach. The second is the track oriented approach where object tracks are initiated, updated, and scored before being formed into hypotheses.

The belief $Bel(\widehat{x}_t)$ of the MHKF is represented by a set of weighted pose hypotheses, H_t.

$$\begin{aligned} H_t &= \bigcup_{i=1}^{n} \{h_t^i, p_t^i\} \\ &= \bigcup_{i=1}^{n} \{< \bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i} >, p_t^i\} \end{aligned} \tag{3.38}$$

Here, each h_t^i is a Gaussian random variable, $h_t^i \sim N(\bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i})$, representing an object hypothesis (a possible object state) by a mean $\bar{\mathbf{h}}_t^i$ and a covariance $\Sigma_{\mathbf{h}_t^i}$, and p_t^i are non-negative numerical factors called the *importance factors*, which sum up to one. As the name suggests, the importance factors determine the weight (=importance) of each hypothesis. In the context of a MHKF the importance factors can be thought of as the probability of being the correct hypothesis. This representation is also sometimes called a sum or mixture of Gaussians. The belief of a state can be evaluated as follows:

$$Bel(x_t) = \sum_{i=1}^{n} p_t^i * N(x_t; \bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i}) \tag{3.39}$$

The computational structure of the MHKF is illustrated in Figure 3.7. The MHKF maintains a KF, or one of its variants, for every hypothesis. Consequently, the models and equations used during the prediction and update stage are identical to the ones used in a standard KF. The implementation of the prediction stage is straight forward, despite the fact that the system model has to be applied to every hypothesis (Apply Motion Model). The update stage has to be extended such that it can handle multiple measurements, multiple hypotheses (add new, update existing and delete unlikely hypotheses), and can perform the probability computations for all hypotheses.

For the following, it is assumed that the measurement vector consists of a number of possible poses or object observations, which were determined by the sensor data processing or feature extraction algorithms. The first task performed by the update stage is to copy all existing hypotheses to the set of new hypotheses. This accounts for the fact, that none of the hypotheses might be reconfirmed by a measurement and is also referred to as track splitting.

Then, the update stage has to assign the newly determined observations or measurements to the existing hypotheses. This process is called data association. An association is usually performed on the basis of a validation gate (Test Validation Gate). Typically, for Gaussian pdfs the Mahalanobis (see Eq. 2.9) or Bhattacharyya (see Eq. 2.12) distance are used as a validation gate. If an observation falls within the validation gate of an existing hypothesis, then they are assumed to have originated from the same physical object. Consequently, the measurement and the hypothesis can be fused (Associate) by the KF update equations and are used to create a new hypothesis.

```
algorithm MULTIPLE HYPOTHESIS KALMAN FILTER ($H, data$)
let
    $H$            % set of hypotheses representing previous belief state
    $H'$           % set of hypotheses representing updated belief state
    $data$         % data item (action or signal)
    $h, h^i, h^j$  % hypothesis
    $Z$            % set of measurements or observations
    $z, z^i, z^j$  % measurement or observation
    $p$            % weight of an hypothesis
    $\nu$          % normalising constant

do
    switch ($data$)

      case ($data$ is an action data item $u$) :
          % prediction stage
          $H' \leftarrow \emptyset$;
          for $i \leftarrow 1$ to $|H|$ do
              $h \leftarrow$ APPLY MOTION MODEL ($h^i, u$);
              $H' \leftarrow H' \cup \{< h, p^i >\}$;

      case ($data$ is a signal (perceptual) data vector $Z$) :
          % update or measurement stage
          $H' \leftarrow H$;
          $\nu \leftarrow \sum_{i=1}^{|H|} p^i$;
          for $j \leftarrow 1$ to $|Z|$ do
              for $j \leftarrow 1$ to $|H|$ do
                  if TEST VALIDATION GATE ($z^i, h^j$) then
                     $h \leftarrow$ ASSOCIATE ($z^i, h^j$);
                     $p \leftarrow p(h|Z)$;
                     $H' \leftarrow H' \cup \{< h, p >\}$;
                     $\nu \leftarrow \nu + p$;
          % generate new hypothesis
          for $i \leftarrow 1$ to $|Z|$ do
              if $z^i \notin H'$ then
                 $H' \leftarrow H' \cup \{< h^i, p_{start} >\}$;
                 $\nu \leftarrow \nu + p_{start}$;
          for $j \leftarrow 1$ to $|H'|$ do
              $p^i \leftarrow \nu^{-1} p^i$;
          % prune set of hypotheses
          $H' \leftarrow$ PRUNE HYPOTHESES($H'$);

return ($H'$);
```

Figure 3.7: The Multiple Hypothesis Bayes Filtering Algorithm.

The computation of the importance factor (probability) of an hypothesis requires the evaluation of the expression $P(h|Z)$, where h is the hypothesis for which the probability is computed, and Z is the set of all current observations. The evaluation of this expression is highly task dependent. Jensfelt and Kristensen (2001) introduced a variant that solves the initial localisation problem for an autonomous mobile robot (see also Section 5.4.2). Bar-Shalom and Fortmann (1988) presented a solution for the multiple object tracking problem in cluttered environments and Cox and Leonard (1994) proposed a modification, which allows dynamic environment modelling.

Observations, that cannot be assigned to an existing hypothesis, are used to initialise new hypotheses. Usually, the probabilities of these hypotheses are initialised with a predefined constant probability. Alternative approaches are to derive an initial probability from the measurements covariance matrix.

Finally, in order to constrain the growth of the set of hypotheses and the computational demand required by the MHKF, the set of hypotheses is pruned (PRUNE HYPOTHESES). It is obvious, that several efficient pruning strategies exist. For example, similar hypotheses can be merged and unlikely ones can be discarded. An upper bound on the maximal possible number of hypotheses allows computational resources to be saved. It is noteworthy that the deployed pruning strategies and their parameters are usually very much application dependent.

3.4.7 Discrete Bayes Filters

Discrete Bayes filter (BDF) algorithms assume that the state space can be divided into a fixed number of discrete states. The idea of DBFs is to represent the belief $Bel(\widehat{x}_t)$ with a set that has a fixed number of weighted discrete states:

$$Bel(\widehat{x}_t) = \begin{cases} p_t^i & , \widehat{x}_t \in x_t^i \\ 0 & , otherwise. \end{cases} \tag{3.40}$$

Here, each x_t^i is a possible state, and p_t^i are non-negative numerical factors representing the probability of this state. The probabilities of all states sum up to one. Hence, $Bel(\widehat{x}_t)$ can be regarded as a piecewise constant pdf.

The big advantage of the DBFs is that system and observation models do not necessarily have to be linear. In fact, the associated pdf can have arbitrary shapes. The prediction state of the DBF is similar to Eq. 3.13. Since the state space is discretised into a fixed number of states the integral can be replaced by a sum:

$$Bel(\widehat{x}_t) = \sum p(\widehat{x}_t | \widehat{x}_{t-1}, u_{t-1})\ Bel(\widehat{x}_{t-1}) \tag{3.41}$$

The update stage of the DBF is identical to Eq. 3.14:

$$Bel(\widehat{x}_t) = \nu\ p(z_t | \widehat{x}_t)\ Bel(\widehat{x}_{t-1}) \tag{3.42}$$

A summary of the DBF can be found in Figure 3.8. Successful DBF implementations and robotic applications can be found in (Simmons and Koenig, 1995; Burgard et al., 1996; Fox, Burgard, and Thrun, 1999). The clear advantage of DBFs are that the belief, the system model, and the observation model can have almost arbitrary shapes and are not limited to the use of Gaussian pdfs. Thus, DBFs allow for global localisation. The disadvantages of DBFs are their limited accuracy, the required computational demand for iterating over the set of states, and the fact that the number of states has to be known in advance and always remains fixed. Consequently, DBFs scale poorly with an increasing environment.

```
algorithm DISCRETE BAYES FILTER ($Bel(\widehat{x}), data$)
let
    $Bel(\widehat{x})$      % previous belief state
    $Bel'(\widehat{x})$     % updated belief state
    $data$        % data item (action or signal)
    $\nu$           % normalising constant

do
    $\nu \leftarrow 0$;
    switch ($data$)

        case ($data$ is an action data item $u$) :
            % prediction stage
            for each $\widehat{x}$ do
                $Bel'(\widehat{x}) \leftarrow \sum_{\widehat{x}'} p(\widehat{x}|u, \widehat{x}')\ Bel(\widehat{x})$;

        case ($data$ is a signal (perceptual) data item $z$) :
            % update or measurement stage
            for each $\widehat{x}$ do
                $Bel'(\widehat{x}) \leftarrow p(z|\widehat{x})\ Bel(\widehat{x})$;
                $\nu \leftarrow \nu + Bel'(\widehat{x})$;
            for each $\widehat{x}$ do
                $Bel'(\widehat{x}) \leftarrow \nu^{-1}\ Bel'(\widehat{x})$;

return ($Bel'(\widehat{x})$);
```

Figure 3.8: The Discrete Bayes Filtering Algorithm.

3.4.8 Monte Carlo Bayes Filters

All previously described methods are either limited to unimodal Gaussian state estimates or assume that the size of the state space is limited and can be represented by a discrete representation. Both assumptions are frequently violated in robotic applications.

To handle these problems, sequential Monte Carlo Bayes filter (MCBF), also known as particle filters (PFs), and have been introduced by (Handschin and Mayne, 1969; Akashi and Kumamoto, 1977). In the mid 1990s, several PF algorithms were proposed independently under the names of Monte Carlo filters (Kitagawa, 1996), sequential importance sampling (SIS) with resampling (SIR) (Doucet, 1998), bootstrap filters (Gordon, Salmond, and Smith, 1993), condensation tracker (Isard and Blake, 1996a; Blake and Isard, 1998), dynamic mixture models (West, 1993), survival of the fittest (Kanazawa, Koller, and Russel, 1995), etc. One of the major innovations during the 1990s was the inclusion of a resampling step to avoid degeneracy problems inherent to the earlier algorithms (Gordon, Salmond, and Smith, 1993). In the late nineties, several statistical improvements for PFs were proposed

```
algorithm MONTE CARLO BAYES FILTER ($S, data$)
let
    $S$              % set of samples representing previous belief state
    $S'$             % set of samples representing updated belief state
    $data$           % data item (action or signal)
    $x^i, x^j$       % sample pose
    $p^i, p^j$       % weight of a sample pose
    $\nu$            % normalising constant

do
    $S' \leftarrow \emptyset$;
    $\nu \leftarrow 0$;
    switch ($data$)

      case ($data$ is an action data item $u$) :
          % prediction stage
          for $i = 1$ to $|S|$ do
             % generate new samples
             $x^j$ = DRAW SAMPLE ($S$);
             $x^i$ = DRAW SAMPLE ($p(x^i|x^j, u)$);
             $S' \leftarrow S' \cup \{< x^i, p^j >\}$;

      case ($data$ is a signal (perceptual) data item $z$) :
          % update or measurement stage
          for $i = 1$ to $|S|$ do
             $p^i \leftarrow p(z|x^i)$;
             $\nu \leftarrow \nu + p^i$;
             $S' \leftarrow S' \cup \{< x^i, p^i >\}$;
          for $i = 1$ to $|S'|$ do
             $p^i \leftarrow \nu^{-1} p^i$;

return ($S'$);
```

Figure 3.9: The Monte Carlo Bayes Filter Algorithm.

and some important theoretical properties were established. In addition, these algorithms were applied and tested in many domains. An up-to-date survey of the field can be found in (Doucet, de Freitas, and Gordon, 2000).

The idea of MCBF algorithms (and other particle filter algorithms) is to represent the belief $Bel(\widehat{x}_t)$ with a set of weighted samples, S_t, distributed according to $Bel(\widehat{x}_t)$:

$$S_t = \bigcup_{1}^{n} \{x_t^i, p_t^i\} \tag{3.43}$$

Here, each x_t^i is a sample (a state), and p_t^i are non-negative numerical factors called the *importance factors*, which sum up to one. As the name suggests, the importance factors determine the weight (=importance) of each sample. The belief of a state can be evaluated as follows:

$$Bel(\widehat{x}_t) = \begin{cases} p_t^i & , \widehat{x}_t = x_t^i \\ 0 & , otherwise. \end{cases} \tag{3.44}$$

In global mobile robot localisation, the initial belief is a set of poses drawn according to a uniform distribution over the robot's state space, annotated by the uniform importance factor $\frac{1}{n}$. The recursive update is realized in three steps. The first two steps correspond to the prediction stage and the third step to the update stage of a Bayes filter. Altogether they compute the expression in Eq. 3.12 from the right to the left.

1. State x_t is sampled from $Bel(\widehat{x}_t)$, by drawing a random x_{t-1}^i from the sample set representing $Bel(\widehat{x}_{t-1})$ according to the (discrete) distribution defined through the importance factors p_{t-1}^i.
2. The sample x_{t-1}^i and the action u_{t-1} is used to sample x_t^j from the distribution $p(x_t|x_{t-1}, u_{t-1})$. The predictive density of x_t^j is now given by the product $p(x_t|x_{t-1}, u_{t-1})Bel(\widehat{x}_{t-1})$.
3. Finally, the samples x_t^j are weighted by the (non-normalised) importance factor $p(z_t|x_t^j)$, the likelihood of the sample x_t^j given the observation z_t.

After the generation of n samples, the new importance factors are normalised so that they sum up to 1 and, hence, define a probability distribution. This procedure implements Eq. 3.12 using an (approximate) sample-based representation. A summary of the algorithm is given in Figure 3.9. Obviously, this algorithm constitutes just one possible implementation of the particle filter idea. In Sensor Resetting Localisation (SRL) (Lenser and Veloso, 2000), the idea is to draw a fraction of the samples in the resampling step not from the previous set, but instead directly based on where the measurements indicate that there should be samples. Adaptive MCL (AMCL) (Crisman et al., 2002) extends SRL with a schema for adaptively determining how many samples should be added. Mixture MCL (MMCL) (Thrun, Fox, and Burgard, 2000) also draws samples from the observations, but the samples are properly weighted, with the probability assigned to the position where the sample is placed. This probability is typically estimated based on a grid approximation. Further MCBF implementations and successful applications to autonomous robots can be found in (Kitagawa, 1996) and (Dellaert et al., 1999a; Dellaert et al., 1999b; Fox et al., 1999), respectively (see also Section 5.4.2).

The clear advantages of MCBFs are that the belief, the system model, and the observation model can have arbitrary shapes and are not limited to the use of Gaussian pdfs. Accordingly, MCBFs allow for global localisation. Furthermore, MCBFs scale easier to larger environments than DBFs. The disadvantages of MCBFs are their limited accuracy and the required computational demand. Several extensions of MCBFs were proposed, which allow for adapting the number of samples and, thus, to reduce the computational demand required upon the uncertainty of the estimate represented in the belief (Fox, 2001; Fox, 2003; Kwok, Fox, and Meila, 2003).

	KF	MHKF	DBF	MCBF
System/Perceptual model	Gaussian	Gaussian	Non-Gaussian	Non-Gaussian
Belief state/Posterior	Gaussian	Multimodal Gaussian	Piecewise constant	Multimodal point
State space	arbitrary	arbitrary	fixed	arbitrary
Efficiency (memory)	++	+	−	−
Efficiency (time)	++	+	o	−
Implementation	+	+	o	o
Accuracy	++	++	+	o
Robustness	o	+	++	++
Global/Initial localisation	−	$\surd$	$\surd$	$\surd$
Multiple-Object tracking	o	$\surd$	o	o

Table 3.1: Strengths and weaknesses of Bayes filters.

3.5 Conclusions

Several different variants of the Bayes filter exist and are commonly used in successful robotic applications. The strengths and the weaknesses of the variants are summarised in Table 3.1.

The clear advantage of the Kalman filter based methods (KF and MHKF) is the high accuracy of the state estimates. Fast update times and limited memory requirements make them highly employable for real time applications in robotics, where only limited resources are available. The clear disadvantage of the KF is its limitation to Gaussian system/perceptual models and its representation of the belief by an unimodal Gaussian pdf. This leads to limited robustness when noisy observation are integrated and to the incapability to solve the global localisation problem. This problem can be greatly overcome by the MHKF. However, the need for efficient pruning heuristics makes it less applicable and a good working solution is highly domain specific and problem dependent.

DBF and MCBF exploit more capabilities of the Bayes filter. They are applicable with arbitrary system and perceptual models and allow for more complex (almost arbitrary) representations of the belief. This makes them very robust towards noise and allows them to solve the global localisation problem. This advantage comes with the drawback of limited achievable accuracy, particularly, a MCBF converges with a rate of $1/\sqrt{n}$ with n being the number of samples (Tanner, 1993; Thrun, Fox, and Burgard, 2000), and tremendous requirements towards computational resources and storage capacities. An additional disadvantage of the DBF is that the state space and its discretisation have to be known prior to robot deployment and cannot be adapted or changed while the filter is running. Recently proposed improvements (Kwok, Fox, and Meila, 2003; Fox, 2001) allow a MCBF to adapt the number of particles and achieve real time performance. With the further improvement of the state-of-the-art computer technology, MCBF will become the method of choice.

Chapter 4

Physical Embedding and Integration of State Estimation Modules in a Robot Control Programme

4.1 Introduction

RoboCup is an international research and educational initiative. The goal of this initiative is to foster Artificial Intelligence and Robotics research by providing a standard problem, where a wide range of technologies can be examined and integrated. Robot soccer creates a new class of applications. Autonomous agents have to cooperate within a highly dynamic, complex and partially destructive environment in order to achieve a common goal. Successful solutions to this problem have to combine results from the fields of multi-sensor, multi-robot, and multi-agent-systems research. Finally, robotic soccer has become a standard "real-world" testbed for the control of cooperative autonomous multi-robot systems, the development and the evaluation of state estimation techniques embedded in robot control programmes (Stone et al., 2000).

The survey in Section 4.2 introduces the Robot World Cup Initiative (short RoboCup) along with their activities and areas of research. Furthermore, all aspects relevant to this thesis, as well as the rules of the different leagues are outlined. Section 4.3 introduces the approach chosen by the *The AGILO RoboCuppers*, the RoboCup team of the Munich University of Technology, in order to solve the RoboCup problem. Section 4.4 presents the robots' hardware architecture, which serves as experimental platform for the research conducted for this thesis. All aspects of the robots' software architecture and their common belief state are presented in Section 4.5.

4.2 Autonomous Robot Soccer - RoboCup

The concept of soccer-playing robots was first introduced in 1993. Following a two-year feasibility study, and a Workshop held in Osaka, on November 1996. In July 1997, the first official Symposium and Competition was held in Nagoya, Japan. Followed by Paris, Stockholm, Melbourne, Seattle, Fukuoka/Busan, and Padua the annual events attracted more and more participants. The last RoboCup in 2003 involved 1004 participants and 188 Teams, from 29 Nations, and attracted 117,000 visitors. Today, more than 3,000 researchers from 35 countries and regions are participating all around the world in various projects such as international competitions, conferences, research and educational programmes. Following previous Symposia and Competitions, the 9th RoboCup will be hosted in the City of Lisbon (Portugal), by The University of Lisbon, in July 2004.

RoboCup Research activities and tournaments are divided in several classes or leagues. The RoboCup Simulation Leagues are entirely virtual and purely based on software. Real robot leagues include the Small Size League (small robots occupying an area of up to $180cm^2$), Middle Size League (middle sized robots occupying an area of up to $2000cm^2$), the Legged League (Sony's four legged robot dogs called AIBO), the Humanoid League (humanoid like robots on two legs) and the RoboCup Rescue Robot League (real robots assisting search and rescue teams in disaster areas).

RoboCup Simulation League: A RoboCup Simulation match is operated as a client-server-application. Each team consists of 11 software clients, the so called autonomous software agents. These agents communicate with the soccer server through predefined communication channels.

With every simulation step, the soccer server provides the clients with information about the current state of the match, i.e. observations of the relative positions of the software agents, the ball and the field's landmarks. Within a predefined period of time, the clients are required to analyse the provided data, estimate the game state, and choose an action. This process is made more difficult through the association of observations with systematic and random errors. However, it is still possible for the agents to determine the correct and complete game state. The selected action is sent back to the soccer server. The soccer server executes the action and updates the state of the match. If a client is not able to react in time, its action will be executed in the next time step. Each client is only allowed to control one player. Limited communication among players can only take place via the soccer server.

The main research areas in the RoboCup Simulation League are action selection, behaviour modelling and learning, strategy acquisition, and multi-agent cooperation.

RoboCup Small Size League – F 180: A small size robot soccer game takes place between two teams of five robots each. The robots must fit within an 180mm diameter circle and must be no higher than 15cm unless they use on-board vision. The robots play soccer on a green carpeted field that is 2.8 m long and 2.3 m wide with an orange golf ball. Robots come in two flavours, those with local on-board vision sensors and those with global vision. Global vision robots use an overhead camera and off-field PC to identify and track the robots as they move around the field. Local vision robots have their sensing on the robot itself. The vision information is either processed on-board the robot or is transmitted back to the off-field PC for processing. An off-field PC is used for communicating referee commands, and in the case of an overhead vision for giving information about actions to the robots. Typically, the off-field PC also performs most, if not all, of the processing required for coordination and control of the robots. Communication is wireless and typically uses dedicated commercial FM transmitter/receiver

Object	**Color**
Field surface	GREEN
Lines on the field and the walls.	WHITE
Ball	ORANGE
One of the goals	BLUE
The other goal	YELLOW
Flagposts	BLUE and YELLOW
Robot bodies	BLACK
Markers of robots for team A	LIGHT BLUE
Markers of robots for team B	MAGENTA/PURPLE

Figure 4.1: Summary of object colourings in the RoboCup Middle Size League.

units.

Building a successful team requires clever design, implementation and integration of many hardware and software sub-components into a robustly functioning system. The RoboCup small size league focuses on the problem of intelligent multi-agent cooperation and control in a highly dynamic environment with a hybrid centralised/distributed system. Perception and game state estimation is straight forward when a global vision system is used. Game state estimation with local sensors is a challenge but still a lot easier, which is due to the small field, than in the middle size league.

RoboCup Middle Size League – F 2000: The concepts and techniques developed in this thesis are implemented and tested on a system of middle robots and are evaluated during matches of the RoboCup Middle Size league. Consequently, this section will investigate the scenario and the rules of the middle size league in greater detail than for the other leagues.

In the RoboCup Middle Size League, two teams of four autonomous robots — one goal keeper and three field players — play soccer against each other. The soccer field is between 8 to 12 meters long and 5 to 10 meters wide. Lines mark the boundary of the field, the middle line, the centre circle and the penalty area. The goals are two meters wide and about 50 cm deep. The corners of the playing field are marked by four corner flagposts. Each post has a diameter of 20 cm and a height of one meter. The robots may occupy an area of up to $2000cm^2$, which is approximately equivalent to a diameter of up to 50 cm. The primary target for each team is the same as in real soccer, to dribble or kick the ball (a standard FIFA soccer ball) into the opponent goal. A match is divided into two halves of 10 minutes.

All computation, such as sensor data processing, state estimation, environment modelling, path planning and robot control, have to be performed on board the robot's own computer. The employed sensors mainly include cameras (directional and omni-directional), laser-range-finders (180 and 360 degrees), tactile, odometric, infrared and ultrasonic sensors. All objects of interest are encoded with predefined unique colours, see Figure 4.1. Global sensor systems for guiding the robots are not permitted, i.e. the robots rely exclusively on the information obtained by their local sensors, with a possibly limited field of view. However, the robots may communicate and exchange information in order to obtain a more complete and correct belief state of the world. Considering the size of the field, self-localisation, object detection and robot tracking with a conventional camera is a hard problem. This constitutes a though challenge for the research areas of state estimation and distributed sensor processing. Since 2002, global sensor systems can be used for match analysis and the determination of ground truth data that can be used after a match in order to evaluate different state estimation approaches.

Further important rules state that the attacking and defending robots (except the goal keeper) may enter the penalty area only for a limited period of time (typically 5 to 10 seconds). This requires the robots to have a vague idea of their own pose on the field.

Charging opponent robots is prohibited. This rule requires the robots to possess a good obstacle detection, obstacle localisation and obstacle avoidance mechanism. Movements of observed and observing robots make this problem more difficult. The faster a robot moves, the more important an accurate absolute localisation gets.

The key characteristics of middle size robot soccer is that the robots are completely autonomous. Consequently, all sensing and all action selection is done on board of the individual robots. Skillful play requires robots to recognise objects, such as other robots, field lines, and goals, estimate a complete and comprehensive state of the game, perform cooperative action selection, and perform path planning while they have to meet some tough real time constraints.

RoboCup Legged and Humanoid Leagues: One of the ultimate dreams in robotics is to create life-like robotic systems, such as humanoid robots and animal-like legged robots. The robots used in the Legged and Humanoid Leagues are the dog like AIBO robots of SONY and custom-made humanoid robots, respectively. In order to perceive the environment, these robots are mainly equipped with vision sensors. The absence of a communication link between the robots and their leg based locomotion might hamper successful state estimation additionally. Consequently, game state estimation in these leagues is at least as challenging as in the RoboCup Middle Size League, if not more.

RoboCup Rescue Leagues: Disaster rescue is one of the most serious social issues which involves very large numbers of heterogeneous agents in a hostile environment. The intention of the RoboCup Rescue project is to promote research and development in this socially significant domain at various levels. These involve multi-agent team work coordination, physical robotic agents for search and rescue, information infrastructures, personal digital assistants, a standard simulator and decision support systems, evaluation benchmarks for rescue strategies, and robotic systems that are all integrated into a comprehensive systems in future.

The two leagues that are currently active are the RoboCup Rescue Simulation and RoboCup Real Robot Rescue Leagues. While the former aims at research in the areas of multi-agent coordination and the application of traditional AI techniques to resource management problems, the latter confronts physical robots with the toughest conceivable perception and state estimation problems.

4.3 The Approach of the AGILO RoboCuppers

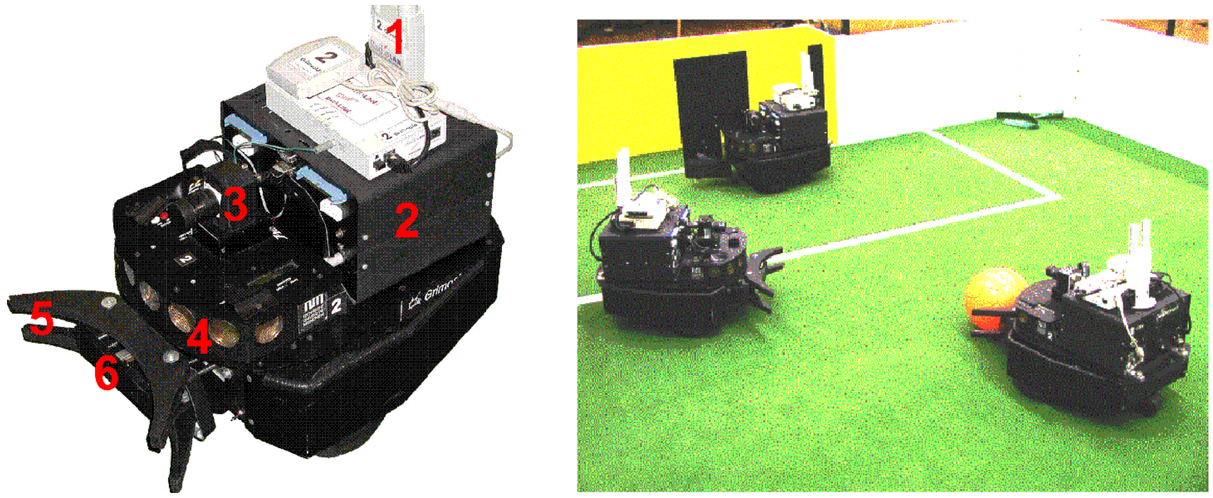
The approach chosen by *The AGILO RoboCuppers* is to tackle the RoboCup problem with a standardised robotic platform. This platform consists of a robot chassis, a conventional CCD video camera and off-the-shelf computing hardware, such as a laptop computer with moderate computational resources. Perception and state estimation of the environment is performed on the basis of video images and odometry readings only. They are used to determine the pose of the robot, the position of the ball and positions of the opponents. Several different probabilistic state estimation techniques are applied and they cooperate in order to generate probabilistic estimates for these quantities and store them in an explicit probabilistic world model, the belief state (Aström, 1965). A probabilistic world model was chosen as it allows for an easy representation and easy integration of uncertain and ambiguous knowledge, such as sensor data. Furthermore, the uncertain information acquired by several robots can be integrated in an efficient way and support the implementation of a cooperative state estimation system. The belief state is the central data structure used for the control of the robot. All action selection and path planning is performed on the basis of the belief state, i.e. actions are chosen deliberatively as a result of specific game situations (e.g. the ball is possessed by an opponent player) and a path for the robot is planned such that it avoids collisions with all dynamic objects contained in the belief state. Detailed descriptions of this approach can also be found in (Beetz et al., 2002a; Schmitt, Buck, and Beetz, 2001; Buck et al., 2000; Bandlow et al., 1999a; Klupsch et al., 1998)

This approach of *The AGILO RoboCuppers* has proven to be applicable and successful during five RoboCup World Championships and three German Championships. A more detailed description of the used Hardware and Software is presented in the next two sections.

4.4 Hardware Platform and Experimental Setup

The AGILO RoboCuppers (Schmitt et al., 2002; Beetz et al., 2004; Beetz et al., 2002b; Beetz et al., 2002a; Schmitt, Buck, and Beetz, 2001; Buck et al., 2000), the RoboCup team of the Munich University of Technology, consist of five Pioneer I robots; one of them is depicted in Figure 4.2(a). The names are an homage to the Agilolfinger, the earliest dynasty ruling in Bavaria during the 6th century. The dynasty's most famous representatives are Garibald, Grimoald, Hugibert, Odilo, Tassilo and Theodo. Only four of them are allowed to play at a match at a given time. Tassilo usually serves as hardware backup and substitution player. The robots are equipped as follows:

1. A wireless Ethernet (RadioLAN 10 Mbit/sec, 5.8 GHz) serves as communication device among the robots themselves and between the referee and the robots. Through this



(a) (b)

Figure 4.2: (a) An AGILO soccer robot and (b) a match situation.

communication channel a kinds of information are distributed among the members of the team. This includes information about the poses of the robots, the position and velocities of the ball and the opponent players and the commands given by referee (game start, game stop, score, yellow/blue/red cards), etc.

Through several competitions held from 1999 to 2002, the RadioLAN wireless Ethernet proved to be very robust with a net transmission rate of approximately 7 Mbit/sec. Interference or disturbance of other wireless devices, particularly IEEE 802.11b devices, were not observed.

2. The head and the brain of the robots is their own on board Linux notebook. It contains a Pentium III 900 MHz CPU and 256 MB of RAM. Over the years two different Linux distributions, Debian 2.2.r5 and SuSe 7.4, were used. Several processes are running in parallel on this computer and perform tasks like image processing, path planning, action selection and object tracking. Currently, every robot processes approximately 30 frames per second. From every frame a pose estimate, a ball estimate and a set of opponent observations are extracted and sent to all other robots of the team via wireless Ethernet. Every robot runs the opponent tracking algorithm once for every set of opponent observations (own and teammates) and computes a new estimate of the world state. This estimated state serves as input for action selection and path planning.

3. A colour CCD progressive scan camera (Point Grey - Dragonfly) with a wide angle lens (Cosmicar/Pentax TV Lens, opening angle approximately 90^o, focal length $f = 2.8mm$, illumination $1 : 1.2$, cs-mount adapter) is mounted firmly on top of the robot. The camera is connected to the Linux notebook via an IEEE 1394 (FireWire) Cardbus (Sunix 1706). In order to dispense the CPU from receiving the frames, the direct memory access (DMA) mode is used. The camera delivers up to 30 progressive scan frames per second with a resolution of 640 x 480 pixels. The frames are encoded according to a Bayer

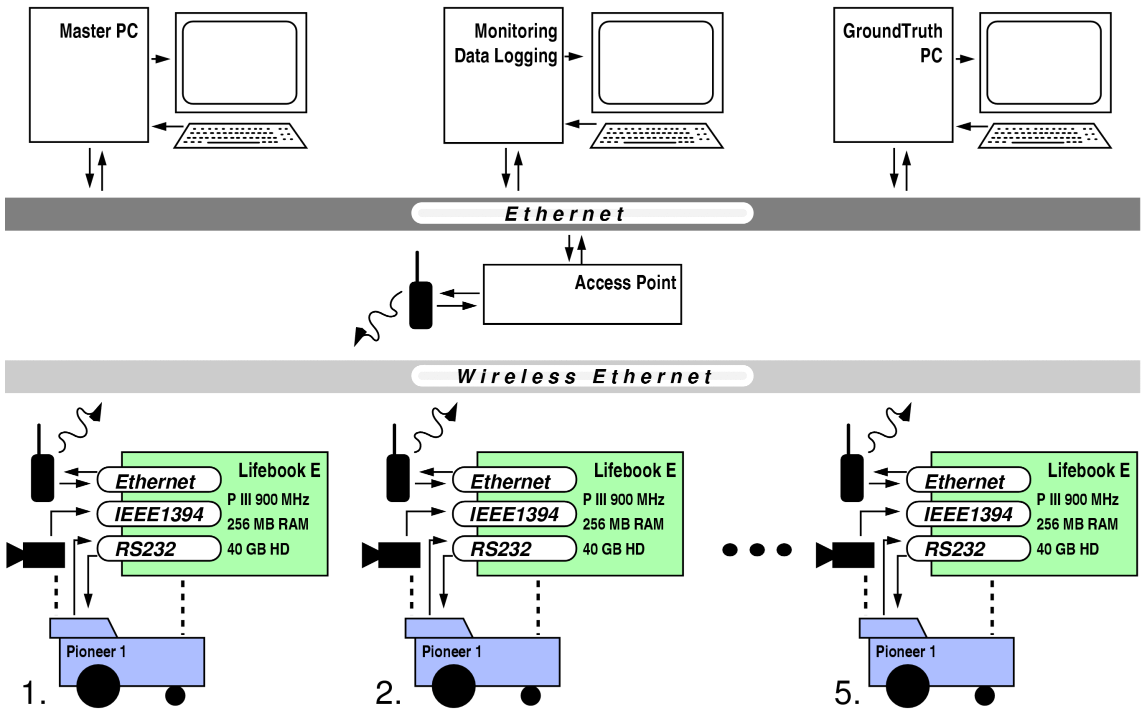


Figure 4.3: The hardware architecture of *The AGILO RoboCuppers*.

pattern colour field array. Thus, down sampling or interpolation of this pattern has to be performed on the on board computer. The camera obeys the nice features that two or three cameras on the same IEEE 1394 bus synchronise automatically. This allows a robot to be equipped with two or more cameras, all of which capturing their frames simultaneously[1]. All algorithms presented in the subsequent chapters are able to handle the information extracted from multiple synchronously captured images without any changes.

4. The sonar sensors were used for collision avoidance. However, due to their extremely noisy measurement they were soon replaced by a more sophisticated vision-based obstacle detection algorithm.

5. A dribbling device is used for ball handling and dribbling.

6. The kicking device can shoot the ball with moderate speed up to three meters.

The overall hardware architecture of *The AGILO RoboCuppers* robot soccer team is displayed in Figure 4.3. Outside the field, two to three master computers are located. They are linked via Ethernet with each other and via a wireless Ethernet with the robots on the pitch. The master computers are used for the development and debugging of the control software running on the robots, sending commands to the robots[2], monitoring the team's performance and logging data about the robots' states, such as their pose estimates and observed positions

[1]Simultaneously captured frames are all captured within a time interval of 3 ms.

[2]During a match, only start and stop commands are sent to the robots.

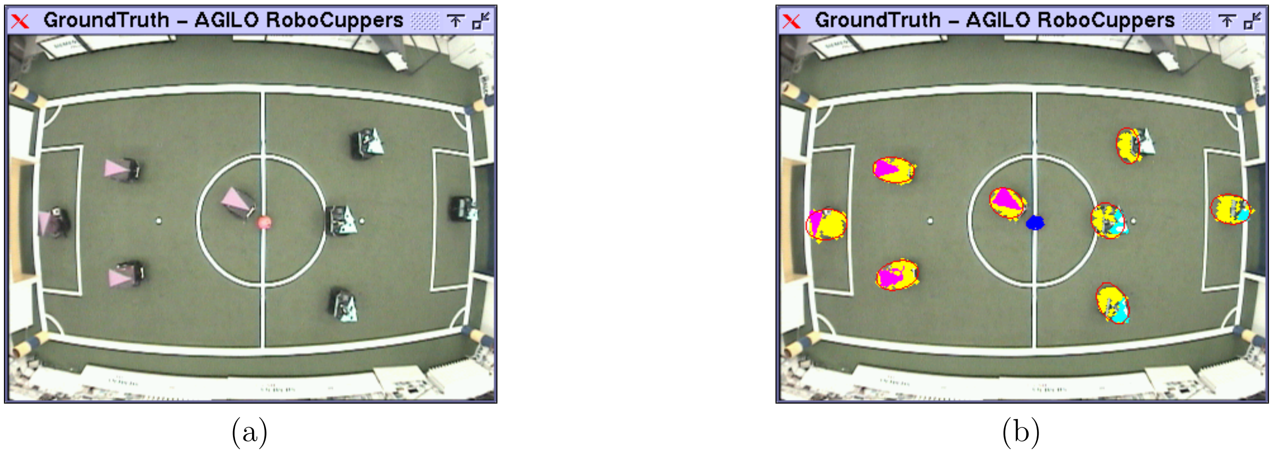

GroundTruth – AGILO RoboCuppers
GroundTruth – AGILO RoboCuppers
(a)
(b)

Figure 4.4: (a) An image captured by the ground truth camera system with (b) extracted regions containing the robots and the detected triangle used in order to determine the orientation of a robot.

of opponent robots. During a match, central processes such as the ball observation fusion module, the opponent tracking module and high-level team coordination modules are also running on the master computers.

A global vision system is mounted above the RoboCup field. The images captured by this ground truth camera are analysed by a process running on a separate computer, see Figures 4.4(a) and (b), and a global belief state is extracted. This data is primarily used for the evaluation of a match and the performance of the teams. For experiments, such as the automatic acquisition of an error model of the localisation or a motion model, this global belief state may also be feed back to the robot.

4.5 Software Architecture

A complex hardware system such as *The AGILO RoboCuppers* requires an efficient, reliable and scalable software architecture that supports the efficient development of software for a multi-robot system. The software architecture of *The AGILO RoboCuppers* is depicted in Figure 4.5. It consists of two main modules: (1) the perception module and (2) the control system. The former contains the subsystems for sensor-data processing and state estimation, while the latter consists of the subsystems for action selection, path planning and low level robot control. The belief state serves as central data structure and communication link between the two main modules.

All modules and subsystems are embedded in the object oriented Sequence/Functor framework (Klupsch, 1998; Klupsch, 2001). In this framework, functionality is implemented in Functor objects. On an abstract level, these objects enclose the functionality for continuously capturing, transforming, or analysing dynamic data. In addition, Functor objects provide application independent practical properties and interfaces, e.g. connections to the input and output Data-Sequences objects, attributes and methods for analysing required computation times, and a general interface for the repeated execution of a set of image processing operators.

Communication among Functors takes place via Data-Sequence objects, which serve as

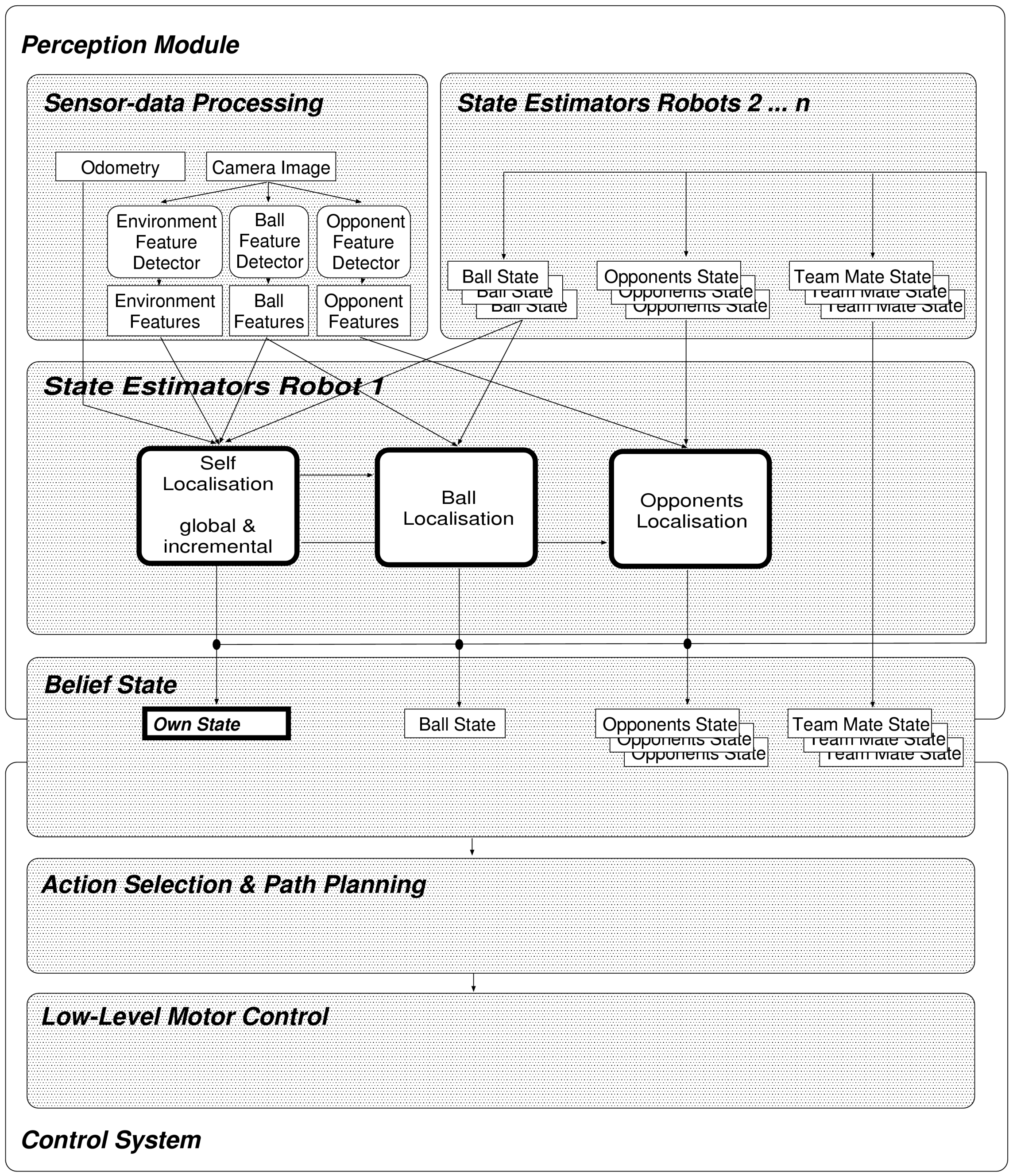


Figure 4.5: Software architecture of the multi-robot team, *The AGILO RoboCuppers*.

input and output streams. Furthermore, Data-Sequences provide general properties and interfaces, which are common for all kinds of data sequences, such as data initialisation, access to current and old values, access to their temporal properties, and methods for updating the Data-Sequence or interpolating values.

With the help of the Sequence/Functor framework, data flow within and among subsystems is easy to implement. It eliminates the need for an explicit specification of a data flow control programme. Functor objects are automatically executed and their output Sequences are updated with the arrival of new input data or upon the request for new output data. This behaviour can be adapted according to topological and functional needs. Overall the Sequence/Functor framework proved and justified its applicability to real time applications, such as multi-sensor multi-robots systems over the years.

The following sections present the perception module and the control system, explain the tasks of the individual subsystems, and the interactions between them. A detailed description of the approach chosen for the individual subsystems will follow in the subsequent chapters and can also be found in (Schmitt, Hanek, and Beetz, 2003; Schmitt and Beetz, 2003; Schmitt et al., 2002; Beetz et al., 2004; Schmitt et al., 2002; Schmitt et al., 2001; Hanek et al., 2000; Bandlow et al., 1999b)

4.5.1 Perception Module

The perception module of each soccer robot receives an asynchronous stream of sensor data and maintains a belief state with respect to its own position on the field, the positions of its teammates, the ball, and the opponent players. Figure 4.5 shows the components of the perception module and its embedding into the *The AGILO RoboCuppers* software architecture.

This module consists of the sensor-data processing subsystem, the state estimator, and the belief state. The sensor-data processing subsystem itself consists of a camera system with several feature detectors and a communication link that enables the robot to receive information from other robots. The belief state contains a pose estimate for every robot of the team and several position estimates for dynamic objects, such as the ball, opponent robots and the referees. All estimates are also associated with a measure of uncertainty, a covariance matrix.

Sensor-Data processing

The sensor-data processing subsystem provides the following kinds of information: (1) feature maps extracted from captured images, (2) odometric information, and (3) partial state estimates broadcast by other robots. The estimates broadcast by the robots of the own team comprise the estimate of the ball's location. In addition, each robot of the own team provides an estimate of its own position. Finally, each robot provides an estimate for the position of every visible opponent. From the captured camera images the feature detectors extract problem-specific feature maps that correspond to (1) static objects in the environment including the goal, the borders of the field, and the lines on the field, (2) a colour blob corresponding to the ball, and (3) the visual features of the opponents.

The working horse of the sensor-data processing subsystem is a colour classification (see Section 5.3.1) and segmentation algorithm that is used to segment a captured image into coloured regions and blobs (see Figure 5.4). The colour classifier is learned in a training session before tournaments in order to adapt the vision system to specific lighting conditions and effects.

Object detection is performed on the basis of blob analysis. The colour segmented image is processed by a feature extraction algorithm (see Section 6.3.1) that estimates the 3D positions with associated uncertainties of the objects of interest. The position of an object is estimated on the basis of a pinhole camera model.

State Estimation

The state estimation subsystem consists of three interacting estimators: the self localisation system, the ball estimator, and the opponents estimator. State estimation is an iterative process where each iteration is triggered by the arrival of a new piece of evidence, a captured image, an odometric measurement or a state estimate broadcast by another robot.

The self localisation (see Section 5.3) estimates the probability density of the robot's own position based on extracted environment features, the estimated ball position, and the predicted position. Two different localisation approaches are used: (1) Monte Carlo Localisation (MCL) and (2) Cooperative Incremental Iterative Localisation (CIIL). MCL is a Monte Carlo Bayes filter and belongs to the class of global self localisation algorithms. It allows a robot to generate a relatively coarse estimate of its initial pose. MCL is based on computational expensive particle filter techniques and cannot be run at frame rate. The MCL-based localisation procedure is not part of this thesis. A detailed description can be found in Neumann (2003). CIIL is a very efficient high-precision pose tracking algorithm, based on least-square estimation and Kalman filter techniques. It can easily run at frame rate. If the CIIL algorithm fails, it is reinitialised with a pose estimate generated by the MCL algorithm. CIIL and MCL are run simultaneously at different repetition frequencies, 30 Hz and 10 - 15 Hz, respectively.

The ball localiser (see Section 5.3.7) estimates the probability density for the ball position, given the robot's own estimated position and its perception of the ball, the predicted ball position, and the ball estimations broadcast by the other robots.

Finally, the positions of the opponents are estimated (see Section 6.3.2) based on the position of the observing robot, the robots' appearances in the captured images, and their positions as observed by the teammates.

4.5.2 Belief State

The central data structure of AGILO software architecture is a global belief state (Aström, 1965), which is maintained by every robot. It serves as a communication link between the perception and the control module and is constructed as follows. The own position, the position of the ball, and the positions of the opponent players are updated by local state estimation processes. The estimated positions of the teammates are the broadcast results of the self localisation processes of the respective teammates. This is done because the accuracy of self localisation is much higher than the accuracy of the position estimation for moving objects.

4.5.3 Control System

The Control System consists of three major subsystems for Action-Selection, Path Planning and Low-Level Robot Control. The subsystems for Action-Selection (Buck, Beetz, and Schmitt, 2001; Buck, Beetz, and Schmitt, 2002b; Buck, Schmitt, and Beetz, 2002) decides which action or behaviour the robot should perform on the basis of the belief state. Usually, an action is associated with a target pose the robot has to achieve. In oder to reach this pose, the Path Planning subsystem (Buck et al., 2001) is used which determines a collision free path. Finally, the Low-Level Robot Control subsystem (Buck, Beetz, and Schmitt, 2002a; Buck et al., 2002) computes motor control commands in such a way that the robots follow the planned paths as accurately as possible and reaches their target poses. *The AGILO RoboCuppers* make use of (1) a Situated Action Selection and (2) a Plan-based Control System.

Situated Action Selection and Execution

Throughout the game, the AGILO robots have a fixed set of tasks with different priorities. The tasks include the following: *shoot the ball into the goal*, *dribble the ball towards the goal*, *look for the ball*, *block the way to the goal*, *get the ball*, etc. The situated action selection module (Buck, Schmitt, and Beetz, 2002) enables the robots to select a task and to carry it out, in such a way that the team's objectives will advance to the most. Action selection and execution is constrained by (1) tasks being achievable only if certain conditions hold (e.g. the robot has the ball) and (2) a robot being able to execute only one action at a time.

A task assignment a_1 is better than a_2 if such a task exists in a_2 that has lower priority than all the ones in a_1 or if they achieve the same tasks but a task t exists in a_1 such that all tasks with higher priority are performed at least as fast as in a_2 and t is achieved faster by a_1 than by a_2. This performance criterion implies that if an AGILO robot can shoot a goal, it will always try because this is the task with the highest priority. Also, if the *The AGILO RoboCuppers* can get to the ball they will try to get there with the robot that can reach the ball the fastest. This strategy might not yield optimal assignments but guarantees that the highest priority tasks are achieved as quickly as possible.

To achieve a high degree of autonomy, the AGILO robots perform the task assignment and execution distributedly on the individual robots. This makes the task assignment more robust against problems in inter robot communication. These problems can be caused by robots being sent off the field, computers being crashed after heavy collisions, and communication being corrupted due to interferences with other communication channels.

The most salient features of the situated action selection are the following ones. First, to realize a competent and fast task assignment and execution mechanism, the AGILO controllers make ample use of automatic learning mechanisms. Second, the task assignment mechanism works distributedly on the individual robots and are robust against communicational disruptions. Finally, the task assignment and execution mechanism always produces purposeful behaviour and always aims at the achievement of high priority tasks.

An important means for developing competent robot soccer skills is a robot simulator that allows for realistic, controllable, and repeatable experiments. For this reason, a robot simulator (Buck et al., 2001; Buck, Beetz, and Schmitt, 2002b) has been developed that accurately simulates how the dynamic state of a robot changes as the robot's control system issues new driving commands, such as setting the target translational and rotational velocities.

Plan-based Control

While situated action selection aims at choosing actions that have the highest expected utility in the respective situation, it does not take into account a *strategic* assessment of the alternative actions and the respective *intentions* of the teammates. This is the task of the plan-based action control. While situated action selection achieves an impressive level of performance, it is still hampered by the requirement for small action and state spaces, a limited temporal horizon, and without explicitly taking the intentions of the teammates into account.

The goal of plan-based control in robotic soccer is, therefore, to improve the performance of the robot soccer team by adding the capability of learning and execute soccer plays. Soccer plays are properly synchronised, cooperative actions that can be executed in certain game contexts and have, in these contexts, a high success rate. Plans for soccer plays specify how the individual players of a team should respond to changing game situations in order to perform the play successfully.

The integration of soccer plays into the game strategies enables robot teams to consider play specific state spaces for action selection, parameterisation, and synchronisation. In addition, the state space can reflect the intentions of the other robots. An action that is typically bad might be very good if a robot knows that its teammate intends to make a particular move. Further, action selection can consider a wider time horizon, and the robots can employ play specific routines for recognising relevant game situations.

In order to realize an action assessment based on strategic consideration and on considerations of the intentions of the teammates, a robot soccer play book has been developed, which is a library of plan schemata that specify sequences of actions to be performed by the individual robots. The plans, or better plays, are triggered by opportunities, e.g. the opponent team leaving one side open. The plays themselves specify highly reactive, conditional, and properly synchronised behaviour for the individual players of the team.

The high-level controller of each soccer robot is realized as a structured reactive controller (SRC) (Beetz, 2001) and implemented in an extended RPL plan language (McDermott, 1991). The high-level controller works as follows. It executes the situated action selection as the default strategy. At the same time, the controller continually monitors the estimated game situation in order to detect opportunities for executing a play scheme. If an opportunity is detected, the controller decides based on circumstances including the score and the estimated success probability of the intended play scheme whether it is performed or not.

4.6 Conclusions

This chapter described the RoboCup initiative and discussed the hardware and software approach chosen by *The AGILO RoboCuppers* autonomous robot soccer team. Similar to advanced autonomous robotic agents acting in human working environments, *The AGILO RoboCuppers* employ sophisticated state estimation and control techniques, including experience-based learning and plan-based control mechanisms.

It was shown that the application of probabilistic state estimation techniques together with information exchange between the robots result in game state estimators that are capable of estimating complete and complex states. It was also shown that the amble use of experience-based learning has resulted in powerful control mechanisms, including competent coordination, with little runtime computational cost. It was explained how plan-based control mechanisms can enhance the robot's playing skills by enabling the robots to perform complex soccer plays. The results of the 2001 robot soccer world championship showed that these techniques allow for competitive soccer play despite an inferior hardware equipment.

Overall, the approach of *The AGILO RoboCuppers* has proven to be applicable and successful during five RoboCup World Championships and three German Championships from 1998 until 2002. Furthermore, they have performed at several fairs, such as the Systems 2001 (Munich/Germany), The Science Festival 2000 (Freiburg/Germany) and the Deutsche Arbeitsgemeinschaft für Mustererkennung 2001 (DAGM/Munich/Germany), and demonstrated their abilities in various friendly matches. *The AGILO RoboCuppers* appeared in several newspaper articles, scientific publications, and two short movies: "The RoboCup Story" (2001) is a commercial for technological innovation that was shown at the Systems 2001 computer fair. A documentation about *The AGILO RoboCuppers* was broadcast in 2001 on the German TV channels BR3 and 3sat.

Chapter 5

Cooperative Incremental Iterative Localisation

5.1 Introduction

The most fundamental problem a robot soccer player has to solve is to determine its own pose and the position of the ball on the field. In highly dynamic environments, such as the RoboCup scenario, these tasks do not only have to be achieved in real time but also with high accuracy. A further constraint is imposed by the fact that small and inexpensive sensors are used in order to keep size, weight and cost of a robot soccer player low. Currently, to tackle this problem the most universal sensor that can be used is a digital video or CCD camera. However, their comprehensive advantages are opposed by the huge amount of data provided by them. This confronts the sensor data processing algorithms with a tough challenge if they have to be real time capable.

This chapter presents a fast (real time capable) and accurate vision-based self- and ball-localisation procedure, called the Cooperative Incremental Iterative Localisation (CIIL) algorithm. CIIL allows a robot to track its own pose and the ball position with high precision. Based on a model of the environment and the observations performed by the robot, the CIIL algorithm iteratively estimates the probability density over the possible robot positions. A further advantage of the CIIL algorithm is that it is also capable to take observations into account (e.g. ball observations), performed by other robots. By doing this, a team of robots performs cooperative localisation and is able to solve localisation problems a single robot would not be able to solve on its own.

In the remainder of this chapter the CIIL algorithm is presented and evaluated. Section 5.2 presents the belief state and the environment model used by the CIIL algorithm. The CIIL algorithm and its implementation are described in Section 5.3. Related work is briefly described in Section 5.4. A summary and discussion is given in Section 5.5 and this also concludes the chapter. The algorithm and its properties are extensively evaluated in a series of real world experiments in Chapter 7.

5.2 Belief State, Pose Estimate and Environment Model

Before the Cooperative Incremental Iterative Self-Localisation algorithm is described in detail, this section will briefly review its central data structures, used to represent the belief state and the environment model.

5.2.1 Pose and Ball Estimate

The first data structure used by the CIIL algorithm is the combined estimate of a robot's pose and the relative ball position. In the context of this thesis, the pose describes the position of the robot's vertical rotation axis in world coordinates and its orientation. The rotation axis of an AGILO robot is located in the middle of the two driving wheels and does not coincide with the centre of the robot's body.

The belief, $Bel(\widehat{x}_t)$, of the CIIL algorithm is approximated by a multi-variate Gaussian random variable, $x \sim N(\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$. The mean vector, $\bar{\mathbf{x}}$, represents the joined pose and position estimate of the robot and the ball respectively, while the covariance, $\Sigma_{\mathbf{x}}$, represents the associated uncertainty of this estimate. Given the above definitions, the belief of the CIIL algorithm can be represented as follows:

$$Bel(\widehat{x}_t) = N_t(\widehat{x}_t; \bar{\mathbf{x}}_t, \Sigma_{\mathbf{x}_t}) \tag{5.1}$$

The implementation of the algorithm allows the use of two different representations of the mean, $\bar{\mathbf{x}}$:

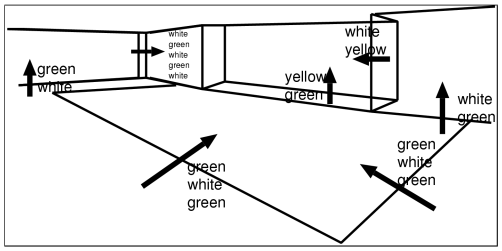
$$\begin{aligned} \bar{\mathbf{x}}_1 &= (x, y, \phi_z, x_{ball}, y_{ball})^T \\ \bar{\mathbf{x}}_2 &= (x, y, z, \phi_x, \phi_y, \phi_z, x_{ball}, y_{ball})^T \end{aligned} \tag{5.2}$$

The first, $\bar{\mathbf{x}}_1$, represents the robot's pose and the relative ball position. The associated covariance matrix, $\Sigma_{\mathbf{x}_1}$, is a 5×5 matrix. This representation is used by *The AGILO RoboCuppers* since 1998.

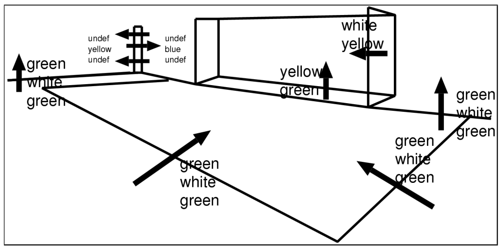
The second representation, $\bar{\mathbf{x}}_2$, is more general. In addition to the first representation, it also contains the height of the robot's body and the two remaining rotational degrees of freedom about the horizontal and longitudinal axes. This representation describes the position of a robot in 3D Cartesian coordinates and with all three rotational angles. The associated covariance matrix, $\Sigma_{\mathbf{x}_2}$, is a 8×8 matrix. This representation is used for robots with more degrees of freedom, such as the Sony AIBO robot dog.

5.2.2 Environment Model

The robot's model of the static part of its environment is composed of landmarks together with their positions and orientations. The landmarks themselves are described by 3D curve functions that specify edge curves. Edge curves represent colour transitions, i.e. they separate



(a)



(b)

Figure 5.1: Model of the neighbourhood of a goal (a) until 2001 (b) from 2002. The model contains the edges of the objects and the associated colour transitions.

image regions that differ in the distribution of colour vectors. A 3D curve feature is represented by a pair consisting of a curve function and a colour transition.

In the context of the RoboCup scenario two types of curve functions are distinguished. An edge curve separates two image regions which differ in the distribution of colour vectors. An edge is specified by a single curve function. In the RoboCup scenario edges occur at the goals and the corner flagposts. For a field line, two curve functions are used which not only describe the position of the line but also its width. Using the world model, a camera model and a pose estimate, the robot is able to predict the positions of colour transitions in a captured image.

Figure 5.1 depicts excerpts of the environment model representing the neighbourhood around a goal, which is used for self localisation in the robot soccer domain. The goal is modelled as a set of 3D lines, where each line is associated with a colour transition.

The individual features of the model are described by curve functions $C_i : D_i \rightarrow \mathcal{R}^3$. A curve function C_i defines the set $\mathcal{G}(C_i)$ of curve points in 3D world coordinates by $\mathcal{G}(C_i) = \{C_i(s) | s \in D_i\}$, where $D_i = [s_{i,min}, .., s_{i,max}]$ is the domain for s. All common curves, such as circles or B-splines, can be specified or at least approximated by curve functions. A straight line segment is simply given by

$$C_i(s) \quad = \quad s \cdot B_{i1} + (1 - s) \cdot B_{i2} \tag{5.3}$$

where the points B_{i1} and B_{i2} are the endpoints of the line segment and D_i is equivalent to $[0, .., 1]$. A horizontal circle is defined by

$$C_i(s) \quad = \quad M_i + r \cdot (\sin(s), \cos(s), 0)^T \tag{5.4}$$

where M_i is the centre point, r is the radius, and $D_i = [0, .., 2\pi]$. Line features with multiple colour transitions, see Figure 5.1, are defined by multiple curve functions.

The edges of the corner cylinders (flagposts) are described as follows (see Figure 5.2):

$$C_i(s) \quad = \quad C_{center_i}(s) + r * \left(\mathcal{R}_z \left(r_\gamma \right) * \frac{d}{|d|} \right) \tag{5.5}$$

where $C_{center_i}(s) = C$ represents the centre axis of the cylinder (see Eq. (5.3)), r its radius, $v = OC$ the vector from the camera's optical centre, $O(x)$, to the cylinder's centre axis, $C_{center_i}(s)$,

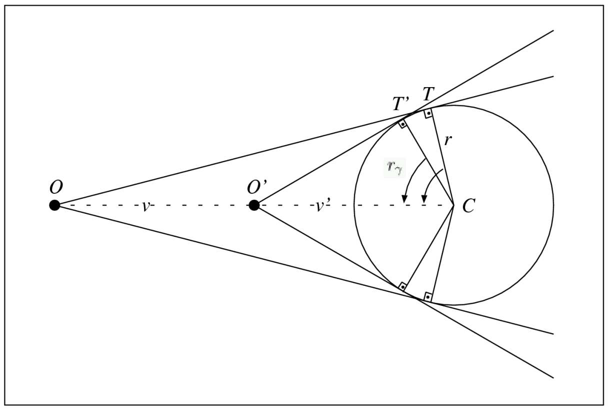


Figure 5.2: The width of the cylinder in an image and, thus, the position of its edges are a function of the distance between the camera's optical centre, O or O', and the position of the cylinder's centre axis, C.

$$d = O(x) - C_{center_i}(s) \tag{5.6}$$

and $r_\gamma = \angle TCP$ the angle between the perpendicular on the tangent point, T, the cylinder's centre axis, C, and the camera's optical centre, O.

$$r_\gamma = \pm\left(\frac{\Pi}{2} - asin\left(\frac{r}{|d|}\right)\right) \tag{5.7}$$

It is noteworthy that for r_γ two possible solutions exist, depending on the sign they represent, the tangent to the left side or to the right side of the cylinder. The camera's optical centre, $O(x)$, can be determined from the robots pose, x, through the inverse of a simple coordinate transformation (see also Section 2.4.2)

$$O(x) = \mathcal{R}_v^{-1}(x) + \mathcal{T}_v = \mathcal{R}_v^T(x) + \mathcal{T}_v \tag{5.8}$$

5.3 The Cooperative Incremental Iterative Localisation Algorithm

The Incremental Iterative Self-Localisation (CIIL) algorithm is used to track the pose of an AGILO player and the ball. It is initialised with a pose estimate generated by a global self-localisation procedure (Neumann, 2003), which is based on a MCBF (see also Section 4.5.1). The CIIL algorithm then refines this pose and tracks the robots pose and the ball until it fails and requires reinitialisation. By providing two self-localisation procedures computational resources are saved. The fast and accurate CIIL runs at frame rate (30 Hz), while the global self-localisation runs at 10 Hz or less.

```
algorithm SELF LOCALISATION ($<\bar{\mathbf{x}}, \Sigma_{\mathbf{x}}>, data$)
let
    $<\bar{\mathbf{x}}, \Sigma_{\mathbf{x}}>$      % previous belief state <mean,covariance>
    $<\widetilde{\mathbf{x}}, \Sigma_{\widetilde{\mathbf{x}}}>$      % predicted belief state <mean,covariance>
    $<\widehat{\mathbf{x}}, \Sigma_{\widehat{\mathbf{x}}}>$      % updated belief state <mean,covariance>
    $data$              % data item (action or signal)
    $P$                 % set of projected points
    $\widetilde{P}$                 % set of correspondences (correspondence and uncertainty)

do
    $<\widetilde{\mathbf{x}}, \Sigma_{\widetilde{\mathbf{x}}}> \leftarrow$ APPLY MOTION MODEL ($\bar{\mathbf{x}}, \Sigma_{\mathbf{x}}$);
    $<\widehat{\mathbf{x}}, \Sigma_{\widehat{\mathbf{x}}}> \leftarrow (\widetilde{\mathbf{x}}, \Sigma_{\widetilde{\mathbf{x}}})$;
    loop
        switch ($data$)

            case ($data$ is an $image$) :
                $P \leftarrow$ PROJECT CURVE POINTS ($\widehat{\mathbf{x}}$);
                $\widetilde{P} \leftarrow$ SEARCH CORRESPONDENCES ($P, image$);
                $\chi_I^2 \leftarrow$ QUALITY OF FIT IMAGE ($\widetilde{P}, \widehat{\mathbf{x}}$);
                $\chi^2 \leftarrow \widetilde{\chi}^2 + \chi_I^2$;

            case ($data$ is odometric data, $odometry$) :
                $\chi_O^2 \leftarrow$ QUALITY OF FIT ODOMETRY ($odometry, \widehat{\mathbf{x}}$);
                $\chi^2 \leftarrow \widetilde{\chi}^2 + \chi_O^2$;

            case ($data$ is a ball observation from a teammate, $ball$) :
                $\chi_T^2 \leftarrow$ QUALITY OF FIT TEAM ($ball, \widehat{\mathbf{x}}$);
                $\chi^2 \leftarrow \widetilde{\chi}^2 + \chi_T^2$;

        % step of the Newton minimisation
        $\widehat{\mathbf{x}} \leftarrow \arg\min_x \chi^2$;

    until (change of $\widehat{\mathbf{x}}$ is small or $data \neq image$)

    $\Sigma_{\widehat{\mathbf{x}}} = 2\, \mathbf{H}(\chi^2)^{-1}$;

return ($<\widehat{\mathbf{x}}, \Sigma_{\widehat{\mathbf{x}}}>$);
```

Figure 5.3: The Cooperative Incremental Iterative Self- and Ball-Localisation (CIIL) algorithm integrates image data, odometric data, and observations of teammates over time to a MAP estimate.

The CIIL algorithm (see Figure 5.3) running on each robot integrates three different types of data: (1) image data, (2) odometric data, (3) ball observations made by other teammates. These data items are integrated over time to a maximum a posteriori (MAP)[1] estimate of the robot's pose. First, the parts of the self-localisation algorithm, which are identical for all three data types, are described. Then, the three cases corresponding to the three data types (lines 15 to 27 in Figure 5.3) are presented in the remainder. The MAP estimate $\widehat{\mathbf{x}}$ of the CIIL algorithm is given by

$$\begin{aligned} \widehat{\mathbf{x}} &= \arg\max_x \; p(x|data) \\ &= \arg\max_x \; \nu \cdot p(x) \cdot p(data|x) \end{aligned} \tag{5.9}$$

where ν is a normalising constant, $p(data|x)$ is the observation model and $p(x)$ is the prior of the pose x summarising all evidence gathered in the past.

The prior $p(x)$ at the current time step is obtained by predicting the pose distribution, estimated for the previous time step. The constant velocity motion model used in line (10) assumes that the robot continues with constant rotational and translational velocity. The associated covariance is propagated using a linear approximation of the motion model.

The second term $p(data|x)$, in equation (5.9), is the likelihood that the robot received *data* given its current pose x. The computation of the likelihood corresponding to the three data types (I = image data, O = odometric data, T = data from teammates) is explained below. The optimisation of the product in equation (5.9) can be transformed into a numerically more favourable optimisation of a sum by taking the negative logarithm:

$$\begin{aligned} \widehat{\mathbf{x}} &= \arg\min_x \; \ln(\chi^2) \\ &= \arg\min_x \; \ln(\widetilde{\chi}^2) + ln(\chi^2_{data}) \\ &= \arg\min_x \; -2\ln\left(p(x)\right) - 2\ln\left(p(data|x)\right) \end{aligned} \tag{5.10}$$

The function $\widetilde{\chi}^2$ evaluating the fit between the pose x and the prior is given by

$$\widetilde{\chi}^2 = (\widetilde{\mathbf{x}} - x)^T \Sigma_{\widetilde{\mathbf{x}}}^{-1} (\widetilde{\mathbf{x}} - x) + C \tag{5.11}$$

where $\widetilde{\mathbf{x}}$ is the mean vector and $\Sigma_{\widetilde{\mathbf{x}}}$ is the covariance matrix of the prior. The constant $C = \ln|2\pi\, \Sigma_{\widetilde{\mathbf{x}}}|$ is independent of the pose x. Hence, it can be omitted when χ^2 is optimised for x using Newton iteration (Press et al., 1996). After the optimisation of χ^2, the covariance matrix $\Sigma_{\widehat{\mathbf{x}}}$ characterising the uncertainty of the robot's pose is estimated according to:

$$\Sigma_{\widehat{\mathbf{x}}} = 2\, \mathbf{H}(\chi^2)^{-1} \tag{5.12}$$

[1]Bar-Shalom and Fortmann (1988) proved in their book (pages 12f and 64f), that a MAP estimate of Gaussian pdfs is equivalent to the estimate of a KF. However, for this thesis the notation of MAP estimation was adopted because of its analogy to Bayes estimation and Bayes filtering.

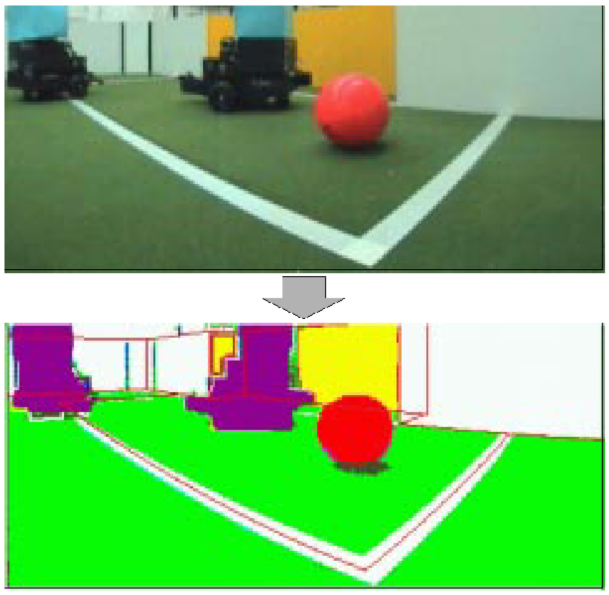


Figure 5.4: Image captured by the robot (top) and the colour labelled image (bottom). The feature maps, computed for self, ball, and opponent localisation, correspond to the different colour labels.

where $\mathbf{H}(\chi^2)$ is the Hessian of the objective function χ^2 in the estimate $\widehat{\mathbf{x}}$ (Press et al., 1996).

In all three cases (see Figure 5.3) a distinct function evaluates the fit between the pose and the latest data. For image data, the corresponding χ_I^2 function is obtained by the following steps. In line (16) the 3D curve features, which are predicted to be visible, are projected into the image plane. In line (17) a local search is performed to establish correspondences between the predicted and detected 3D curve features. In line (18) the function χ_I^2 is obtained by evaluating the distances between the projections and the found image points. Finally, in line (29) the resulting maximum a posteriori criteria is optimised for the robot pose. For odometry and ball data, the corresponding $\chi_{O,T}^2$ function can directly be determined and no optimisation steps are required. In both cases the optimal solution is given by a weighted sum of the prediction and the observation[1].

In the following Sections, the implementation of the single steps of the CIIL algorithm are presented.

5.3.1 Image Preprocessing and Feature Extraction

The CIIL algorithm and the CODT algorithm (see Chapter 6) use the colour information provided by the objects of the RoboCup scenario to identify objects, to perform ball and opponent observations, and to estimate the states of the ball, the own robots, and the opponent robots. In order to make efficient use of the colour information provided, a fast preprocessing and colour classification procedure is required. This section will describe a fast approach for colour classification and for colour-based image segmentation in a RoboCup scenario.

In middle size robotic soccer all object categories have different colours: the ball is red, field lines are white, and the goals are yellow and blue. These rules allow for the application of a simple method for *object recognition*: segment the image into coloured blobs and identify the blobs based on their colour (see Figure 5.4). The colour classifier is learned in a training session before a tournament in order to adapt the vision system to the specific light conditions.

A RGB-16 colour vector is classified with the help of a look-up table. The use of a look-up table has two advantages: (1) The classification is very fast and (2) no assumptions about the classes' shapes have to be made. The look-up table is constructed in a two step process: First, an initial look-up table is obtained from classified training images. This look-up table is sparse, since usually most of the 2^{16} RGB vectors are not observed in the classified images. Second, the unclassified RGB vectors are classified using a nearest-neighbour classifier. The Euclidean distances are computed in a 4D space defined by the transformation:

$$\begin{aligned} T(R,G,B) &= \left(\frac{R}{I},\frac{G}{I},\frac{B}{I},\frac{R+G+B}{3C_2}\right)^T \\ \text{where} \quad I &= \frac{R+G+B}{3}+C_1. \end{aligned} \tag{5.13}$$

The constant $C_1 > 0$ is chosen to be quite small, such that the first three components are roughly independent of the intensity. Only for RGB vectors close to the colour black the constant C_1 is important by ensuring a smooth transformation. Other common colour spaces such as HSI cause unwanted discontinuities for unsaturated colour values. The fourth component of T describes the intensity, which is also valuable in order to separate the colour classes given in RoboCup. However, within one class usually the intensities vary more than the normalised values of the first three elements. Therefore, the intensity is normalised by C_2, e.g. $C_2 = 10$.

It is noteworthy that the CIIL algorithm only requires pixels to be classified that are accessed during the search for model curves in the image. This constitutes a big advantage and speed up over traditional methods, like edge detection or Hough transformation, which require the whole image to be processed.

5.3.2 Projecting 3D Curve Points into the Image

Before the CIIL algorithm can search for correspondences between the environment model and the image it has to determine a set of starting points in pixel coordinates.

In order to accomplish this task, points of visible 3D curve features are projected into the image plane (see line (16) in Figure 5.3). The observation of a 3D curve $\mathcal{G}(C_i)$ in an image is a 2D image curve

$$\mathcal{G}(c_i) = \{c_i(s,x) | s \in [0,1]\} \tag{5.14}$$

where x is the robot pose and c_i is the image curve function given by

$$c_i(s,x) = \; proj(C_i(s),x) \tag{5.15}$$

Index s is used to specify the point the curve function is evaluated for. For example, $C_i(0.0)$ and $C_i(1.0)$ determine the 3D world coordinates of the starting and ending points of the 3D

Figure 5.5: Search for correspondences. (a) Due to the occlusion caused by the robot, for one projected point no corresponding image point is found with the required colour transition. (b) By comparing the line width in the image with the line width of the projection, wrong correspondences (crosses) can be detected.

curve, respectively. The function *proj* (see Section 2.4) projects a 3D point, given in world coordinates, into the image and returns the pixel coordinates of the resulting 2D point. This mapping assumes that an estimate of the robots pose, the relative displacement and rotation between the robot pose and the camera, as well as the intrinsic camera parameters are known.

First, the mapping *proj* converts the 3D world coordinates into 3D robot coordinates (see Eq. (2.21)). This conversion depends on the robot's pose x. Then, the 3D robot coordinates are transformed into 3D camera coordinates (see Eq. (2.20)). This transformation is straight forward, as the relative displacement between the camera and the centre of the robot can be measured in advance. In the next step, the 3D camera coordinates are projected into the image plane according to Eq. (2.22), the radial lens distortions are taken into account with the application of Eq. (2.24) and finally, the discrete pixel coordinates can be computed (see Eq. (2.25)). The resulting image curve function c_i describes the relation between the robot pose x and the position of the model curve points in the image.

5.3.3 Searching for Correspondences

In this step the CIIL algorithm tries to establish correspondences between model and image curves. Starting from the projected image coordinates it searches along the perpendiculars for colour transitions associated with the 3D model features.

For each visible model curve, $\mathcal{G}(C_i)$, a small set, P, of projected curve points, $P_{i,j}$, is determined. For each of these projected curve points, $P_{i,j}$, the algorithm searches for colour transitions, $\widetilde{P}_{i,j}$, which are consistent with the transitions specified by the model (see line (17)). To find the respective transition points, $\widetilde{P}_{i,j}$, in the image the algorithm locally searches along the perpendiculars of the model curve starting at the projected curve point $P_{i,j}$ (see Figure 5.5(a)). This search is performed within a fixed search area. Image points outside the search area are regarded as outliers and are omitted. While this is appropriate for the RoboCup environment, search areas adapted to the uncertainty of the curve as in (Blake and Isard, 1998) could be beneficial in more complex environments.

For each accepted point, $\widetilde{P}_{i,j}$, found in the image, a standard deviation, $\sigma_{i,j}$, is used which describes the expected precision of the observation along the perpendicular. In the RoboCup scenario for each $\widetilde{P}_{i,j}$ a constant value of $\sigma_{i,j}$ is used. This value was determined through a series of experiments. However, for less restricted scenes it might be necessary to use

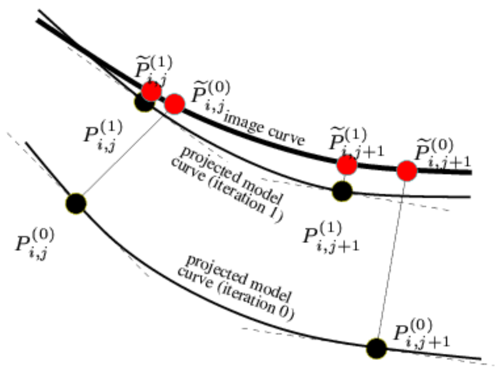

$\widetilde{P}^{(1)}_{i,j}$
$\widetilde{P}^{(0)}_{i,j}$
image curve
$\widetilde{P}^{(1)}_{i,j+1}$
$\widetilde{P}^{(0)}_{i,j+1}$
$P^{(1)}_{i,j}$
projected model curve (iteration 1)
$P^{(1)}_{i,j+1}$
$P^{(0)}_{i,j}$
projected model curve (iteration 0)
$P^{(0)}_{i,j+1}$

Figure 5.6: One iteration step of the CIIL algorithm and the new projected model curve moved closer to the actual image curve.

different $\sigma_{i,j}$. For example, individual $\sigma_{i,j}$ can consider information provided by the local image gradient.

The lines on the soccer field are treated as double colour transitions. The two curve functions defining a line curve are used to predict the line width in the image. By comparing the predicted width with the observed width, wrong correspondences can be rejected (see Figure 5.5(b)).

5.3.4 Evaluating the Fit between an Image and the Robot's Pose

To evaluate the quality of a fit between an image and the robot pose, a χ^2_I error function is defined. The evaluation given by χ^2_I is based on the distances between points $\widetilde{P}_{i,j}$ found in the image and the corresponding model curves c_i.

The accurate distance between a point and a curve is defined by a nonlinear function and requires usually a nonlinear optimisation. For efficiency reasons the curve is locally approximated by its tangent. The displacement (signed distance) $d_{i,j}(x)$ between an observed image point $\widetilde{P}_{i,j}$ and the corresponding tangent of the projected model curve c_i is

$$d_{i,j}(x) = (n_{i,j}(x))^T \cdot \left[c_i(s_{i,j}, x) - \widetilde{P}_{i,j}\right] \tag{5.16}$$

where $n_{i,j}(x)$ is the normal vector of the curve c_i in the point $c_i(s_{i,j}, x)$.

The observations $\widetilde{P}_{i,j}$ are noisy. The displacements between the observations $\widetilde{P}_{i,j}$ and the real image curve are assumed to be statistically independent and Gaussian distributed with a mean value of 0 and covariances $\sigma^2_{i,j}$. Under these assumptions, the observation model, i.e. the likelihood for an observation given the robot's pose x, is defined as:

$$p\left(\widetilde{P}|x\right) = \prod_{(i,j)} \frac{1}{\sqrt{2\pi}\sigma_{i,j}} \exp\left(\frac{-d^2_{i,j}(x)}{2\sigma^2_{i,j}}\right) \tag{5.17}$$

where $\widetilde{P}$ denotes the set of all curve points $\widetilde{P}_{i,j}$ found in the image. The function χ^2_I evaluating the fit between an image and the robot pose is defined as:

$$\chi_I^2 = -2 \ln p \left(\widetilde{P} | x \right) \tag{5.18}$$

5.3.5 Optimising the Robot's Pose

Iterative optimisation enables the CIIL algorithm to take nonlinearities of the observation model into account. The nonlinearities of the model curve functions are caused by different reasons, such as dependence on the robot's orientation and projection into the image plane. This way, the self-localisation algorithm can avoid inaccuracies typically yielded by other state estimation algorithms such as the extended Kalman filter (e.g. see (Faugeras, 1993)).

In line (30) of the self-localisation algorithm, a single Newton iteration step (Press et al., 1996) minimising the objective function χ^2 is performed. The Newton step is initialised by the current pose estimate $\widehat{\mathbf{x}}$ and yields a new estimate. Figure 5.6 illustrates the result of a single iteration step. After one iteration step, the new projected model curves are closer to the observed image points. However, the projected model points $P_{i,j}$ are shifted along the model curves. Hence, these new projected points do not correspond to the initial observations $\widetilde{P}_{i,j}$. Therefore, with every iteration the model points have to be projected again, and a new search for the corresponding colour transition is performed along the perpendiculars. Since the deviation between image curves and projected model curves is already reduced, the new search can be performed at clearly lower computational cost. The process is iterated until the change of the estimate $\widehat{\mathbf{x}}$ is smaller than a predefined value.

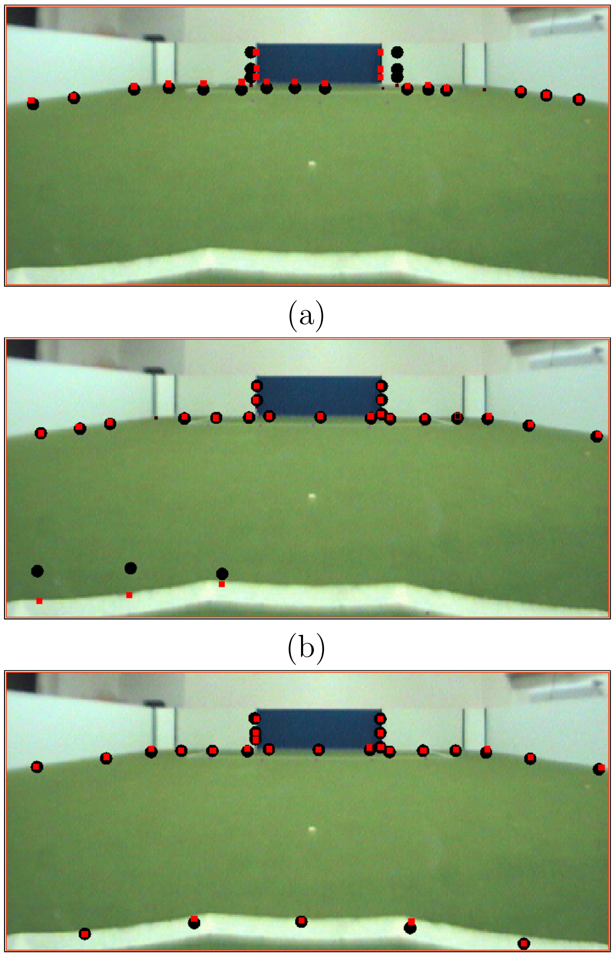
Since the pose consists of three variables, three independent displacements $d_{i,j}$ corresponding to three image point $\widetilde{P}_{i,j}$ are sufficient to estimate the pose. However, in order to increase robustness and accuracy, for each visible feature at least three correspondences are established.

For the proposed method usually only few iterations (about two or three) are necessary. Since the computational cost for the self-localisation is much smaller than the cost for the blob analysis, the overall cost increases only slightly by using multiple iterations. In general, the higher computational cost is well compensated by a higher accuracy, especially in dynamic scenes where predictions are quite uncertain.

Figure 5.7 depicts the results for a sequence of iteration steps. The result after one iteration step (see Figure 5.7(a) and (b)) is identical to the estimate of an extended Kalman filter. This estimate is substantially improved by the new nonlinear method using multiple iteration steps (see Figure 5.7(b) and (c)).

5.3.6 Incorporation of Odometric Data

Similarly to the image data, in line (18) a function χ_O^2 is used for evaluating the fit between consecutive odometric sensor readings and consecutive robot poses. The last two sensor readings are used to determine a Gaussian random variable, $u \sim N(\bar{\mathbf{u}}, \Sigma_{\mathbf{u}})$, representing the relative motion of the robot. The mean, $\bar{\mathbf{u}} = (\Delta s, \Delta\phi)^T$ consists of the distance Δs travelled and the change in orientation $\Delta\phi$ between the last two robot poses. It is assumed that the errors of u are zero-mean Gaussian distributed, mutually statistically independent, and statistically independent of the noise corrupting the image measurements. Consequently the errors can be represented with a 2×2 covariance matrix, $\Sigma_{\mathbf{u}}$.



(a)

(b)

(c)

Figure 5.7: Illustration of a sequence of iteration steps. Dots indicate projected model points, squares depict corresponding image points. (a) Initialisation. (b) The result of one iteration step is equivalent to the estimate of an extended Kalman filter. (c) Further iterations can yield significant improvements.

The resulting function χ_O^2 is quadratic in the robot's pose. Hence, for the optimisation of the objective function, see line (23) in Figure 5.3, only one iteration step is required. In fact only the weighted sum of the predicted pose and the odometric pose has to be computed. The weights are determined by the respective covariance matrices.

5.3.7 Ball-Localisation

In the RoboCup scenario the ball is a unique and robust observable object. Typically it moves slowly and all robots participating in a match try to face and to observe the ball. This suggests not only to estimate the position of the ball, but also to regard the ball as an additional dynamic landmark that can be used for self-localisation.

On one hand, the estimated world coordinates of the ball depend on the world coor-

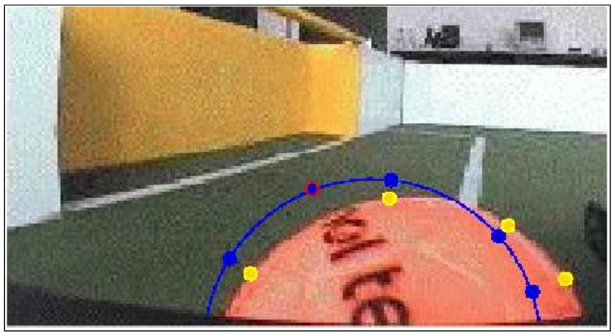


Figure 5.8: Search for image points along the perpendiculars. The ball's contour is projected into the image and the algorithm establishes correspondences for several selected points (big filled circles). For one image point no correspondence is found, due to the inscription on the ball.

dinates of the observing robot. On the other hand, knowledge of the ball-position can be used for self-localisation of the observing robot. In order to exploit these interdependencies, the self-localisation and the localisation of the ball are performed simultaneously in a single optimisation.

For ball localisation a similar approach is used as for self-localisation. The silhouette of the ball is described by a curve function $c(x)$ (see Figure 5.8). Here, the vector x to be estimated contains the pose of the observing robot and the position of the ball. The optimisation is done (as for the self-localisation) simultaneously over all curves, no matter whether a curve feature belongs to the static world model or to the ball. The initialisation of the ball position is obtained from an analysis of red blobs. Several consistency checks are applied testing the shape and the relation between the size and the ball distance. The biggest accepted blob is back-projected, which yields the initial ball position.

After a robot has updated its estimate $\hat{\mathbf{x}}$, it broadcasts a second order Taylor approximation χ^2_T of the part of the optimised MAP objective function, which depends on the latest observations. A receiving robot r updates its world model by including the approximation χ^2_T into its own estimation. If the same robot r afterwards observes the ball, the knowledge of the vague ball position is used in order to refine the own pose. Robot r uses the ball as a dynamic landmark. In order to predict the position of the ball, a linear motion model is used. Section 7.3.2 in the experimental chapter will demonstrate that certain ambiguous localisation problems can uniquely be solved with the help of the ball.

5.4 Related Work

As mentioned earlier, a large body of research in solving the mobile robot localisation and pose tracking problem exists. Roughly, all techniques can either be assigned to the class of outlier sensitive estimation techniques or to the class of robust estimation techniques. In the following sections, different variants of these two main approaches, and their application to various types of sensors within the RoboCup domain and other robotic applications, are discussed.

5.4.1 Outlier Sensitive Estimation Techniques

In general, estimation techniques that are sensitive to outliers, false correspondences, false observations, and false positives are very accurate as long as observations are noise free and require only limited computational resources. These are probably the main reasons why they are still widely used for state estimation in autonomous robot systems.

The three most commonly used techniques include Kalman filter, MAP estimators, and least square estimators. In the past, all these techniques have been applied to different types of sensors in order to estimate the pose of an autonomous mobile robot. In the following, approaches for laser range finders, cameras, and omnidirectional vision systems are discussed.

Laser Range Finder

A family of algorithms that is related to the CIIL algorithm uses laser range data for robot localisation and navigation. It is their task to match laser range scans, which consist of data points with range and bearing information of potential obstacles, with (1) a given map, or (2) a previously recorded scan or set of scans.

The approach of Cox: An algorithm for solving the former problem is the Cox algorithm (Cox, 1990; Cox, 1991). Given a laser range scan and a map consisting of line features, this algorithm computes the relative displacement (translation and rotation) between the scan and the map with a least square approach. Correspondences between data points of a laser scan and line features of the map are established on the basis of a minimal Euclidean distance between a data point and the intersection with the line feature obtained by the perpendicular projection of the data point onto a line feature.

Due to nonlinearities in the coordinate transformations no closed form solution exists for the Cox algorithm and it has to iterate until convergence is achieved. The computational complexity of the algorithm is $O(nmk)$, where n is the number of data points of a laser scan, m is the number of line features contained in the map and k is the number of iterations required to achieve convergence.

Algorithms for solving the latter problem are commonly referred to as scan matching algorithms. The big advantage of scan matching algorithms is that they do not need a geometric map in order to determine a pose estimate. Given a laser range scan and a map consisting of several previously recorded scans, these algorithms compute the relative displacement (translation and rotation) between the scan and the map based on a least square approach. Approaches differ in the representations and optimisation criteria used.

The approach of Weiss and von Puttkamer: The CCF (cross correlation function) algorithm is based on the use of a cross correlation function (Weiss and von Puttkamer, 1995). Here, both scans are replaced by stochastic representations (histograms) and the matching is solved by finding the maximum of a cross correlation function. This algorithm works only well in polygonal environments. Furthermore, the basic algorithm as presented in (Weiss and von Puttkamer, 1995) is especially suited for an orthogonal environment. Several enhancements allow the application of this algorithm to nonorthogonal environments (e.g. see (Gutmann and Schlegel, 1996)). The computational complexity of the CCF algorithm is $O(n + m^2)$, where n is the number of data points of a laser scan, m is the number of histogram cells used.

The approach of Lu and Milios: The IDC (Iterative dual correspondence) (Lu and Milios, 1994) algorithm does a point-to-point correspondence for calculating the scan alignment. The correspondence problem is solved by two heuristics: (1) the closest point rule and (2) the matching range rule. Again, due to nonlinearities in the coordinate transformations no closed form solution exists and the IDC algorithm has to iterate until convergence is achieved. The computational complexity of the algorithm is $O(n^2k)$, where n is the number of data points of a laser scan and k is the number of iterations required to achieve convergence. An extension of this algorithm is used by (Lu and Milios, 1997) to build globally consistent maps.

Gutmann (Gutmann and Schlegel, 1996; Gutmann et al., 1998; Gutmann, 2000) investigated several combinations of, and enhancements for the above algorithms. An extremely fast and precise variant was successfully applied in the CS Freiburg RoboCup team (Gutmann, Weigel, and Nebel, 2001; Weigel et al., 2001; Weigel et al., 2002). Further variants have been implemented by the CoPS and Attempto RoboCup team from the Universities of Stuttgart and Tübingen, respectively.

Vision Sensors

The approach of Lowe: Early approaches to parameter estimation and model fitting in video images are described in (Lowe, 1987; Lowe, 1991). This method is able to handle objects with arbitrary curved surfaces and can estimate their pose as well as any number of internal parameters representing articulations, variable dimensions, or surface deformations. Lowe applies a least square estimation techniques, which is similar to the one used by the CIIL algorithm. This is augmented by a numerical stabilisation method that incorporates a prior model of the range of uncertainty in each parameter and estimates the standard deviation of each image measurement. This allows useful approximate solutions to be obtained for problems that would otherwise be under-determined or ill-conditioned. In addition, the Levenberg-Marquardt method is used to always force convergence of the solution to a local minimum.

The relatively simple search and matching procedure of (Lowe, 1991) is extended in (Lowe, 1992) such that it can handle matching and measurement errors. These errors can be treated in an integrated way by using the computation of variance in predicted feature measurements to determine the probability of correctness for each potential matching feature. In return, a best-first search procedure uses these probabilities to find consistent sets of matches, which eliminates the need to treat outliers during the analysis of measurement errors. The most reliable initial matches are used to reduce the parameter variance on further iterations, minimising the amount of search required for matching more ambiguous features. The computational complexity of Lowe's algorithm is $O(nmk)$, where n is the number of data points per curve, m is the number of curve features contained in the map, and k is the number of iterations required to achieve convergence. Not contained in this estimate is the computational demand for image preprocessing. The system is shown to be capable of tracking 3D objects at frame rates of 3 to 5 Hz.

Lanser (1997) extends Lowe's approach and applied it first for pose estimation, object detection, and object identification in the context of autonomous mobile robot systems. However, the high computational demand for image preprocessing and the applied search and matching strategies make Lowe's and Lanser's approach impractical for highly dynamic environments, such as the RoboCup scenario.

The approach of Se et al.: A very interesting and purely vision-based approach to Simultaneous Localisation And Map Building (SLAM) for autonomous robots is presented in (Se, Lowe, and Little, 2002b; Se, Lowe, and Little, 2002a; Se, Lowe, and Little, 2002c). The algorithm makes use of natural visual landmarks, called SIFT (Scale Invariant Feature Transform) features, which are invariant to image translation, scaling, rotation, and partially invariant to illumination changes and affine or 3D projections. SIFT features are determined by a *Triclops* stereo vision system and their position is stored in a database. Every feature is tracked with its own Kalman filter. By matching a set of currently observed landmarks with the landmarks contained in the database, the robot can localise itself globally. In (Se, Lowe, and Little, 2002a) two different algorithms are presented. This first one is based on a least square procedure, while the second algorithm is based on the RANSAC algorithm (see Section 5.4.2 for a more detailed description). While both algorithms perform equally well as far as accuracy of their estimates is concerned (the mean translational and rotational errors are around 6.1 cm an 1.3^o, respectively), assessments of the computational requirements reveal a little advantage for RANSAC. However, response times of more than 0.3 s for one localisation cycle prohibit its use in the RoboCup scenario.

The approach of Olson: Olson (2000) describes a probabilistic self-localisation technique for mobile robots, that is based on the principal of maximum-likelihood estimation. The basic method is to compare a map, generated at the current robot position, with a previously generated map of the environment to probabilistically maximise the agreement between the maps. This method can operate in both indoor and outdoor environments using either discrete features or an occupancy grid to represent the world map. The map may be generated using any method to detect features in the robot's surroundings, including vision, sonar, and laser range-finder. A global search of the pose space is performed that guarantees the best position in a discretised pose space to be found according to the probabilistic map agreement measure. In addition, fitting the likelihood function to a parameterised surface allows both subpixel localisation and uncertainty estimation to be performed.

Various other approaches: A final class of algorithms especially used in the RoboCup domain is based on the Hough transformation for line and circle detection. Various teams employ special variants of this algorithm in order to perform self-localisation and ball detection (Iocchi and Nardi, 1999; Jonker, Caarls, and Bokhove, 2000; Marques and Lima, 2000b). A common problem of all approaches is that the Hough transformation is usually carried out on a complete image and, thus it is computationally extremely demanding.

Omnidirectional Vision Sensors

Omnidirectional mirrors mounted in front of a camera system (see Figure 5.9(a)) enables the camera to obtain a 360^o view of the environment with the acquisition of a single image (Figure 5.9(b)). In the context of robotic applications, omnidirectional mirrors are gaining more and more attention. This is mainly due to the fact, that a robot equipped with an omnidirectional camera system can obtain information about its close neighbourhood by

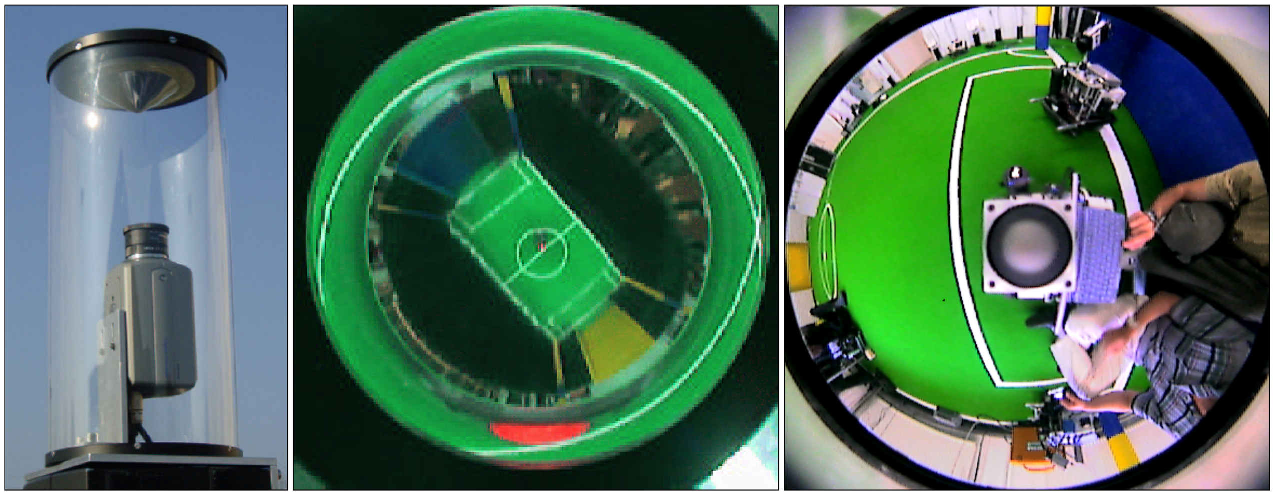


Figure 5.9: Images captured with omnidirectional camera systems. (a) The omnidirectional camera system of the CS Freiburg middle size RoboCup team and an image taken by the omnidirectional camera system of the (b) CS Freiburg and (c) Fraunhofer AIS Musashi middle size RoboCup team.

analysing a single image, i.e. the robot can look more or less in all directions simultaneously. This makes omnidirectional images particularly interesting for real time applications, such as localisation and object tracking. The major drawback of omnidirectional mirrors is their limited resolution and, consequently, limited achievable accuracy of pose estimates. Despite this drawback, a number of omnidirectional vision systems has been developed over the last years. This section briefly discusses design issues for omnidirectional mirrors and presents several approaches successfully applied within the robotic soccer domain.

Several different shapes of omnidirectional mirrors exist. Applicable shapes range from conical, spherical, ellipsoidal to hyperbolical (Nayar, 1997; Baker and Nayar, 1998b; Baker and Nayar, 1998a; Yagi, 1999). The shape is critical, since it always constitutes a trade-off between accuracy and the available field of view. Marchese and Sorrenti (2000) proposed an optimised shape according to the requirements of the RoboCup scenario. Firstly, objects in the near vicinity of the robot have to be recognised with high accuracy in order to facilitate obstacle avoidance and handling of the ball. Secondly, far-off or tall objects, e.g. other robots, have to be recognised, whereas accuracy plays not an important role. Thirdly, features providing evidence of the robot's location, e.g. field lines, have to be recognised with a constant distance error in order to simplify the self-localisation process. Based on these requirements Schulenburg, Weigel, and Kleiner (2003) calculated a mirror profile which is composed of three parts. The first part is isometric and shaped such that it removes distortion due to the mirror projection. This part allows a linear mapping from objects on the field to the camera image up to a distance of 6 meters. The second part is designed with constant curvature and does not remove the distortion of the image, but allows a reliable detection of high and far-off objects with a maximum height of 0.6 meters and a maximum distance of 10 meters. The third part, designed with curvature as well, allows an accurate detection of objects within a range of 0.2m and 0.8m.

The approach of Plagge et al.: Plagge et al. (1999) and Adorni et al. (2001) use the omnidirectional image to extract range data similar to the data obtained by a laser range finder and then apply the Cox algorithm (Cox, 1991) to estimate the robots pose. The range data is determined as follows. First, the algorithm analyses the omnidirectional image from the centre, along several lines of sight and searches for colour transitions from green to white, blue or yellow. Then, a camera model of the omnidirectional camera system is used to transform these colour transitions into range information. The algorithm is shown to be real time capable and achieves a mean position accuracy of 20 to 30 cm. The drawback of this algorithm is that it requires the RoboCup field to be surrounded by walls and makes no use of further visible features such as lines and the corner flagposts.

The approach of Schulenburg et al.: Another similar but more applicable approach, which also makes use of all straight field lines, is presented in (Schulenburg, Weigel, and Kleiner, 2003). Again, the omnidirectional image is analysed from the centre in 2^o increments and a set of colour transitions, representing field lines, is extracted. Since the projection function of the mirror is known, the world coordinates of these possible line points can be determined. Finally, the extracted line points are grouped together into lines by standard divide and conquer algorithms and a set of pose hypothesis is generated by a line matching algorithm (Gutmann, Weigel, and Nebel, 2001). A nice variant of this algorithm is that it can also be used with laser range data. The pose estimates, generated by both sensors, are fused by an extended Kalman filter. Purely, omnidirectional vision based self-localisation achieves an accuracy of 20 to 30 cm and three to four degrees, for position and orientation respectively. Laser based self-localisation achieves positional accuracy of 12 to 15 cm and an orientational accuracy of approximately three degrees. Poses estimated generated with both types of information available, achieve an approximate accuracy of 10 cm and 2.5^o. A further nice property of this algorithm is its capability to solve the global localisation problem within the RoboCup scenario.

Further omnidirectional camera systems for the robot soccer domain have been built by (Lima et al., 2001; Marques and Lima, 2000b; Marques and Lima, 2000a) and (Nakamura et al., 2000).

5.4.2 Robust Estimation Techniques

Robust estimation techniques are robust to outliers, false correspondences, false observations, and false positives. In general they produce less accurate estimates and require greater computational resources than the outlier sensitive estimation techniques presented in the previous Section.

The four most commonly used techniques for self-localisation are Multiple Hypothesis Localisation (MHL), Markov Localisation (ML), Monte Carlo Localisation (MCL) and RANdom SAmple Consensus (RANSAC). Further robust estimation techniques are described in (Kumar and Hanson, 1994).

Multiple Hypothesis Localisation

The approach of Jensfelt and Kristensen: Jensfelt and Kristensen (2001) present a probabilistic approach for mobile robot localisation using an incomplete topological world model. The method is called multiple hypothesis localisation (MHL) (Jensfelt, 2001; Jensfelt and Kristensen, 2001), and uses multiple hypothesis Kalman filter (MHKF) based pose tracking, combined with a probabilistic formulation of hypothesis correctness, to generate and track Gaussian pose hypotheses online (see also Section 3.4.6). Apart from a lower computational complexity, this approach has the advantage, over traditional grid based methods, that incomplete and topological world model information can be utilised.

Within the MHL framework new Hypotheses are usually generated with the observation of unique landmarks. Jensfelt and Kristensen (2001) distinguish between creative and supportive features. A creative feature carries enough pose evidence to initiate a new hypothesis (e.g. a door), whereas the supportive one can only support already existing hypotheses (e.g. a corner in a corridor or wall). The covariance of a new hypothesis is estimated from the landmark positions and the observation uncertainties. Data association between an existing hypothesis and an observation is performed on the basis of the Mahalanobis distance. Before an hypothesis is updated, track splitting is performed. This ensures that no tracks are lost due to spurious and false observations. Hypothesis pruning is based on two heuristics: Firstly, hypotheses whose probability mass is below a certain threshold are deleted. Secondly, if two hypotheses are almost similar in a Mahalanobis distance sense, the less likely hypothesis is deleted.

MHL is intensively tested and evaluated in (Kristensen and Jensfelt, 2003). It is shown to be as accurate as MCL and similar methods, but much faster in solving the global localisation and kidnapped robot problems.

Markov Localisation

Markov Localisation (ML) is a commonly used global localisation procedure which is based on the discrete Bayes filter (DBF) (see Section 3.4.7). The idea of ML is to maintain a discrete position probability density over the whole threedimensional state space of the robot in its environment. This density is updated whenever the robot moves or receives new information from its sensors.

The different variants of this technique can roughly be distinguished by the type of discretisation used for the representation of the state space. In (Nourbakhsh, Powers, and Birchfield, 1995; Simmons and Koenig, 1995; Kaelbling, Cassandra, and Kurien, 1996; Thrun, 1998) Markov localisation is used for landmark-based corridor navigation and the state space is organised according to the topological structure of the environment. Based on an orthogonality assumption (Nourbakhsh, Powers, and Birchfield, 1995; Simmons and Koenig, 1995; Kaelbling, Cassandra, and Kurien, 1996) consider only four possible headings of the robot. In (Burgard et al., 1996) a finegrained gridbased discretisation of the state space is proposed. The advantage of this approach is that it provides accurate position estimates and that it can be applied in arbitrarily unstructured and even densely populated environments (Burgard et al., 1998; Fox et al., 1998). The disadvantage of this approach, however, is the huge state space which has to be maintained. The Dynamic Markov Localisation (DML) approach presented in (Burgard et al., 1998) overcomes this problem because it dynamically adopts the size of the state space according to the robot's certainty in its position. It is able to globally localise the robot whenever necessary, and to efficiently keep track of the robot's position in normal situations in which the robot has almost certain knowledge about its own location.

The approach of Gutmann: Gutmann (2002) presented a further approach called Markov-Kalman localisation (MKL) which is a combination of Markov localisation and Kalman filter. MKL is well suited for robots observing known landmarks, having a rough estimate of their movements, and which might be displaced to arbitrary positions at any time. Experimental results show that this method outperforms both of its underlying techniques by inheriting the accuracy of the Kalman filter and the robustness and relocalisation speed of Markov localisation.

Monte Carlo Localisation

Monte Carlo Localisation (MCL) is based on the Monte Carlo Bayes filter (MCBF) presented in Section 3.4.8. MCL maintains a set of particles, which represent a discrete position probability density over the whole threedimensional state space of the robot in its environment. Just as for ML, this density is updated through resampling, whenever the robot moves or receives new information from its sensors.

Several variants of MCL have already been discussed in Section 3.4.8. In the following, a special real time particle filter and an extension of MCL for multi-robot systems is presented.

The approach of Fox et al.: Fox (2003) presents a statistical approach to increase the efficiency of particle filters by adapting the size of sample sets during the estimation process. The key idea of the KLD-sampling method is to bound the approximation error introduced by the sample-based representation of the particle filter. The name KLD-sampling comes from the fact that the approximation error is measured by the Kullback-Leibler distance. The adaptation approach chooses a small number of samples if the density is focused on a small part of the state space, and it chooses a large number of samples if the state uncertainty is high. Both the implementation and computation overhead of this approach are small. Extensive experiments using mobile robot localisation as a test application show that this approach yields drastic improvements over particle filters, with fixed sample set sizes, and over a previously introduced adaptation technique.

The approach of Fox and Burgard: A MCL algorithm for cooperative mobile robot localisation in an office environment is proposed in Fox et al. (2000). When teams of robots localise themselves in the same environment, probabilistic methods are employed to synchronise each robot's belief whenever one robot detects another one. As a result, the robots localise themselves faster, maintain higher accuracy, and high-cost sensors are amortised across multiple robot platforms. The technique has been implemented and tested using two mobile robots equipped with cameras and laser range-finders for detecting other robots. The results, obtained with the real robots and in series of simulation runs, illustrate drastic improvements in localisation speed and accuracy when compared with conventional single-robot localisation. A further experiment demonstrates that under certain conditions, successful localisation is only possible if teams of heterogeneous robots collaborate during localisation.

RANdom SAmple Consensus

Many computer vision algorithms include a robust estimation step where model parameters are computed from a data set containing a significant proportion of outliers. The RANdom SAmple Consensus (RANSAC) algorithm introduced by Fischler and Bolles (1981) is possibly the most widely used robust estimator in the field of computer vision.

RANSAC has been applied in the context of short baseline stereo (Torr, Zisserman, and Maybank, 1995; Torr, 1995; Hartley and Zissermann, 2000), wide baseline stereo matching (Pritchett and Zissermann, 1998; Schaffalitzky and Zisserman, 2001; Tuytelaars and Van Gool, 2000), motion segmentation (Torr, 1995), mosaicing (McLauchlan and Jaenicke, 2000), detecting geometric primitives (Clarke, Carlsson, and Zisserman, 1996), robust eigenimage matching (Leonardis and Bischof, 2000) and elsewhere.

The structure of the RANSAC algorithm is simple but powerful. Repeatedly, subsets are randomly selected from the input data (e.g. a subset of model image correspondences) and the model parameters fitting the sample set are computed. The size of the random samples is the smallest sufficient for determining model parameters. In a second step, the quality of the model parameters is evaluated on the full data set. Different cost functions may be used (Torr and Zisserman, 2000) for the evaluation, the standard being the number of inliers, i.e. the number of data points consistent with the model. The process is terminated when the likelihood of finding a better model becomes low. The strength of the method stems from the fact that, to find a good solution, it is sufficient to select a single random sample not contaminated by outliers. Depending on the complexity of the model (the size of random samples) RANSAC can handle contamination levels well above 50%, which is commonly assumed to be a practical limit in robust statistics (Rousseeuw and Leroy, 1987).

The speed of RANSAC depends on two factors. First, the level of contamination determines the number of random samples that have to be used to guarantee a certain confidence in the optimality of the solution. Second, the time spent evaluating the quality of each of the hypothesised model parameters is proportional to the size N of the data set. It is obvious, that the application of the RANSAC algorithm for state estimation problems in dynamic scenarios poses a huge demand for computational resources.

5.5 Conclusions

This chapter presented the Cooperative Incremental Iterative Localisation (CIIL) algorithm and applied it to solve the joint estimation problem of self- and ball-localisation in the RoboCup scenario. The algorithm analyses digital images with the help of a known model of the environment and runs on off-the-shelf computing hardware.

CIIL allows a team of robots to track its poses and the ball position with high precision at frame rate. The high speed of the algorithm is achieved through a model driven feature extraction process that requires the analysis of only very few pixels per image. In contrast to this, data driven approaches such as blob analysis, edge detection and Hough transformations usually require the analysis of large areas if not the whole image.

The accuracy of the algorithm is achieved by iterating over the same image several times without loss of performance. This is enabled by the efficient feature extraction process. With every iteration, the nonlinear observation model is relinearised and the estimate is refined.

The resulting estimate is then used to initialise the next iteration. This procedure is repeated a fixed number of times or until convergence is achieved.

A further advantage of the CIIL algorithm is its capability to exploit observations performed by other robots. These observations are added to the environment model as dynamic objects and are handled by the algorithm as if they were stationary. The only preconditions these objects have to fulfil is that their geometric properties are known in advance, and that they can uniquely be identified. By sharing and exploiting observations performed by other teammates, a team of robots performs cooperative localisation and is able to solve localisation problems a single robot would not be able to solve on its own.

The CIIL algorithm and its properties are extensively evaluated in a series of real world experiments in Chapter 7.

Chapter 6

Cooperative Object Detection and Tracking

6.1 Introduction

The second essential perception problem a robot soccer player has to solve is the estimation of the positions and the moving directions of all other non-stationary dynamic objects on the soccer field. The objects that are of particular interest are the opponent robots and humans[1] entering the soccer field. Furthermore, it is often necessary to detect and track robots from the own team and the ball. The estimates obtained by this detection and tracking process are a necessary input for the action selection and path planning routines and allow them to plan collision free paths and perform obstacle avoidance. Again, this task has to be accomplished in real time and only based on information obtained by vision sensors.

Detection and tracking of opponent robots considers the following state estimation problem. The world is populated with a set of stationary and moving objects. The number of objects may vary and they might be occluded, or out of sensor range. Robots are equipped with sensing routines that are capable of detecting objects within sensor range, estimating the positions and moving directions of the detected objects. Due to a general but simple object model used for opponent detection, noise contained in the video stream, and varying lighting conditions these routines are error-prone. Sometimes they hallucinate objects and at other times they overlook objects. In order to deal correctly with false positive observations, false negative observations, and inaccurate observations a MHKF framework is applied. It demonstrates how a MHKF framework can be used to model dynamic environments in multi-robot systems and equip it with mechanisms to handle multiple mobile sensors with uncertain positions.

This chapter presents a fast (real time capable) and accurate vision-based Cooperative Object Detection and Tracking (CODT) procedure. CODT allows a robot to observe dynamic obstacles (robots and humans) and to track them at frame rate with high precision. Based on a simple model of the opponents, the observations are performed by a robot. With every set of observations the probability densities describing the presence of obstacle is then updated by the CODT. A clear advantage of the CODT algorithm is that it is also capable of taking

[1]During a match the referee and robot maintenance staff is allowed to enter the soccer field.

observations into account that were performed by several teammates. By doing this, a team of robots performs cooperative opponent detection and tracking and is able to gain a more precise and complete view of the environment than a single robot is able to gain with its own camera.

In the remainder of this chapter the CODT algorithm is presented and evaluated. Section 6.2 presents the belief state, object hypotheses, and object tracks used by the CODT algorithm. The CODT algorithm and some implementation details are described in Section 6.3. Related work is briefly described in Section 6.4. A summary and discussion is given in Section 6.5 and this also concludes the chapter. The algorithm and its properties are extensively evaluated in a series of real world experiments in Chapter 7.

6.2 Belief State, Object Hypotheses and Tracks

Before the Cooperative Opponent Detection and Tracking algorithm is described in detail, this section will briefly review its central data structures, used to represent the belief state, the object hypotheses, and the object tracks.

When tracking the positions of a set of opponent robots, there are two kinds of uncertainties the state estimator has to deal with. The first one is caused by the inaccuracy of the robots' sensors. The second kind of uncertainty is introduced by the data association problem (Bar-Shalom and Fortmann, 1988), i.e. the assignment of observed objects to existing object hypotheses from the previous time step. An appropriate data structure being capable of representing both kinds of uncertainty will be presented in the next two sections.

6.2.1 Object Hypotheses

The central data structure used by the CODT algorithm is the object hypothesis, h_t^i. Indices i and t specify the hypothesis number and creation time, respectively. Each hypothesis represents an estimate of a possible position and velocity of an opponent robot or another dynamic object on the playing field. For every time step the algorithm maintains a set of all existing hypotheses, $H_t = \{h_t^1, \ldots, h_t^n\}$. It should be noted, that with every time step the number of hypotheses contained in this set may vary.

The belief, $Bel(\widehat{x}_t)$, of the CODT algorithm is approximated by a set of weighted multivariate Gaussian random variables, $h_t^i \sim N(\bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i})$. The mean vector, $\bar{\mathbf{h}}_t^i$, represents the joined positions and velocity estimate of an opponent robot hypothesis, while the covariance, $\Sigma_{\mathbf{h}_t^i}$, represents the associated uncertainty of this estimate. The validity of a hypothesis is represented through its weight, p_t^i. Given these definitions, the belief of the CODT algorithm can be represented as follows:

$$Bel(\widehat{x}_t) = \sum_{i=1}^{n} p_t^i * N(\widehat{x}_t; \bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i}) \tag{6.1}$$

The mean vector, $\bar{\mathbf{h}}_t^i$, of an object hypothesis, h_t^i, consists of a 2D position and velocity estimate with an associated 4×4 covariance matrix:

$$\bar{\mathbf{h}}_t^i = (x, \dot{x}, y, \dot{y})^T \tag{6.2}$$

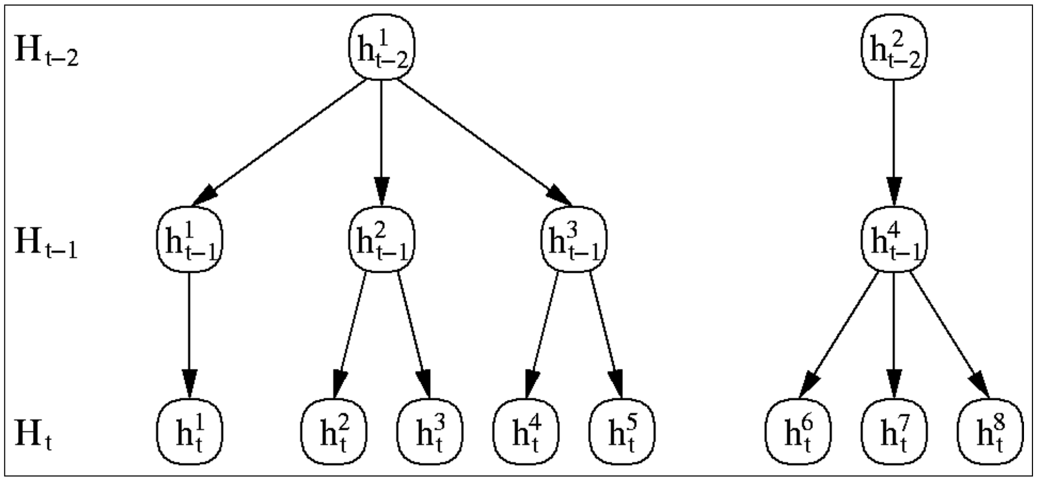


Figure 6.1: Hypotheses trees maintained by the CODT algorithm. Every branch of a tree represents one possible track of a physical object. New branches are introduced as an attempt to solve data association ambiguities through track splitting.

6.2.2 Object Tracks

With the arrival of new sensor data and opponent observations the algorithm has to solve the data association problem, i.e. decide which existing hypothesis is reconfirmed by which observation. Usually, this problem is full of ambiguities and cannot be solved uniquely. For example, it is possible that one opponent observation reconfirms several hypotheses or that an existing hypothesis is reconfirmed by more than one observation.

A way to overcome the data association problem is to avoid unique associations and consider all possible assignments instead. This procedure is called track splitting (see Figure 6.1) and generates a new hypothesis for every possible combination of a hypothesis from the previous time step and an observation from the current time step. Every newly created hypothesis is added to the set of current hypotheses and is linked with its predecessor contained in the previous set of hypotheses. Over time this creates a tree like structure, where each branch represents a possible track of a dynamic object.

More formally, the CODT algorithm maintains the last N sets of hypotheses H_{t-N} to H_t at any time. Hypotheses that are assumed to originate from the same physical object are linked in a tree like structure from one set of hypotheses to the next. A node h^j_t is the son of the node h^i_{t-1} if h^j_t results from the assignment of an observation with a predicted state $\widetilde{h}^i_{t-1}$ of the hypothesis h^i_{t-1}.

6.3 The Cooperative Opponent Detection & Tracking Algorithm

In this section the CODT algorithm will be described by first detailing the underlying computational structure and then presenting the computational steps of the algorithm.

The computational structure of the CODT algorithm is outlined in Figure 6.2. In the

```
algorithm Cooperative Opponent Detection and Tracking ($H, data, t_{data}$)
 1  let
 2     $H$               % set of hypotheses representing previous belief state
 3     $H'$              % set of hypotheses representing updated belief state
 4     $data$            % data item
 5     $Z$               % set of observed opponents
 6     $t_{last}, t_{data}$   % time stamps of last and current data item
 7     $\Delta t$        % time difference between two data items
 8
 9  do
10     $\Delta t = t_{data} - t_{last}$;
11
12     switch ($data$)
13
14        case ($data$ is an $image$) :
15           $Z \leftarrow$ Interpret Sensor Data ($image$);
16           % Broadcast observations to teammates
17           Broadcast Observations ($Z, t_{data}$);
18           % prediction stage
19           $H' \leftarrow$ Multiple Hypothesis Kalman Filter ($H, \Delta t$));
20           % update or measurement stage
21           $H' \leftarrow$ Multiple Hypothesis Kalman Filter ($H', Z$));
22
23        case ($data$ is a set of opponent observations from a teammate, $Z$) :
24           % prediction stage
25           $H' \leftarrow$ Multiple Hypothesis Kalman Filter ($H, \Delta t$));
26           % update or measurement stage
27           $H' \leftarrow$ Multiple Hypothesis Kalman Filter ($H', Z$));
28
29     $t_{last} = t_{data}$;
30
31  return ($H'$);
```

Figure 6.2: The Cooperative Opponent Detection and Tracking (CODT) algorithm.

algorithm H denotes the set of object hypothesis, which is updated every time new sensor data arrives. Possible types of sensor data are (1) image data acquired by the camera mounted on the robot (see Figure 6.2 lines 14 to 21) and (2) a set of opponent observations, Z, observed and communicated by a teammate (lines 23 to 27). In the first case, the image data is processed and a set of opponent positions, Z, is observed (line 15). Before the data is integrated into the current set of hypotheses, it is broadcast to all teammates (line 7). In the second case, opponent observations performed by teammates can be integrated without further processing.

It is noteworthy, that communication delays are regarded to be negligible and that teammate observations are processed in a first come first serve order. The remainder of the algorithm is identical for both types of data. The multiple hypothesis Kalman filter algorithm (see Section 3.4.6 or 6.3.3 for more details) is called twice. First, in order to predict the current set of hypotheses (lines 19 and 25) and then to update it with the current set of observations (lines 21 and 27).

In the following, the computational steps of the CODT algorithm will be presented: (1) detecting feature blobs in the captured image that correspond to an opponent, (2) estimating the position and uncertainties of the opponent in world coordinates, and (3) associating them with the correct object hypothesis.

6.3.1 Image Preprocessing, Feature Extraction and Object Detection

This section outlines the feature extraction process (see Algorithm 6.3) which is performed in order to extract a set of feature blobs out of an image, that correspond to opponent robots.

It is assumed that the opponent robots are coloured black and have approximately circular shape. The discrimination of teammates and opponent robots is enabled through predefined colour markers (cyan and magenta, see Figure 5.4) on the robots. Each marker colour may be assigned to any of the two competing teams. Consequently, it is important that the following algorithms can be parameterised accordingly. Furthermore, it is assumed that the tracked object almost touches the ground and an inverse camera model can be applied for distance estimation (see Figure 6.4). These predefined robot colours allow for a relatively simple feature extraction process.

The following procedure extracts a set of regions from a captured image, where each region corresponds to a currently visible robot. After capturing an image (see Figure 6.3 line 11) the black colour-regions are extracted using colour classification and morphological operators. In order to be recognised as an opponent robot a black blob has to satisfy several constraints (line 12), e.g. a minimum/maximum size and a red or green colour-region adjacent to the bottom region row. These constraints enable the routine to distinguish robots from black logos and adverts affixed on the wall surrounding the field. Furthermore, blobs that contain or have a colour-region of the own team colour in the immediate neighbourhood are discarded.

In the next step the physical size of the object corresponding to a blob is examined. For every extracted region the object's physical diameter is estimated. If it exceeds an upper threshold, two robots are assumed to be directly next to each other. In this case the blob is divided into two.

In order to detect cascaded robots (line 13), i.e. opponent robots that are partially occluded by other robots, the algorithm uses discontinuities in row width. As soon as the length of a row differs significantly from the length of its lower predecessor and the respective world coordinates are more than 10 cm above the ground it is assumed that a partly occluded object is detected. However, before the region can safely be split into two, the resulting subregions have to obey several further constraints such as a minimum width and height.

Finally, for every extracted region three features are computed (line 16): The bottom most pixel *row*, the column *col* representing the centre of gravity, and a mean blob *width*. For the latter two features only the three bottom most rows, which exceed a certain length, are used.

```
algorithm INTERPRET SENSOR DATA ($< \hat{\mathbf{x}}, \Sigma_{\hat{\mathbf{x}}} >, image$)
let
    $< \hat{\mathbf{x}}, \Sigma_{\hat{\mathbf{x}}} >$    % belief state <mean,covariance> of the observing robot's pose
    $< \bar{\mathbf{y}}, \Sigma_{\mathbf{y}} >$    % intermediate <mean,covariance>
    $image$         % image data
    $R = \{r_1, \ldots, r_{n_r}\}$   % set of regions
    $Z$             % set of observed opponents (features)
    $i$             % loop counter

do
    $Z \leftarrow \emptyset$;
    $R \leftarrow$ EXTRACT BLACK REGIONS (GET SENSOR DATA ());
    $R \leftarrow$ CHECK CONSTRAINTS ($R$);
    $R \leftarrow$ EXTRACT CASCADED ROBOTS ($R$);

    for $i \leftarrow 1$ to $|R|$ do
        $(row, col, width) \leftarrow$ EXTRACT FEATURES ($r_i$);
        $\bar{\mathbf{y}} \leftarrow [\hat{\mathbf{x}}, row, col, width]^T$ ;
        $\Sigma_{\mathbf{y}} \leftarrow \begin{pmatrix} \Sigma_{\hat{\mathbf{x}}} & 0 & 0 & 0 \\ 0 & \sigma_{row} & 0 & 0 \\ 0 & 0 & \sigma_{col} & 0 \\ 0 & 0 & 0 & \sigma_{width} \end{pmatrix}$;
        $Z \leftarrow Z \cup$ UNSCENTED TRANSFORMATION($\bar{\mathbf{y}}, \Sigma_{\mathbf{y}}, opp$);

return ($Z$);
```

Figure 6.3: The Algorithm used for opponent detection and uncertainty estimation.

Occlusions through the ball are also considered while these rows are determined.

Empirically it was found that this feature extraction procedure is sufficient to determine accurate positions of opponent robots. Mistakenly extracted objects are generally resolved in a fast manner by the further computational steps of the CODT algorithm.

6.3.2 Estimation of Opponent Position and Uncertainty

This section discusses how the position and the respective covariance of an observed robot is estimated. Each opponent observation is modelled in world coordinates by a bi-variate Gaussian random variable, $z \sim N(\bar{\mathbf{z}}, \Sigma_{\mathbf{z}})$, with mean, $\bar{\mathbf{z}}$, and a 2×2 covariance matrix, $\Sigma_{\mathbf{z}}$. In order to achieve an accurate estimate for z this procedure takes the estimated pose and the covariance of the observing robot into account as well as the position of the detected feature blob in the image and the associated measurement uncertainties.

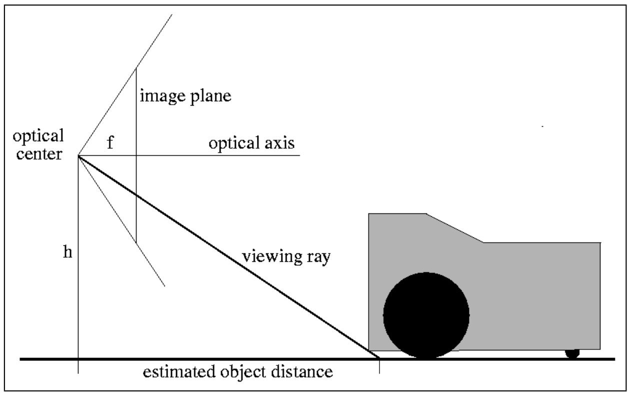


Figure 6.4: An estimate of the robot's distance is given through the intersection of the viewing ray with the ground plane of the field.

The observation model *opp* is defined on the basis of an inverse camera model (see Section 2.4.6). This model determines the world coordinates of an opponent robot based on the pose estimate, $x \sim N(\bar{\mathbf{x}}, \Sigma_{\mathbf{x}})$, of the observing robot, the pixel coordinates *row*, *col* of the centre of gravity and the *width* of the opponent robot's blob. Due to rotations and radial distortions of the lenses *opp* is nonlinear. First, the function *opp* converts the blob's pixel coordinates to relative polar coordinates, applying equations Eq. (2.26) to (2.30). On this and on the width of the observed blob the radius of the observed robot is estimated. Since the polar coordinates only describe the distance to the opponent robot but not the distance to its centre, the radius is added to the distance. Finally, the polar coordinates are transformed into world coordinates taking the observing robot's pose estimate $\widehat{x}$ into account.

In order to estimate the position $\bar{\mathbf{z}}$ and the covariance $\Sigma_{\mathbf{z}}$ of an opponent robot, the unscented transformation of Julier and Uhlmann (1997) is used (see also Section 2.3.4). This transformation allows the efficient propagation of uncertainties without creating the necessity to derive the partial derivatives of the propagation functions. Julier and Uhlmann also proved that the unscented transformation provides more realistic uncertainty estimates than an approximation of the observation model obtained through linearisation with a truncated first order Taylor series.

For the application of the unscented transformation, an augmented mean $\bar{\mathbf{y}}$ (see Figure 6.3 line 17) and covariance $\Sigma_{\mathbf{y}}$ (line 18) describing jointly the observing robot's pose estimate and the observed robot features is required. $\bar{\mathbf{x}}$, *row*, *col* and *width* are assumed to be uncorrelated with variances σ_{row}, σ_{col} and σ_{width}. These sigmas are dependent on the image processing hardware and can be determined from a series of experiments.

The unscented transformation (line 19) is then applied to the augmented mean and covariance using the nonlinear mapping *opp*. This yields the opponent robot's position and uncertainty as Gaussian random variable $z \sim N(\bar{\mathbf{z}}, \Sigma_{\mathbf{z}})$, which are stored in the set of observations Z_t.

In Figure 6.5, the uncertainties of objects depending on the uncertainty of the observing robot and their relative distances are displayed using σ-contours. For illustrative purposes the uncertainty ellipses are scaled by an order of five. Each robot observes two obstacles in 3.5 and 7 m distance. Robot Odilo is very certain about its pose estimate and thus the covariance

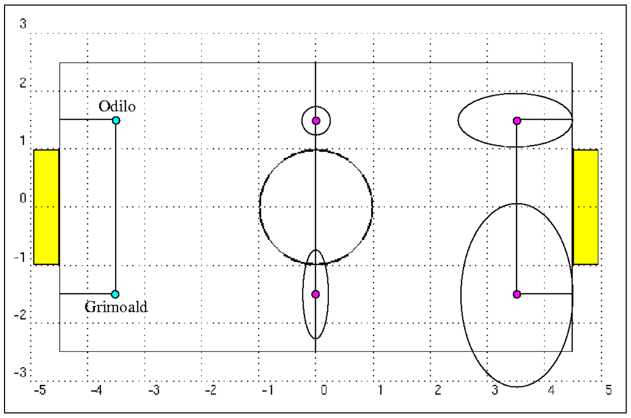


Figure 6.5: Propagation of uncertainties. For illustrative purposes the uncertainty ellipses are scaled by an order of five.

of the observed robot depends mainly on its distance. The estimated distances according to the unscented transformation do not diverge from the real distances. Robot Grimoald has a high uncertainty in its orientation, approximately 7^o. Consequently, the position estimate of the observed obstacle is less precise and highly influenced by the orientation uncertainty.

In the subsequent step, the extracted opponent positions z_t^i are associated with the predicted hypotheses $\widetilde{h}_t^j$ by the multiple hypothesis tracking algorithm.

6.3.3 Multiple Hypothesis Tracking

For associating new observations with existing object hypotheses, a variant of the multiple hypothesis Kalman filter (MHKF) is used. The first version of a MHKF was developed by Reid (1979) in the context of a tracking application. Hence, the name that was chosen for this algorithm was multiple hypothesis tracking (MHT) algorithm. The objectives of the MHT algorithm are to keep a set of object hypotheses, each describing a unique real object and its position, to maintain the set of hypotheses over time, and to estimate the likelihood of the individual hypotheses.

Before the details of the MHT algorithm are presented the following brief description will give an intuition of how it works. The MHT algorithm maintains a forest of hypotheses trees, that is a set of trees (see Section 6.2.2). The nodes in the forest are object hypotheses and represent the association of an observed object with an existing object hypothesis. Each hypothesis has an association probability, which indicates the likelihood that observed object and object hypothesis refer to the same object. In order to determine this probability, the motion model is applied to object hypothesis of the previous iteration to predict the object's position. Then the association probability is computed by weighting the distance between the predicted and the observed object position. Thus, in every iteration of the algorithm each observation is associated with each existing object hypothesis.

The computational structure of the algorithm is shown in Figures 6.6 and 3.7. An iteration begins with the set of hypotheses of object states $H_t = \{h_t^1, \dots, h_t^{n_t}\}$ from the previous iteration t. Each h_t^i is a Gaussian random variable, $h_t^i \sim N(\bar{\mathbf{h}}_t^i, \Sigma_{\mathbf{h}_t^i})$, ranging over the state

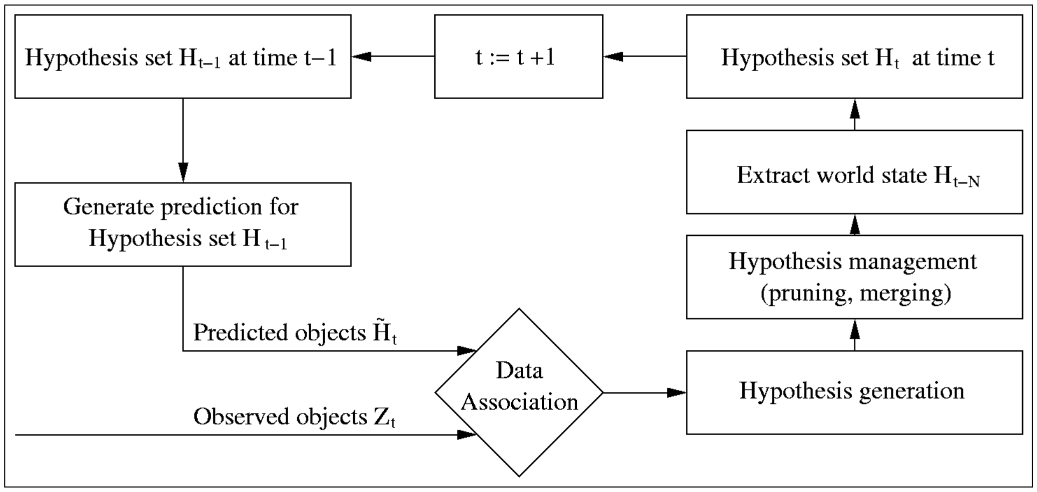

Hypothesis set H_{t-1} at time t−1
t := t +1
Hypothesis set H_t at time t
Extract world state H_{t-N}
Generate prediction for Hypothesis set H_{t-1}
Hypothesis management (pruning, merging)
Predicted objects $\tilde{H}_t$
Data Association
Hypothesis generation
Observed objects Z_t

Figure 6.6: The multiple hypotheses framework for dynamic environment modelling.

space of a single object and represents a different assignment of measurements to objects, which was performed in the past. The algorithm maintains a Kalman filter for each hypothesis.

With the arrival of new sensor data (see Figure 3.7 line 12), $Z_{t+1} = \{z^1_{t+1}, \ldots, z^{n_{t+1}}_{t+1}\}$, the motion model (line 18) is applied to each hypothesis and intermediate hypotheses $\tilde{h}^i_{t+1}$ are predicted. For this step, a constant velocity model is used. Assignments of measurements to objects (line 27) are established on the basis of a statistical distance measure, such as the Mahalanobis distance. Each subsequent child hypothesis represents one possible interpretation of the set of observed objects and, together with its parent hypothesis, represents one possible interpretation of all past observations. With every iteration of the MHT, probabilities (line 29) describing the validity of hypotheses are calculated. Furthermore, for every observed object a new hypothesis with associated probability is created (lines 33 to 36) (Cox and Leonard, 1994).

In order to constrain the growth of the hypotheses trees, the algorithm prunes improbable branches (line 40). Pruning is based on a combination of ratio pruning, i.e. a simple lower limit on the ratio of the probabilities of the current and best hypotheses, and the N-scan-back algorithm (Reid, 1979). This algorithm assumes that any ambiguity at time t is resolved by time $t + N$. Consequently, if at time t hypothesis h^i_{t-N} has m children, the sum of the probabilities of the leaf nodes of each branch is calculated. The branch with the greatest probability is retained and the others are discarded. After pruning, the world state of H_{t-N} can be extracted. Please note that this world state is always N steps delayed behind the latest observations. However, Section 6.3.6 will demonstrate that this delay can greatly be overcome by N observers performing parallel observations.

In the next sections, the applied motion model is presented along with the equations used to compute the probability of a hypothesis and some design decisions taken in order to apply MHT to the RoboCup scenario.

6.3.4 Motion Model

The motion model, used for predicting the movements of the hypotheses, is a simple constant velocity model. The only input used by this model is the elapsed time, Δt, since the last update has been performed. Given the definition of a state, Eq. (6.2), and the equations for state prediction, Eqs. (3.19) and (3.20), the prediction for the mean is given by:

$$\widetilde{\mathbf{h}} \quad = \quad F\bar{\mathbf{h}} + G\Delta t = I\bar{\mathbf{h}} + \begin{pmatrix} \dot{x} \\ 0 \\ \dot{y} \\ 0 \end{pmatrix} \Delta t \tag{6.3}$$

$$\tag{6.4}$$

The prediction for the covariance is determined as follows:

$$\Sigma_{\widetilde{\mathbf{h}}} \quad = \quad F\Sigma_{\mathbf{h}}F^T + \Sigma_{\mathbf{v_t}} = I\Sigma_{\mathbf{h}}I + \begin{pmatrix} \Delta t^3/3 & \Delta t^2/2 & 0 & 0 \\ \Delta t^2/2 & \Delta t & 0 & 0 \\ 0 & 0 & \Delta t^3/3 & \Delta t^2/2 \\ 0 & 0 & \Delta t^2/2 & \Delta t \end{pmatrix} q \tag{6.5}$$

Here q is the variance of a continuous white-noise process (Bar-Shalom and Fortmann, 1988). Further motion models are conceivable but in the RoboCup scenario the above model was found to be sufficient.

6.3.5 Computing the Likelihood of Association Hypotheses

Obviously, the heart of the MHT algorithm is the computation of the likelihood of association hypotheses, $P(\Theta_{t+1}^i|Z_{0\ldots t})$. Each association hypothesis, Θ, represents a different set of assignments of observations to objects, i.e. it is a collection of disjoint tracks.

This section derives the formula that is used to compute the likelihood for the specific application of multiple object tracking. The derivation of this formula is critical because it defines the probabilities that must be specified by programmers to apply the algorithm to specific applications.

Let $Z_{0\ldots t}$ be the sequence of all measurements up to time t. A new association hypothesis at time t is made up of the current set of assignments (also called an event), θ_t, and a association hypothesis, Θ_{t-1}^j, which is based on observed features up to time $t-1$ inclusively.

The probability of an association hypothesis $P(\Theta_t^i|Z_{0\ldots t})$ is transformed by using Bayes' rule and the Markov assumption in order to obtain an easier expression:

$$\begin{aligned} P(\Theta_t^i|Z_{0\ldots t}) \quad &= \quad P(\theta_t, \Theta_{t-1}^i|Z_t, Z_{0\ldots t-1}) & (6.6) \\ &= \quad \nu\, p(Z_t|\theta_t, \Theta_{t-1}^j, Z_{0\ldots t-1})\; P(\theta_t|\Theta_{t-1}^j, Z_{0\ldots t-1})\; P(\Theta_{t-1}^j|Z_{0\ldots t-1}) & (6.7) \end{aligned}$$

Here, ν is a normalisation factor ensuring that $P(\Theta_t^i | Z_{0...t})$ sums up to one over all Θ_t^i. The last term of this equation is the probability of the parent association hypothesis that was computed in the previous iteration. The second factor can be evaluated as follows (Bar-Shalom and Fortmann, 1988):

$$P(\theta_t | \Theta_{t-1}^j, Z_{0...t-1}) = \frac{\phi! \nu!}{m_k!} \mu_F(\phi) \mu_N(\nu) \prod_k (P_D^k)^{\delta_k} (1 - P_D^k)^{1-\delta_k} (P_T^k)^{\tau_k} (1 - P_T^k)^{1-\tau_k} \tag{6.8}$$

where $\mu_F(\phi)$ and $\mu_N(\nu)$ are prior probability mass functions of the number of spurious measurements, ϕ, and new geometric features, ν. Generally, $\mu_F(\phi)$ and $\mu_N(\nu)$ are assumed to be Poisson distributed with mean λ_F and λ_N. P_D^k and P_T^k are the probabilities of detection and termination of track k. δ_k and τ_k are indicator variables. δ_k (τ_k) is 1, if track k is detected (deleted) at time t and 0, otherwise.

$$\delta_k = \begin{cases} 1 & \text{, track } k \text{ (in } \Theta_{t-1}^i\text{) is detected at time } t \\ 0 & \text{, otherwise.} \end{cases} \tag{6.9}$$

$$\tau_k = \begin{cases} 1 & \text{, track } k \text{ (in } \Theta_{t-1}^i\text{) is deleted at time } t \\ 0 & \text{, otherwise.} \end{cases} \tag{6.10}$$

The indicator variable δ_k depends on the observing robot's camera orientation. It is 1, if the track k is within the sensor's field of perception and track k is not occluded by another teammate. $P_T^{\tau_k}$ is used to model the probability of an unobserved object, which decreases over time.

$$P_T^{\tau_k} = 1 - e^{-\frac{\Delta t}{\lambda_T}} \tag{6.11}$$

Δt is the number of consecutive time steps a hypothesis was not observed. λ_T determines the speed of the declination process. Larger λ_T result in a slower declination of the hypothesis probability.

The first term on the right hand side of equation 6.7 denotes the association probability of a measurement and an object state. This probability is defined with the help of a statistical distance measure, such as the Mahalanobis distance (see Eq. (2.9)):

$$N_j(\bar{\mathbf{z}}_t^i) = N_j(z_t^i; \widetilde{h}_t^j, \Sigma_{\mathbf{s}_t^j}) = \frac{1}{\sqrt{(2\pi)^{n_{\widetilde{h}}} det(\Sigma_{\mathbf{s}_t^j})}} e^{-\frac{1}{2}\{(z_t^i - \widetilde{h}_t^j)^T (\Sigma_{\mathbf{s}_t^j})^{-1} (z_t - \widetilde{h}_t^j))\}} \tag{6.12}$$

Here, $\widetilde{h}_t^i$ denotes the predicted measurement for hypothesis j and $\Sigma_{\mathbf{s}_t^j}$ is the associated innovation covariance and is defined by $\Sigma_{\mathbf{s}_t^j} = \Sigma_{\widetilde{\mathbf{h}}_t^j} + \Sigma_{\mathbf{z}_t^i}$.

The probabilities of a new object and a spurious measurement are taken to be uniformly distributed over the observation volume V. In the implementation, the observation volume V is the intersection of the field of view (neglecting occlusions) and the soccer field. Thus, V

is a function of the robot's pose estimate and the camera's field of view. Its complement $\widetilde{V}$ is defined as the total area of the field without V.

$$p(Z_t|\theta_t, \Theta^j_{t-1}, Z_{0\ldots t-1}) = \prod_{i=1}^{m_k} [N_j(z^i_t)]^{\kappa_i} V^{-(1-\kappa_i)} \tag{6.13}$$

The quantity κ_i is another indicator variable which is 1, if z^i_t came from a known track and 0, otherwise.

$$\kappa_i = \begin{cases} 1 & , z^i_t \text{ came from a known track } i \\ 0 & , \text{otherwise.} \end{cases} \tag{6.14}$$

Substituting Eq. (6.13) and Eq. (6.8) into Eq. (6.7) yields the final expression for the conditional probability of an association hypothesis

$$\begin{aligned} P(\Theta^i_t|Z_{0\ldots t}) \;=\; & \frac{1}{c}\frac{\phi!\nu!}{m_k!}\mu_F(\phi)\mu_N(\nu)V^{-\phi-\nu} \, * \\ & \prod_{i=1}^{m_k}[N_j(z^i_t)]^{\kappa_i}\left\{\prod_k (P^k_D)^{\delta_k}(1-P^k_D)^{1-\delta_k}(P^k_T)^{\tau_k}(1-P^k_T)^{1-\tau_k}\right\}P(\Theta^j_{t-1}|Z_{0\ldots t-1}) \end{aligned} \tag{6.15}$$

If the number of false alarms and new features are assumed to be Poisson distributed with densities λ_F and λ_N, respectively, then Eq. 6.16 reduces to

$$P(\Theta^i_t|Z_{0\ldots t}) = \frac{1}{c'}\lambda^\nu_N\lambda^\phi_F\prod_{i=1}^{m_k}[N_j(z^i_t)]^{\kappa_i}\left\{\prod_k (P^k_D)^{\delta_k}(1-P^k_D)^{1-\delta_k}(P^k_T)^{\tau_k}(1-P^k_T)^{1-\tau_k}\right\}P(\Theta^j_{t-1}|Z_{0\ldots t-1}) \tag{6.16}$$

The probability of an association hypothesis, is used to guide the pruning strategies of the Multiple Hypothesis Tracking algorithm (Reid, 1979; Cox and Leonard, 1994).

6.3.6 Choosing the Maximum Depth of a Track Tree

In this section the options for selecting the maximum depth N of the track trees are discussed. This is a crucial decision that has to be taken when a MHT approach is applied.

On the one hand, choosing a deeper maximum will increase the accuracy of the tracks. This allows for the integration of observations from several more time steps. The idea is that with more integrated observations the decision where the track tree has to be pruned becomes clearer and in the best case even unique. Thus, the ideal track tree, as far as track accuracy is concerned, is a track tree with an infinite depth. This would postpone the pruning decision until all observations have been integrated.

On the other hand the required computation time for every update cycle grows exponentially with the maximum depth of the track trees. As robot soccer is an especially dynamic scenario, it requests for fast response times and the track tree have to be shallow.

Thus, the overall decision that has to be made is the decision between the maximum accuracy of the tracks and the response times of the algorithm that allows to meet the real times constraints of the scenario.

N	f (Hz)	t_{min}	$\bar{t}$	t_{max}	N	f (Hz)	t_{min}	$\bar{t}$	t_{max}
1	1015	0.0002	0.0009	0.29	15	35	0.0004	0.0279	0.63
4	219	0.0004	0.0045	0.30	20	20	0.0005	0.0480	1.30
7	140	0.0004	0.0070	0.31	25	13	0.0007	0.0716	2.49
10	77	0.0004	0.0129	0.40	30	9	0.0007	0.1023	2.70

Table 6.1: Update times of the CODT algorithm for various depths of the track trees.

An obvious choice for the maximum depth N of the track trees is to choose N equal to the number of robots R that contribute observations to the team. Assuming that the track trees have already grown to their maximum depth in the previous iteration, this allows for the following procedure[2]:

1. In every time step R robots grab an image, process this image in order to perform opponent observations, and broadcast these observations to their teammates.

2. After a robot has performed its own set of observations it iterates the CODT algorithm once. Its observations are integrated into the track trees, the depth of the trees grows by one, and the trees are pruned down to the maximum depth.

3. Then it receives the set of observations of its teammates and iterates the CODT algorithm again for every set. Again, as part of every iteration observations are integrated into the trees, the depth of the tree grows, and the trees are pruned.

4. Finally, the last set of hypotheses containing the current root nodes of all hypotheses trees is extracted an used as input for the action selection and path planning routines.

With every iteration of this procedure it is assured that all sets of observations performed in the previous time step have been fused by the CODT algorithm. Thus, the action selection and path planning routines receive the opponent positions from the previous time step as input. This delay is well compensated by the increased accuracy and completeness of the opponents' position estimates. In particular, at a frame rate of 30 Hz this delay is only 33 ms. Opponent robots moving at the high speed of 1 m/s will move at most for 3.3 cm during that time.

Given a team of four robots the above procedure requires the CODT algorithm to be capable of running at a frequency of 120 Hz. Table 6.1 displays the achievable update times and frequencies of the CODT algorithm for various depths of the track trees. The CODT algorithm was run over all log files recorded during 2001 and 2002 and the update times were measured. This experiment took place on a Pentium class computer with 700 MHz.

[2]For simplicity it is assumed that the cameras of all robots are synchronised and that communication times for broadcasts are negligible. In fact, during a running match an IP packet sent by a `ping` command requires 2 ms for its round trip between two robots.

6.4 Related Work

As mentioned earlier, a large body of research towards solving the data association problem, establishing motion correspondences, and the general problem of multiple object tracking with multiple mobile sensors exists. Again the two classes of outlier sensitive estimation techniques and robust estimation techniques can be distinguished. In the following sections different variants of these two main approaches and their applications to various types of sensors within the RoboCup domain and other robotic applications are discussed.

6.4.1 Outlier Sensitive Estimation Techniques

The approach of Stroupe et al.: Stroupe, Martin, and Balch (2001) present a first approach towards object tracking in a multi-robot team. Based on observations performed by a vision sensor, a Kalman filter is used to track one uniquely observable object, i.e. the ball of the RoboCup scenario. Since the ball is observable without any ambiguities, this scenario is well behaved and the algorithm does not have to deal with the data association problem. Furthermore, this approach does not take the accuracy of the observing robot's pose into account. First results with almost stationary settings (at most one robot is moving) demonstrate, that the tracking accuracy increases with the application of multiple sensors.

The approach of Dietl and Gutmann: Dietl, Gutmann, and Nebel (2001) develop two different tracking techniques for the RoboCup scenario: (1) for multiple object tracking, i.e. opponent robots, based on observations with several high precision sensors such as laser range finders, and (2) for single object tracking, i.e. the ball, with multiple noisy sensors, such as vision sensors. The position estimates generated by both techniques are stored in the team's world model (Gutmann et al., 1999).

The multi-object tracking algorithm is based on Kalman filtering and opponent observations that are performed with laser range finders. The data association problem is solved by the geometric method developed by Veloso et al. (1998) which assigns measurements to tracks by minimising the sum of squared error distances between observations and tracks. This is done by first computing a distance table of all pairs. This table is then searched for a combination where no two observations correspond to the same track, the number of assignments is maximal, and where the sum of all distances is minimised.

The single-object tracking method involves a combination of Kalman filtering and Markov localisation. This method aims at the combination of both methods' advantages, fast updates and high accuracy on the one hand and robustness towards noise and detection of outliers on the other (see also (Gutmann, 2002)). This is achieved by tracking the ball with a Kalman filter and using a Markov grid as observation filter. The multi-modal probability grid is used to distinguish which ball measurement should be integrated by the Kalman filter and which not. After an update of the grid with the latest measurement the most likely ball position is determined. Only measurements that are close to the most likely position are fused into the Kalman filter and all others are considered as outliers. Furthermore, if the current state of the Kalman filter does not correspond to the most likely ball position in the grid, the Kalman filter can be reinitialised with this position. Experimental results prove the validity and applicability of these techniques.

6.4.2 Robust Estimation Techniques

The approaches of Bar-Shalom et al.: The Probabilistic Data Association filter (PDAF) (Bar-Shalom and Fortmann, 1988; Cox, 1993) is an extension of the Kalman filter that uses a Bayesian approach to the problem of data association, or how to update the state when there is a single object and possibly no measurements or multiple measurements due to noise. Rather than possibly erring by choosing the nearest neighbour (Bar-Shalom and Fortmann, 1988; Cox, 1993) or data closest to what is expected in order to update the state, the PDAF hedges its bets by weighting the influence of the various candidate measurements based on two assumptions. First, it assumes that there is exactly one target giving rise to one true measurement which may sporadically disappear either because the object is temporarily occluded or because of suboptimal feature detection. Second, the PDAF assumes that all other measurements are false and arise from a uniform noise process. The relevant step in the Kalman filter is the computation of the innovation. The PDAF introduces a notion of the combined innovation, computed over the measurements detected at a given time step as the weighted sum of the individual innovations. Though the PDAF is able to cope with noisy and false measurements, it can only track one object.

To overcome this problem the Joint Probabilistic Data Association filter (JPDAF) was proposed (Bar-Shalom and Fortmann, 1988; Cox, 1993). It maintains a Kalman filter for every tracked object and introduces a kind of exclusion principle that prevents two or more trackers from latching onto the same object by calculating target-measurement association probabilities jointly. The JPDAF associates all measurements with each track to form a combined weighted innovation, for every track. This weighted innovation is then applied in the standard Kalman filter update equation for each object that is currently tracked. In order to compute the weighted innovation the association probabilities for every target-measurement association are computed on the basis of the distances between the object's predicted measurement and the actual measurement. Unlike associations have very low probabilities and, thus, have almost no influence on the weighted innovation. Consequently, the JPDAF disregards infeasible associations and, thus, avoids inappropriate state convergence. The major drawback of the JPDAF is that the number of tracked objects remains fixed and has to be known in advance. The precise formula for the probability of each particular target-measurement association in the PDAF and JPDAF framework is given in (Bar-Shalom and Fortmann, 1988; Cox, 1993).

The approach of Rosencrantz: A new variable dimension particle filter algorithm for tracking the location of objects under prolonged periods of occlusion was proposed by Rosencrantz, Gordon, and Thrun (2003). The algorithm has been implemented for a multi-robot system playing the popular game of laser tag. The object of the game is to search for and tag opponents that can move freely about the environment. The algorithm can cope efficiently with variable numbers of targets, through mechanisms that dynamically increase and decrease the number of particle filters. The data association problem is alleviated through a novel tracking technique that represents objects by roles, such as "the robot which went to the left" or "the robot which went to the right", instead of individual identity. When searching for objects, the individual agents greedily maximise their information gain, using a negotiation technique for coordinating their search efforts. Experimental results obtained with a physical robot system equipped with laser sensors in large-scale indoor environments demonstrate the applicability of this approach.

Although at first sight the laser tag scenario seems to be similar to the RoboCup scenario they have four fundamental differences. (1) RoboCup is played on a field, with almost no landmarks for localisation, while laser-tag is usually played in indoor corridor or tunnel system like environments, which provide much more information for localisation. (2) The RoboCup scenario contains two teams of robots with almost identical and indistinguishable robots and, thus, make the data association and multiple object tracking problem more difficult. (3) Due to the fast and dynamic nature of soccer and the limited size of the field, occlusions of agents happen more often than in laser tag. Finally, (4) the movement directions of robots change much more abruptly and robotic platforms with holonomic drives are applied in RoboCup. This makes the assignment of roles, as proposed by Rosencrantz, Gordon, and Thrun (2003), a lot more difficult than in the game of laser tag.

In contrast to the above approach, the CODT algorithm presented in this thesis tackles the multiple object tracking problem in the far more complex RoboCup scenario with less accurate and noisy vision sensors.

The approaches of Schulz et al.: The Sample-Based Joint Probabilistic Data Association Filter (SJPDAF) (Schulz et al., 2001b; Schulz et al., 2001a; Schulz et al., 2003) is an extension of the JPDAF, that combines the advantages of sample-based density approximations with the efficiency of JPDAFs. SJPDAF maintains a particle filter for every object that is currently tracked and applies the idea of JPDAF to assign measurement to the individual tracks. Furthermore, the algorithm also maintains a probability distribution over the number of objects tracked. This allows for the tracking of an arbitrary number of objects and, thus, to initiate and to terminate object tracks. The approach has been implemented and tested on a real robot using laser-range data. The experiments illustrate that the SJPDAF algorithm is able to robustly keep track of multiple moving persons, while the robot is in motion.

A further approach to tracking multiple objects, that combines the accuracy benefits of anonymous sensors and the identification certainty of id-sensors is presented in Schulz, Fox, and Hightower (2003). In this work Rao-Blackwellised particle filters (RBPF) (see also Doucet et al. (2000)) are used to estimate object locations. Each particle represents the association history between Kalman filtered object tracks and observations. After using only anonymous sensors until id estimates are certain enough, id assignments are sampled as well resulting in a fully Rao-Blackwellised particle filter over both object tracks and id assignments. This approach has successfully been tested using data collected in an indoor environment equipped with a large number of stationary sensors.

In contrast to SJPDAF and RBPF the CODT algorithm requires less computational power if the pruning parameters are carefully selected and can handle observations performed by multiple mobile robots with uncertain positions.

The approach of Hue et al.: A further approach to multiple object tracking with particle filters was proposed by Hue, Le Cadre, and Perez (2000). The multiple target particle filter (MTPF) is capable of tracking a fixed number of objects with one single set of particles. The association probabilities of a measurement and a particle from the particle set are computed on the basis of the Gibbs sampler (see (Hue, Le Cadre, and Perez, 2001)). Despite its computational complexity, this algorithm lacks the ability to initiate new tracks and terminate vanished tracks.

The approaches of Cox et al.: The CODT algorithm presented in this thesis is most closely related to the multiple hypothesis tracking approach used in (Cox, Rehg, and Hingorani, 1993; Cox and Leonard, 1994; Cox and Hingorani, 1996). Cox, Rehg, and Hingorani (1993) present a multiple hypothesis approach to edge grouping and contour segmentation. Cox and Leonard (1994) use multiple hypothesis tracking to model a static environment consisting of corners and walls. Their work on multiple hypothesis tracking is extended with this thesis, in which their method is applied to a more challenging application domain with multiple moving observers with uncertain positions. In addition, tracking is performed at an object level rather than at a feature level.

The approach of Rull: A humoristic[3] approach to the multiple object tracking and data association problem is presented in (Rull, 1993). The autonomous train spotter BARRY is shown to be able to perform the task of train spotting for many hours, without requiring human interaction. Theoretically, the system can spot and track trains travelling at relativistic speeds. This allows for the application of BARRY to obvious future applications in the domain of space-based spotting, such as RoboCup in space and planet bowling. Two appropriate and revolutionary camera models such as (1) the black hole camera model and the (2) enormous-mass camera model are also proposed. The article is well worth reading and laughing about[3].

Various other approaches: Further approaches to the multiple object tracking problem can be found in the Greedy Optimal Assignment (GOA) tracker (Veenman, Reinders, and Backer, 2003), the Restrained Optimal Assignment Decision (ROAD) tracker (Veenman, Reinders, and Backer, 2001), and the Joint Likelihood filter (JLF) (Rasmussen and Hager, 2001). More work related to contour oriented vision based particle filter tracking can be found in (Isard and Blake, 1996b; Blake and Isard, 1998; Isard and Blake, 1998) and (Comanicu, Ramesh, and Meer, 2003).

6.5 Conclusions

This chapter presented a fast (real time capable) and accurate vision-based Cooperative Object Detection and Tracking (CODT) procedure. CODT allows a team of robots to observe dynamic obstacles (robots and humans) and to track them at frame rate with high precision. Object observations are performed by a robot based on a general but simple object model of the opponents. With every set of observations the CODT updates the probability density describing the states of the opponents.

A clear advantage of the CODT algorithm is that it is capable of taking observations into account that were performed by several teammates. By doing this, a team of robots performs

[3]Don't take this paragraph seriously. However, I would like to acknowledge that you seem to be deeply interested in the contents of this thesis and that you belong to the class of my best and most enthusiastic readers. Thanks a lot!

cooperative opponent detection and tracking, and is able to gain a more precise and complete view of the environment than a single robot can with its own camera.

The algorithm and its properties are extensively evaluated in a series of real world experiments in Chapter 7.

Chapter 7

Experimental Results

7.1 Introduction

The Cooperative Incremental Iterative Localisation algorithm and the Cooperative Opponent Detection and Tracking algorithm presented in the previous chapters constitute the backbone of the control system of the autonomous RoboCup team, *The AGILO RoboCuppers*. Their performance is evaluated in this chapter with a series of large style experiments. During all real world experiments both algorithms run on all four robots simultaneously in a competitive match environment, i.e. robot soccer matches during a robot soccer world cup (RoboCup 2001, Seattle, USA) or friendly matches during the open day of the Munich University of Technology, Munich, Germany in 2002 (see Figure 7.1). The experiments are performed under ordinary RoboCup match conditions, as defined in (Bonarini et al., 2003), and in the presence of an audience. During the experiments the robots were completely autonomous and performed all actions based on their joint perceptions of the world. The only manual interaction was the transmission of a start or stop signal to the robots at the beginning and the end of the experiments, respectively. During the experiments all data generated by the robots, i.e. pose estimates of the robots and positions of the ball and opponent robots, are logged in a logfile. Furthermore, all experiments were also recorded in 2001 by an external video camera and in 2002 by a ground truth camera system (see Section 4.4) for evaluation purposes. While the former only allows for a qualitative evaluation of the experiments, the latter also allows for a quantitative evaluation of track accuracies, e.g. by providing Root Mean Square Errors (RMSE).

The remainder of this chapter proceeds as follow: First the CIIL algorithm is be used to estimate the relative position of the ground truth camera system above the RoboCup field. Then the accuracy of the ground truth camera system is accessed. The CIIL algorithm is tested under various conditions and situations, and its accuracy is determined with the ground truth camera system. Then the CODT algorithm is tested and its accuracy is determined. Finally, the performance of both algorithms is evaluated in two large style real world experiments.

(a) (b) (c)

Figure 7.1: Friendly matches between *The AGILO RoboCuppers* and *The Ulm Sparrows* at the open day of the Munich University of Technology in 2002. (a) The field, (b) a match situation and (c) the coach area of *The AGILO RoboCuppers*.

7.2 The Ground Truth Camera System

The ground truth camera system was developed in order to evaluate the approaches presented in this thesis. It is designed to determine the position of the ball and the poses of all robots on the field as accurately as possible. The backbone of the system is a blob analysis algorithm, that extracts the pixel coordinates of all regions of interest. An inverse camera model (see Section 2.4.6) is then used to transform pixel coordinates to 3D world coordinates. A precondition for the successful application of an inverse camera model is that the exact pose of the camera, relative to the field, is known. Determining this pose with high accuracy with a simple measuring device, such as a metering rule, is not impossible. However, once a rough first pose estimate has been generated, it can iteratively be refined through the application of the CIIL algorithm.

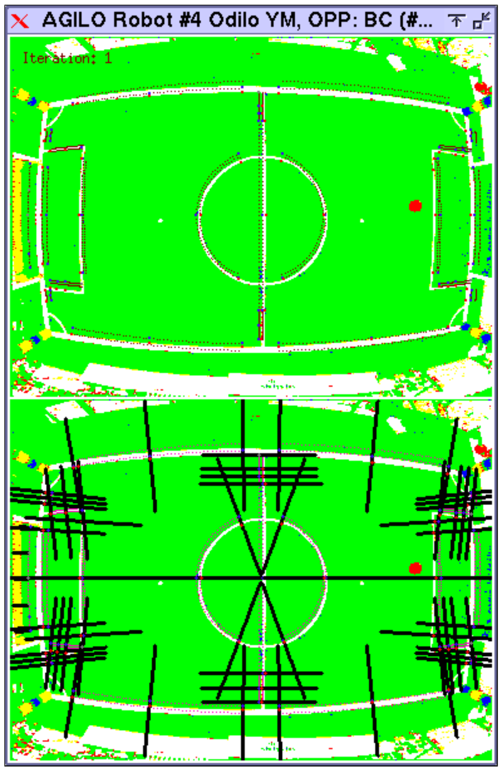
7.2.1 Localisation of the Ground Truth Camera System

In this section the CIIL is used in order to localise the ground truth camera system relative to the field. The results will show that a coarse initialisation of the the CIIL algorithm is sufficient to determine an accurate position for the camera system.

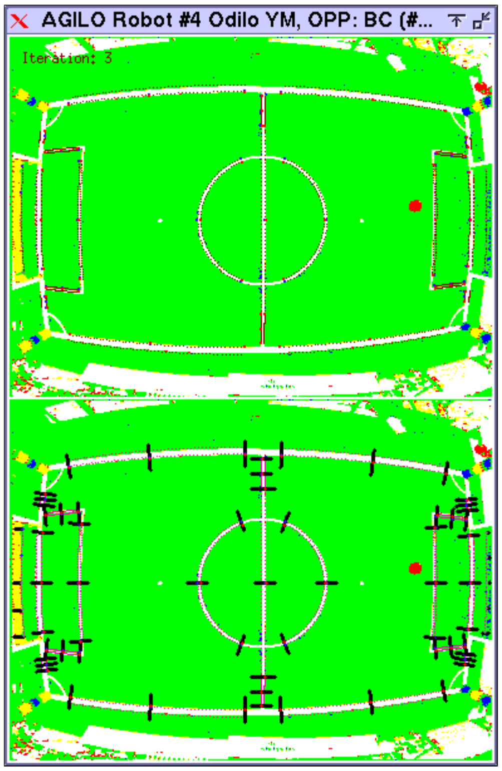
Figure 7.2 illustrates the application of the CIIL algorithm to the pose estimation process of the ground truth camera system. The camera system was placed at an approximate height of 4.5 m above the centre of the field, facing downward. The six dimensional relative pose of the camera system is initialised with

$$\begin{aligned}\bar{\mathbf{x}}_2 &= (x, y, z, \phi_x, \phi_y, \phi_z)^T \\ &= (0.0, 0.0, 4.50, 180.0, 0.0, 0.0)^T\end{aligned}$$

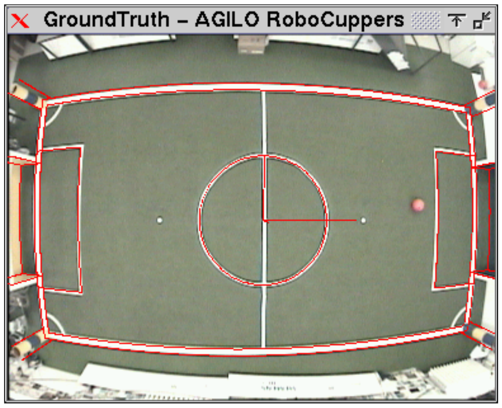
and the CIIL algorithm is run for three iterations. This process is illustrated in Figure 7.2. Subfigures 7.2(a) and 7.2(b) show the projected field model (top) before the first and the third iteration of the algorithm, respectively. At the bottom, the search for correspondences between the field model and the image data is displayed (black lines). The adaptation of the search interval is clearly visible, its width decreases in proportion with the decreasing



(a)



(b)



(c)

Figure 7.2: Localisation of the ground truth camera above the field with the CIIL algorithm. (a) Projected field model and search for correspondences before the first and (b) before the third iteration. (c) Projected field model after the third iteration.

uncertainty of the estimate. The third iteration step estimates the pose of the ground truth camera system as follows:

$$\bar{\mathbf{x}}_2 = (-0.25, 0.1, 4.39, 179.58, -2.85, 0.07)^T$$

Quantitatively this estimate can be assumed to be correct. Figure 7.2(c) also illustrates the qualitatively correctness of this estimate. The overlaying field model was projected into the image using the above estimate. Minor incorrections of the left penalty area lines reveal that these lines were not drawn on the field as accurately as they should have been. This fact was further investigated with a metering rule and these lines were found to have been a couple of centimetres too long.

7.2.2 Evaluation of the Ground Truth Camera System

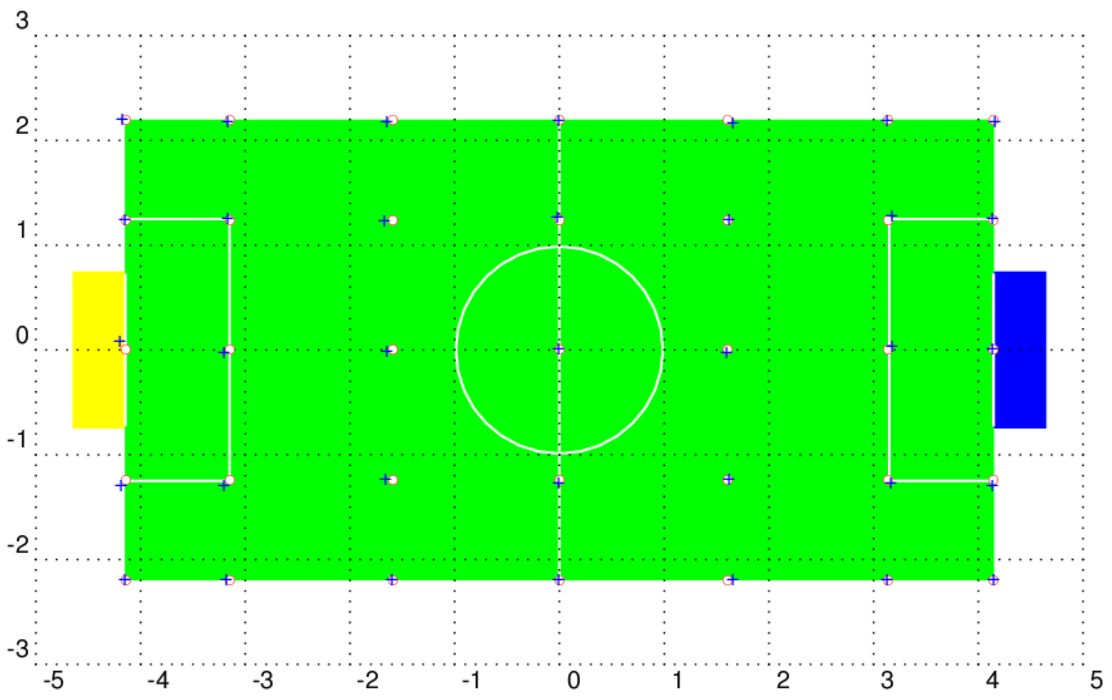
This section assesses the accuracy with which the ground truth camera system is able to determine the positions and orientations of objects on the field. The mean error for ball positions is around 3 cm, and the mean position and orientation error for robots is approximately 10 cm and 5^o, respectively.

Once the precise relative pose of the ground truth camera system is known it can be used to estimate the world coordinates of the ball and of the robots. In the following, the accuracy of these estimates is investigated.

For this the ball was put on 35 predefined positions on the field and its positions were determined by the ground truth camera system. This experiment is displayed in Figure 7.3. The exact predefined positions are marked by a circle, the positions determined by the ground truth camera system are marked by a cross. The quantitative data for this experiment can be found in the table at the bottom of Figure 7.3. For each pair of positions the Euclidean distance was computed and the minimum, mean, and maximum distances of all pairs were determined.

The ground truth for the ball location is found to be quite accurate. A mean error of 3.2 cm is observed for all positions. Displacements range from 0.7 cm to 7.3 cm, respectively. In general, the displacement errors are small in the centre of the field and increase towards the boundaries of the field. The good accuracy of the ball's ground truth can be explained with the relatively simple and unique geometry of the ball. The increasing displacement errors towards the field boundaries are a result of the fact that the ball is more observed from the side, than from the top. Thus, the ground truth position is mistakenly estimated to be further towards the field boundaries, than it actually is. This problem can only be overcome by a more elaborate feature extraction and blob analysis process, or through the application of more cameras.

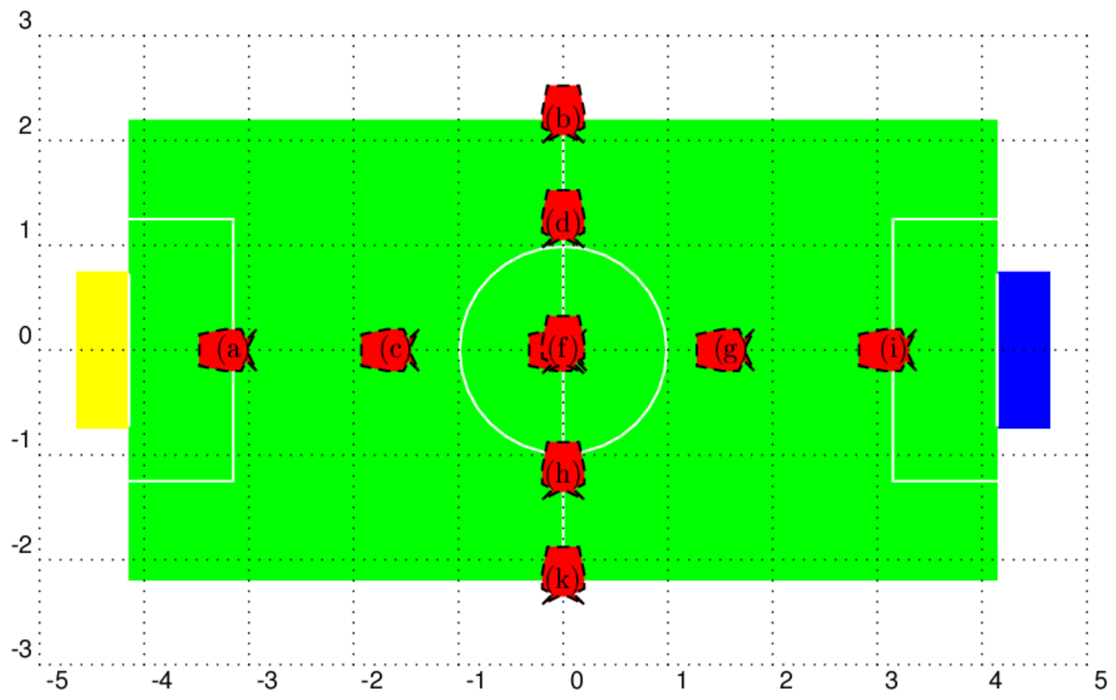
In the next experiment, the localisation accuracy of the ground truth camera system for robots is accessed. Figure 7.4 displays the experimental setup. A robot is placed at 10 different positions on the field and its pose is determined by the ground truth camera system. The quantitative data for this experiment can be found in the table at the bottom of Figure 7.4. The left, middle and right column contain the effective pose of the robot, the pose observed by the ground truth system, and the observation errors, respectively.



x/y	y = 2.20	y = 1.25	y = 0.00	y = -1.25	y = -2.20
x = 4.15	4.17,2.18 **0.028**	4.15,1.26 **0.010**	4.15,0.01 **0.010**	4.15,-1.29 **0.040**	4.16,-2.19 **0.014**
x = 3.15	3.15,2.19 **0.010**	3.18,1.28 **0.042**	3.18,0.03 **0.042**	3.17,-1.28 **0.036**	3.15,-2.21 **0.010**
x = 1.60	1.66,2.17 **0.067**	1.62,1.25 **0.020**	1.61,-0.02 **0.022**	1.63,-1.25 **0.030**	1.66,-2.21 **0.061**
x = 0.00	0.004,2.19 **0.011**	-0.02,1.27 **0.028**	0.0044,-0.0054 **0.007**	-0.01,-1.27 **0.022**	-0.01,-2.20 **0.010**
x = -1.60	-1.64,2.18 **0.045**	-1.67,1.23 **0.073**	-1.64,-0.01 **0.041**	-1.65,-1.23 **0.054**	-1.60,-2.19 **0.010**
x = -3.15	-3.17,2.18 **0.028**	-3.17,1.26 **0.022**	-3.20,-0.03 **0.058**	-3.20,-1.30 **0.071**	-3.18,-2.21 **0.032**
x = -4.15	-4.17,2.20 **0.020**	-4.14,1.24 **0.014**	-4.19,0.008 **0.041**	-4.18,-1.29 **0.050**	-4.15,-2.19 **0.010**

RMSE: Min **0.007 m**, Mean **0.032 m**, Max **0.073 m**

Figure 7.3: Ground truth accuracy for a ball at various positions on the field.



No.	x	y	ϕ	x	y	ϕ	Δ	$\Delta\phi$
(a)	-3.15	0.0	0	-3.36	-0.04	-2.15	0.21	2.15
(c)	-1.60	0.0	0	-1.80	0.02	1.89	0.20	1.89
(e)	0.0	0.0	0	-0.01	-0.01	3.40	0.01	3.40
(g)	1.60	0.0	0	1.68	0.01	-3.51	0.08	3.51
(k)	3.15	0.0	0	3.25	-0.03	1.14	0.10	1.14
RMSE Δ: Min **0.01 m**, Mean **0.12 m**, Max **0.21 m** RMSE $\Delta\phi$: Min **1.89^o**, Mean **2.41^o**, Max **3.51^o**								
(b)	0.0	2.20	-90	-0.05	2.30	-81.12	0.11	8.88
(d)	0.0	1.20	-90	0.00	1.33	-87.02	0.13	2.98
(f)	0.0	0.0	-90	0.00	0.04	-86.19	0.04	3.81
(h)	0.0	-1.20	-90	0.01	-1.25	-87.34	0.05	2.66
(i)	0.0	-2.20	-90	-0.01	-2.28	-79.39	0.08	10.61
RMSE Δ: Min **0.04 m**, Mean **0.082 m**, Max **0.13 m** RMSE $\Delta\phi$: Min **2.66^o**, Mean **5.78^o**, Max **10.61^o**								

Figure 7.4: Ground truth accuracy for a robot at various positions on the field.

The ground truth position estimate for robots is less accurate than for ball. The mean position and orientation error are estimated to be around 10 cm and 5^o, respectively. Observed displacements range from 1 cm to 21 cm, and observed orientation errors range from 1.8^o to 10.6^o. Again, these displacements are a result of the relatively simple feature extraction and blob analysis process used by the ground truth camera system. The main reason why the localisation accuracy for robots is worse than for a ball can be found in the fact that robots are usually asymmetric and their own pose is estimated relative to their vertical rotation axis and not to their centre of gravity (see also Section 5.2.1). However, the generic algorithm of the ground truth camera system determines the position of all objects on the basis of their blob's centre of gravity. This leads to even greater localisation error at the field boundaries, when the robots are observed more from the side. The only way to increase the localisation accuracy for robots is to apply a computationally more expensive model based localisation approach for the ground truth system.

The errors generated by the state estimation techniques applied in this thesis are expected to be in the same order of magnitude as the errors of the ground truth system assessed above. The validity of this assumption will be proven throughout the next sections.

7.3 Stationary localisation

This section investigates the localisation accuracy of the CIIL algorithm and its applicability to different robotic platforms. In particular, a wheeled AGILO Pioneer 1 robot and a quadruped Sony AIBO robot are used.

7.3.1 AGILO Robot

This experiment investigates the localisation accuracy for 3D and 6D estimates of the CIIL algorithm at several poses on the field. The mean position and orientation errors for the 3D estimate are found to be around 3.5 cm and 1^o, respectively. The algorithm performs slightly worse for the 6D estimate, with a mean position error of 6.5 cm and a mean orientation error of 3.4^o.

An AGILO robot is placed on the same positions on the field as in the experiments with the ground truth camera system (see Figure 7.4). The images observed by the robot at the different positions on the field are displayed in Figure 7.5. In order to verify the correctness of a pose estimate, this estimate is used to project the field model into the image.

For the experiments the CIIL algorithm is initialised with a pose that diverges from the true position and the true orientation by a Euclidean distance of 30 cm and 15^o, respectively. The CIIL algorithm is run for three iterations and the estimate is recorded. The quantitative results are shown in Table 7.1. Each experiment is described by four rows. The first row contains the robot's pose as measured with a metering rule. The second row contains the pose observed by the ground truth camera system. Row number three and four contain the 3D and 6D poses and the root of the covariance matrix's diagonal elements ($\pm\sigma$) as estimated by the CIIL algorithm, respectively. As before, the localisation error is measured by the Euclidean distance Δ between the estimate and the position measured by the metering rule. The orientation error $\Delta\phi$ is determined by the sum of the absolute errors of all rotational

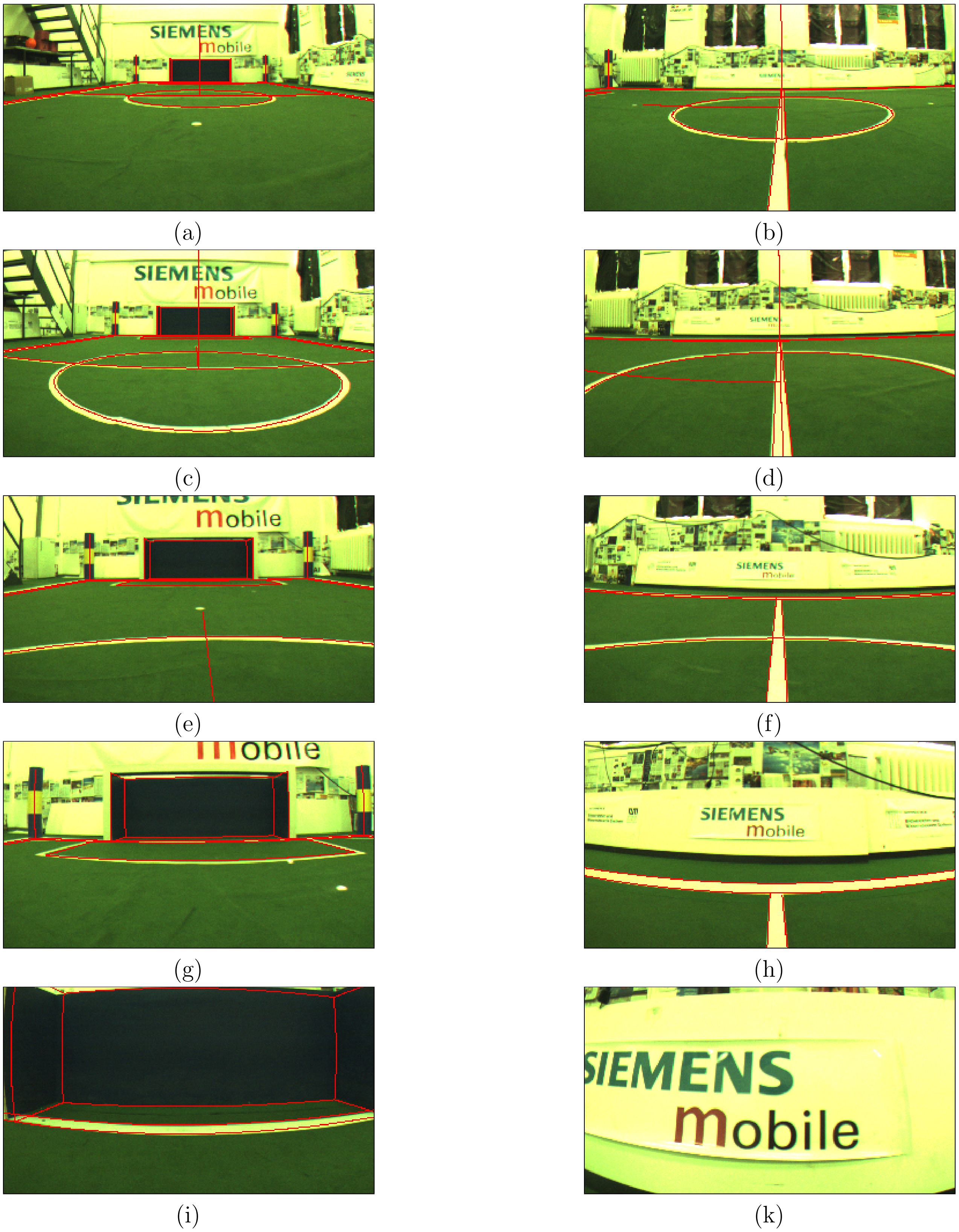


Figure 7.5: Localisation experiments at 10 different poses with superimposed field model.

Experiment		x	y	z	ϕ_x	ϕ_y	ϕ_z	Δ	$\Delta\phi$	Eq	Err	%
(a)	MR	−3.15	0.00	–	–	–	0.00					
	GT	−3.36	−0.04	–	–	–	−2.15	0.21	2.15			
	$3D$	−3.15	−0.03	–	–	–	1.83	0.03	1.83	40	10.61	10.73
	$\pm\sigma$	±0.07	±0.07	–	–	–	±0.06					
	$6D$	−3.16	−0.08	−0.02	0.69	−0.47	2.25	0.08	3.41	43	18.35	11.45
	$\pm\sigma$	±0.07	±0.08	±0.03	±0.57	±0.45	±0.65					
(c)	MR	−1.60	0.00	–	–	–	0.00					
	GT	−1.80	0.02	–	–	–	1.89	0.20	1.89			
	$3D$	−1.63	−0.01	–	–	–	1.09	0.03	1.09	46	8.47	17.73
	$\pm\sigma$	±0.03	±0.05	–	–	–	±0.57					
	$6D$	−1.63	−0.02	−0.06	1.36	−2.19	1.88	0.04	5.43	49	8.30	16.18
	$\pm\sigma$	±0.03	±0.05	±0.01	±0.51	±0.29	±0.53					
(e)	MR	0.00	0.00	–	–	–	0.00					
	GT	−0.01	−0.01	–	–	–	3.40	0.01	3.40			
	$3D$	0.03	0.04	–	–	–	0.48	0.05	0.48	42	6.58	15.30
	$\pm\sigma$	±0.02	±0.05	–	–	–	±0.61					
	$6D$	0.02	−0.01	0.01	1.38	0.56	1.33	0.02	3.27	40	10.83	16.48
	$\pm\sigma$	±0.03	±0.09	±0.02	±0.060	±0.41	±1.16					
(g)	MR	1.60	0.0	–	–	–	0.00					
	GT	1.68	0.01	–	–	–	−3.51	0.08	3.51			
	$3D$	1.64	0.02	–	–	–	0.82	0.04	0.82	65	2.72	14.82
	$\pm\sigma$	±0.01	±0.03	–	–	–	±0.56					
	$6D$	1.64	0.01	0.02	0.02	−0.42	0.97	0.04	1.41	66	14.42	18.92
	$\pm\sigma$	±0.01	±0.03	±0.02	±0.38	±0.69	±0.60					
(i)	MR	3.15	0.0	–	–	–	0.00					
	GT	3.25	−0.03	–	–	–	1.14	0.10	1.14			
	$3D$	3.09	0.12	–	–	–	−2.20	0.13	2.20	12	2.55	1.29
	$\pm\sigma$	±0.02	±0.05	–	–	–	±1.94					
	$6D$	3.10	−0.15	−0.01	2.00	−0.20	4.85	0.15	7.05	12	0.90	1.48
	$\pm\sigma$	±0.03	±0.19	±0.03	±1.45	±1.29	±5.35					

GT RMSE Δ: Min **0.01 m**, Mean **0.12 m**, Max **0.21 m**
GT RMSE $\Delta\phi$: Min **1.89**^{o}, Mean **2.41**^{o}, Max **3.51**^{o}

3D RMSE Δ: Min **0.03 m**, Mean **0.05 m**, Max **0.13 m**
3D RMSE $\Delta\phi$: Min **0.48**^{o}, Mean **1.28**^{o}, Max **2.20**^{o}

6D RMSE Δ: Min **0.02 m**, Mean **0.06 m**, Max **0.15 m**
6D RMSE $\Delta\phi$: Min **1.41**^{o}, Mean **4.11**^{o}, Max **7.05**^{o}

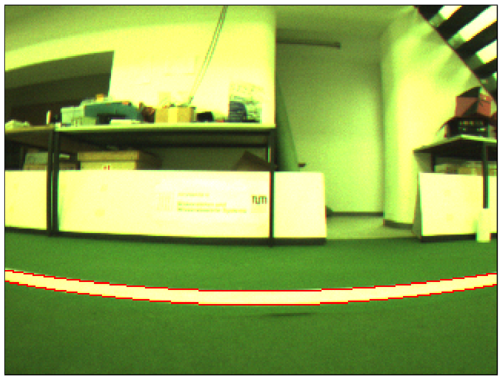
degrees of freedom. The last three columns summarise interesting runtime quantities of the CIIL algorithm. *Eq* specifies the number of correspondences established between the model and the image data. *Err* is an approximated χ^2 error measure and '%' indicates the percentage of image data accessed during the three iterations of the CIIL algorithm.

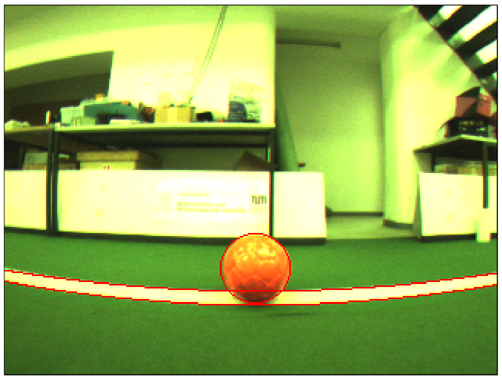
The CIIL algorithm performed very well in eight cases. The mean position and orientation

Experiment		x	y	z	ϕ_x	ϕ_y	ϕ_z	Δ	$\Delta\phi$	Eq	Err	%
(b)	MR	0.00	2.20	–	–	–	−90.00					
	GT	−0.05	2.30	–	–	–	−81.12	0.11	8.88			
	$3D$	−0.01	2.19	–	–	–	−88.13	0.01	1.87	14	2.10	5.80
	$\pm\sigma$	±0.00	±0.03	–	–	–	±0.57					
	$6D$	−0.00	2.21	0.00	0.94	−0.86	−88.49	0.01	3.31	13	0.75	6.81
	$\pm\sigma$	±0.01	±0.09	±0.09	±1.58	±2.68	±0.09					
(d)	MR	0.00	1.20	–	–	–	−90.00					
	GT	0.00	1.33	–	–	–	−87.02	0.13	2.98			
	$3D$	0.01	1.18	–	–	–	−89.42	0.02	0.58	10	3.14	1.08
	$\pm\sigma$	±0.04	±0.20	–	–	–	±1.37					
	$6D$	0.00	1.10	0.02	1.48	0.85	−88.96	0.09	3.37	11	0.22	4.32
	$\pm\sigma$	±0.04	±0.63	±0.11	±0.99	±6.40	±1.24					
(f)	MR	0.00	0.0	–	–	–	−90.00					
	GT	0.00	0.04	–	–	–	−86.19	0.04	3.81			
	$3D$	−0.01	−0.04	–	–	–	−88.98	0.04	1.02	15	0.59	4.03
	$\pm\sigma$	±0.02	±0.03	–	–	–	±1.09					
	$6D$	0.00	−0.09	−0.01	−0.10	0.26	−89.04	0.09	1.32	16	0.30	6.73
	$\pm\sigma$	±0.03	±0.26	±0.09	±0.78	±1.69	±1.65					
(h)	MR	0.00	−1.20	–	–	–	−90.00					
	GT	0.01	−1.25	–	–	–	−87.34	0.05	2.66			
	$3D$	0.01	−1.21	–	–	–	−90.35	0.01	0.35	7	0.08	3.64
	$\pm\sigma$	±0.03	±0.03	–	–	–	±2.73					
	$6D$	0.00	−1.11	0.03	0.64	−1.20	−89.03	0.09	2.81	4	0.04	4.43
	$\pm\sigma$	±0.07	±0.97	±0.75	±1.86	±2.90	±4.69					
(k)	MR	0.00	−2.20	–	–	–	−90.00					
	GT	−0.01	−2.28	–	–	–	−79.39	0.08	10.61			
	$3D$	–	–	–	–	–	–	–	–	0	∞	0.00
	$\pm\sigma$	–	–	–	–	–	–	–	–	0	∞	0.00
	$6D$	–	–	–	–	–	–	–	–	0	∞	0.00
	$\pm\sigma$	–	–	–	–	–	–	–	–	0	∞	0.00
GT RMSE Δ: Min **0.04 m**, Mean **0.082 m**, Max **0.13 m**												
GT RMSE $\Delta\phi$: Min **2.66^o**, Mean **5.78^o**, Max **10.61^o**												
3D RMSE Δ: Min **0.01 m**, Mean **0.02 m**, Max **0.04 m**												
3D RMSE $\Delta\phi$: Min **1.02^o**, Mean **0.95^o**, Max **1.87^o**												
6D RMSE Δ: Min **0.01 m**, Mean **0.07 m**, Max **0.09 m**												
6D RMSE $\Delta\phi$: Min **1.32^o**, Mean **2.70^o**, Max **3.37^o**												

Table 7.1: Localisation error for 10 different poses on the field.

errors for the 3D estimate are around 3.5 cm and 1^o, respectively. The algorithm performed slightly worse for the 6D estimate, with a mean position error of 6.5 cm and a mean orientation error of 3.4^o. This result was expected as higher dimensional estimates have more degrees of freedom and, thus, pose a higher demand on the quality of the correspondences. Particularly


(a)


(b)

Experiment		x	y	ϕ_z	$ball_x$	$ball_y$	Δ	$\Delta\phi$	Eq	Err	%
(a)	MR	−1.60	1.22	90.00	–	–					
	$3D$	−1.80	1.23	89.72	–	–	0.20	0.28	3	0.03	3.01
	$\pm\sigma$	±1.08	±0.04	±0.32	$\pm\infty$	$\pm\infty$					
(b)	MR	−1.60	1.22	90.00	−1.60	−2.20					
	$3D$	1.60	1.23	89.69	−1.60	−2.18	0.01	0.31	7	0.04	3.73
	$\pm\sigma$	±0.03	±0.06	±0.28	±0.03	±0.06					

3D RMSE Δ: Min **0.01 m**, Mean **0.10 m**, Max **0.20 m**

3D RMSE $\Delta\phi$: Min **0.28^o**, Mean **0.29^o**, Max **0.31^o**

Figure 7.6: Localisation experiments with a ball.

in this scenario, position errors can be compensated by rotation errors and vice versa. This problem can be overcome by weighting position and rotation changes differently in the error function.

The pose estimate generated by the CIIL algorithm in experiment (i) is less accurate. In this case no vertical landmarks are visible and only horizontal landmarks are used to establish correspondences. Thus, the algorithm converges but cannot derive any information from the image data, that allows for estimating the y coordinate and the orientation correctly. The generation of a pose estimate failed completely for experiment (k). In this case no field landmarks are visible and, thus, the CIIL algorithm failed to produce a pose estimate. While the outcome of this experiment is not surprising, this example shows that the heuristics applied for establishing correspondences are robust against false matches and that the CIIL algorithm can successfully detect these situations. The problems identified in experiments (i) and (k) can successfully be overcome through the fusion of odometric and image data, as demonstrated in Section 7.4.

	RoboCup 2001				Open Day 2002			
Robot	Break	Hz	Match	Hz	Break	Hz	Match	Hz
#1	3637	2.0	16180	2.9	22087	8.9	35956	8.6
#2	12599	6.9	25061	4.4	22861	9.2	15502	3.7
#3	12551	6.8	24043	4.3	18536	7.5	17648	4.2
#4	14441	7.9	26356	4.7	18848	7.6	30251	7.2
Σ	43228	23.6	91640	16.3	82332	33.2	99357	23.7

Table 7.2: Absolute number and frequency of ball observations during the break and the match.

7.3.2 AGILO Robot with Ball

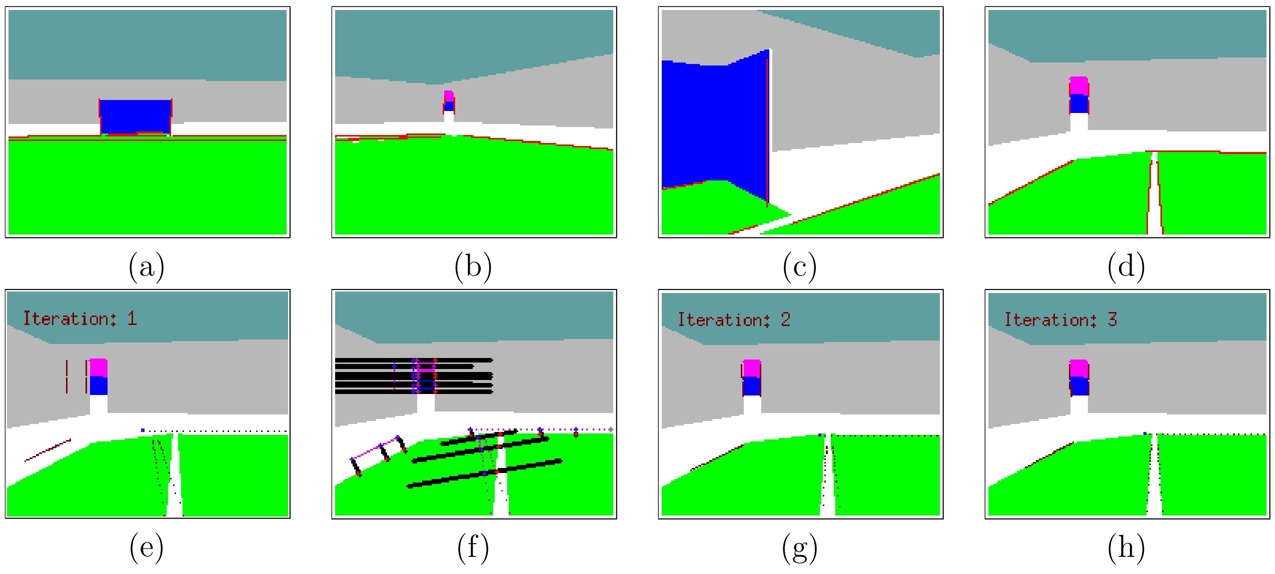
In this experiment the use and impact of ball observations made by teammates on the localisation accuracy of the CIIL algorithm is explored. It is demonstrated that observations of teammates can solve ambiguous localisation problems.

An AGILO Robot is placed such that it faces one of the side lines but cannot observe any of the unique vertical landmarks such as corner flagposts or goals. In the first experiment the robot can only observe the border field line. In the second experiment the ball is added and a ball observation performed by another team member is sent to the robot via wireless LAN and is used by the CIIL algorithm for localisation. The images observed by the localising robot are displayed in Figure 7.6. In order to verify the correctness of a pose estimate, the field model is projected into the image and is overlaid in the image.

As before, the CIIL algorithm is initialised with a pose that diverges from the true position and the true orientation by a Euclidean distance of 30 cm and 15^o, respectively. The CIIL algorithm is run for three iterations and the estimate is recorded. The quantitative results are shown in Table 7.6.

In experiment (a) the CIIL algorithm estimates the Y coordinate very accurately but fails to estimate the X coordinate correctly. This is due to the fact that only the border line of the field is visible and the CIIL algorithm is only able to extract range information between the robot and the field line from the image. In experiment (b) the CIIL algorithm is also provided with a rough (Euclidean distance of 20 cm) position estimate of the ball which was observed by another robot. Due to the additional information the robot is now able to determine its pose and the ball position very accurately, the robot position error is now only 1 cm and the ball position error is around 2 cm.

During a match every robot performs ball observations at a mean observation rate of 4 to 8 Hz (see Figure 7.2). Thus, the combined utilisation of image data and ball observations performed by team members is expected to greatly improve the localisation accuracy, and it enables the algorithm to solve several localisation problems which would not have been possible without ball observations.



Experiment		x	y	z	ϕ_x	ϕ_y	ϕ_z	Δ	$\Delta\phi$	Eq	Err	%
(a)	MR	0.00	0.00	0.15	0.00	0.00	0.00					
	$6D$	0.07	-0.01	0.16	0.01	1.09	-2.46	0.07	3.56	15	0.05	1.53
	$\pm\sigma$	± 0.05	± 0.08	± 0.09	± 1.98	± 2.63	± 3.22					
(b)	MR	0.00	0.00	0.15	0.00	0.00	-45.00					
	$6D$	0.00	0.10	0.14	-1.15	0.00	-41.26	0.10	3.78	12	0.09	7.34
	$\pm\sigma$	± 0.08	± 0.09	± 0.10	± 3.07	± 2.47	± 9.78					
(c)	MR	1.80	0.00	0.15	0.00	0.00	-45.00					
	$6D$	1.84	-0.03	0.13	0.83	-1.54	-40.29	0.05	7.08	7	1.53	4.46
	$\pm\sigma$	± 0.02	± 0.02	± 0.03	± 1.15	± 2.27	± 3.40					
(d)	MR	1.80	0.00	0.15	0.00	0.00	-90.00					
	$6D$	1.76	-0.01	0.17	0.54	0.51	-84.08	0.04	6.97	16	1.13	5.49
	$\pm\sigma$	± 0.03	± 0.11	± 0.03	± 1.10	± 1.85	± 2.18					
6D RMSE Δ: Min **0.03 m**, Mean **0.06 m**, Max **0.10 m**												
6D RMSE $\Delta\phi$: Min **3.56^o**, Mean **5.34^o**, Max **7.08^o**												

Figure 7.7: Localisation experiments with synthetic AIBO images.

7.3.3 AIBO Robot

This experiment demonstrates the applicability of the CIIL algorithm to other robotic platforms, such as the AIBO robot.

Because of its limited onboard computational power and its high dimensional state vector (at least 6D; more dimensions are required if the states of leg joints are also estimated), an AIBO robot is a promising candidate that can take full advantage of the real time capability and high accuracy of the CIIL algorithm.

The images used for the following experiments are generated by the AIBO simulator of the German Team (Burkhard et al., 2003; Burkhard et al., 2001). They differ from real images in two manners, they do not contain any noise nor any radial lens distortions, i.e. $\kappa = 0$. For every image the exact pose of the robot is known. For the experiments the CIIL algorithm is initialised with a pose that diverges from the true position and the true orientation by a Euclidean distance of 30 cm and 15^o, respectively. The CIIL algorithm is run for three iterations and the estimate is recorded. The images and the quantitative results of the experiments are shown in Figure 7.7. Subfigures 7.7(e) to 7.7(h) illustrate the search for correspondences and iterations performed by the CIIL algorithm for experiment (d).

The CIIL algorithm performs well in all four cases. The mean position and orientation errors for the 6D estimate are around 6.5 cm and 5.34^o, respectively. As expected, the estimates are less accurate when only distant features are visible (see Figure 7.7(a) and 7.7(b)) and are more precise in the presence of close features (see Figure 7.7(c) and 7.7(d)). The estimate of the covariance matrix is conservative and in all four cases only a small percentage of the image data is accessed (between 1.5 and 7.3 %).

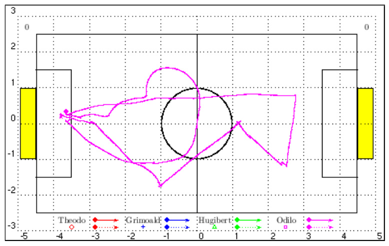
7.4 Localisation of a Robot in Motion

The state estimates generated by the CIIL algorithm are a result of the fusion of image and odometric data. This section evaluates the localisation accuracy of a joint estimate that is based on both types of data. Mean accuracies for robots travelling across the field are found to be between 12 to 16 cm.

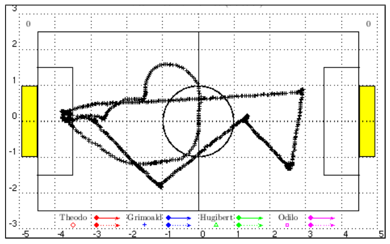
For the experiments, an AGILO Pioneer 1 robot is joysticked across the field for a couple of minutes and the robot's pose estimates as well as the robot's pose determined by the ground truth camera system are recorded in a log file. This data is then used to compute the minimum, mean, and maximum error of the state estimates.

Figures 7.8(a) and 7.8(d) depict the robot's trajectories according to the state estimates generated by the CIIL algorithm. Subfigures 7.8(b) and 7.8(e) display the robot's trajectories that were observed by the ground truth camera system. The joint display of both trajectories can be found in Figures 7.8(c) and 7.8(f).

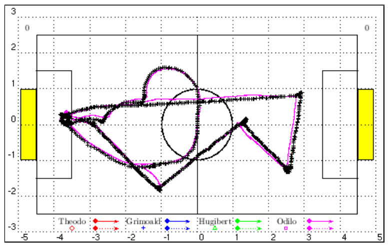
It is clearly visible that the trajectories match for most of the time and as such the localisation accuracy is quite high. The analysis of the log file revealed a mean accuracy (RMSE) of 16 cm and 12 cm for the first (a-c) and the second experiment (d-f), respectively. In both cases the minimum error is around 2 cm. The maximum error is 22 cm for the first and 18 cm for the second experiment. Given the mean localisation accuracy of the ground truth system which is around 10 cm, it can be assumed that the mean localisation error of the robot is much smaller than determined by these experiments. Furthermore, it should be noted that the achieved accuracy is sufficient for the robot soccer scenario.



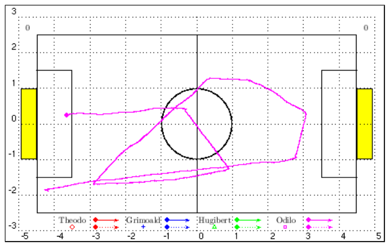
(a)



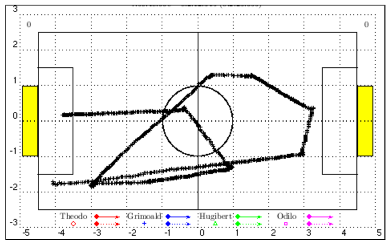
(b)



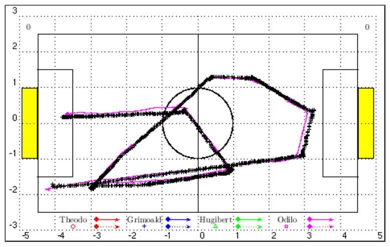
(c)



(d)



(e)



(f)

Figure 7.8: Localisation of a robot in motion.

7.5 Localisation of the Ball and Opponent Players

This section evaluates the accuracies with which a robot can detect the dynamic objects, such as the ball and the opponent players, of the RoboCup scenario. The detection routines are found to be quite precise for distances up to 2.8 m.

For the experiments an AGILO robot is placed in its own penalty area and a ball or an opponent player is moved across the field. The position estimates of the ball and the opponent player generated by the AGILO robot, as well as the filtered trajectories of the CODT algorithm, are recorded in a log file. Again, this data is used to evaluate the accuracy of the estimates.

Figure 7.9(a) depicts the ball's position estimates (dotted line) of the AGILO robot and the ball's position (black circles) determined by the ground truth system. Figure 7.9(c) depicts the observed opponent positions of the AGILO robot (green triangle) and the ground truth system (black triangle). Figures 7.9(b) and (d) plot the filtered trajectory of the CODT algorithm for the ball and the opponent robot, together with the ground truth data.

Ball and opponent observations can be performed up to a distance of approximately 7 m. The observations are quite accurate for distances up to 2.8 m with a maximum error of 0.2 m. The maximum error at 7 m is about 1.5 m. This uncertainty is appropriately represented in the observations covariance matrix. The uncertainty can be explained with the the fact that the camera of a robot is fixed parallel to the floor at a very low height of around 30 cm. The objects' trajectories as determined with the CODT algorithm reflect the accuracies of the observations. The trajectories are smooth and accurate for up to a distance of 3 m. It is noteworthy that noisy observations are detected by the CODT algorithm and are discarded accordingly.

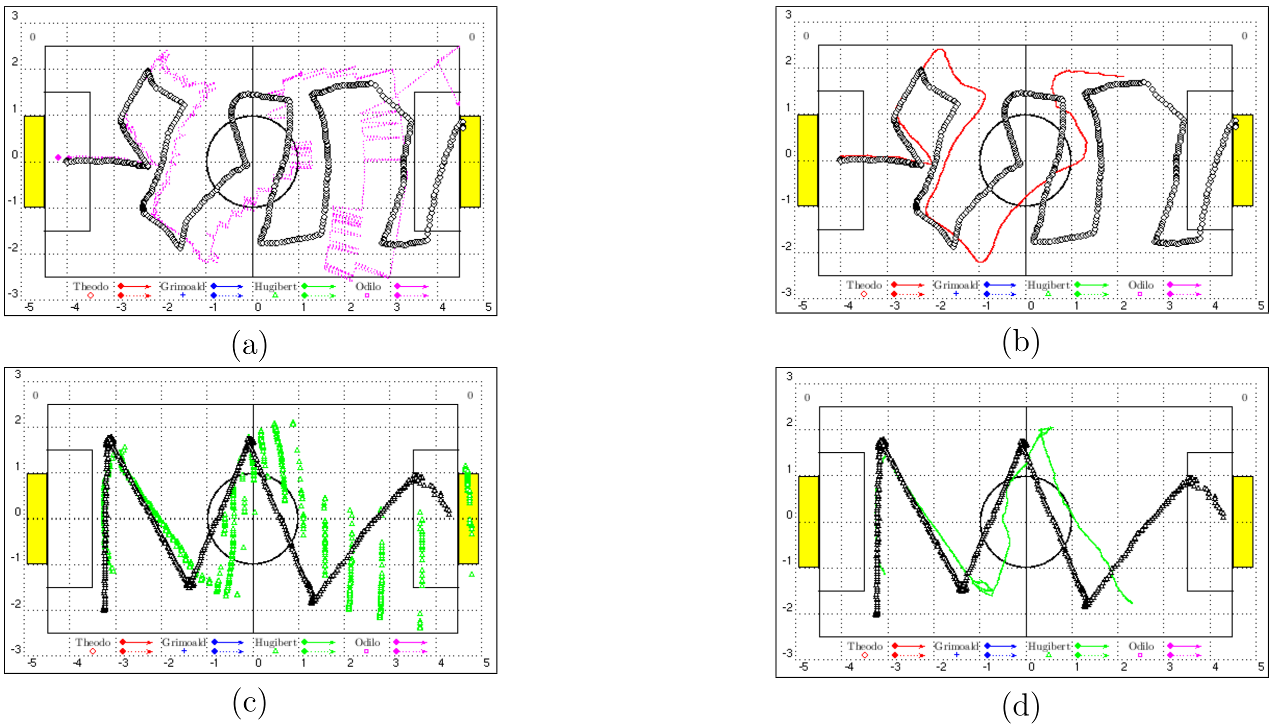


Figure 7.9: Localisation of (a,b) the ball and (c,d) opponent players.

7.6 CIIL and CODT of four Robots in a RoboCup Match

This section evaluates the performance of the CIIL and CODT algorithms in real match situations. The CIIL and the CODT algorithms have successfully been used throughout several RoboCup events since 1999 and 2001, respectively. This includes tournaments like the RoboCup World Championship, German Open, and a series of demonstration and friendly matches. During all events, log files containing the states of the robots were recorded. This section investigates the log files of two events, i.e. the RoboCup 2001 world championship and the open day which was held at the Munich University of Technology in 2002.

During the RoboCup 2001 world championship *The AGILO RoboCuppers* played six matches with a total time of two hours and four minutes, scored 22 : 6 goals and advanced to the quarter finals. The RoboCup field was approximately 5 * 10 m large and was surrounded by a white wall, such that the robots could not escape from the pitch and were not disturbed by noise or unexpected observations from an undefined exterior. At that time the robots were equipped with an industrial PC containing a Pentium 200 MMX CPU, 64 MB of RAM and an off-the-shelf framegrabber expansion card. In this configuration the robots were able to process a mean of 13 frames per second, i.e. the CIIL and CODT algorithms were both run at a frequency of 13 Hz.

At the 2002 open day *The AGILO RoboCuppers* played three matches with a total time of one hour and 50 minutes and achieved a total score of 11 : 8 goals. The RoboCup field was approximately 6 * 12 m large. The surrounding walls had completely been removed and were replaced by a 12.5 cm wide line surrounding the entire field. As a consequence, the robots

	Dist.	Own	$\bar{v}$	v_{min}	v_{max}	$\bar{\omega}$	ω_{min}	ω_{max}	Loc.	Err.	RMSE
Robot	(m)	(%)		(m/s)			(deg/s)		(#)	(%)	(m)
RoboCup 2001 in Seattle, Washington, USA											
6 Matches, Total Score 22 : 6, Total Time: 02:04:38.418 (7478.418 sec), CIIL and CODT with 13 fps											
Break - Total Time: 00:30:34.082 (1834.082 sec)											
#1	63.05	99.71	0.15	-1.22	0.74	15.77	-150.58	119.28	18	(0.02)	-
#2	188.26	88.87	0.34	-1.44	1.42	33.54	-390.97	244.47	58	(0.06)	-
#3	175.93	95.07	0.32	-1.46	1.41	31.94	-425.90	208.18	27	(0.03)	-
#4	182.90	99.40	0.36	-1.44	1.40	30.70	-171.44	228.14	30	(0.03)	-
Match - Total Time: 01:34:04.336 (5644.336 sec)											
#1	668.43	99.83	0.15	-0.96	1.21	7.58	-117.47	183.69	18	(0.02)	-
#2	1157.08	76.34	0.20	-0.90	1.02	33.34	-160.56	221.34	92	(0.09)	-
#3	1144.68	83.92	0.20	-0.91	1.02	34.72	-165.09	177.79	63	(0.06)	-
#4	1024.65	78.87	0.21	-1.08	1.02	33.65	-276.22	166.91	75	(0.09)	-
Friendly matches at the open day 2002 of the Munich University of Technology, Bavaria, Germany											
3 Matches, Total Score 11 : 8, Total Time: 01:50:41.352 (6641.352 sec), CIIL and CODT with 30 fps											
Break - Total Time: 00:41:15.392 (2475.392 sec)											
#1	123.45	93.73	0.20	-0.98	0.81	9.78	-210.45	134.25	84	(0.04)	0.14
#2	385.04	70.21	0.17	-1.15	1.39	27.18	-282.57	267.15	1204	(0.62)	0.37
#3	366.62	66.29	0.20	-0.88	1.18	35.65	-194.12	190.04	583	(0.30)	0.26
#4	360.40	84.15	0.26	-0.96	1.42	35.43	-209.09	197.30	497	(0.25)	0.22
Match - Total Time: 01:09:25.960 (4165.96 sec)											
#1	424.26	99.47	0.19	-0.98	0.96	7.71	-144.23	189.59	2	(0.00)	0.12
#2	1024.60	77.37	0.11	-0.45	0.93	29.90	-149.22	152.85	2292	(1.15)	0.34
#3	1172.88	67.65	0.17	-0.72	0.95	38.33	-234.49	170.08	1555	(0.78)	0.23
#4	1087.33	65.22	0.18	-0.48	0.97	38.03	-302.52	201.83	921	(0.46)	0.19

Table 7.3: Match statistics of two RoboCup events.

vision system had to be adapted and made more robust towards noise and false perceptions. In contrast to 2001 the hardware had undergone major changes. The PC has been replaced by a laptop computer with a Pentium III 900 MHz CPU, 256 MB of RAM and an IEEE 1394 Cardbus interface. In this configuration the robots were able to process all 30 frames per second that were delivered by the camera, i.e. the CIIL and CODT algorithms were both run at a frequency of 30 Hz.

Robot #1 (Theodo) is the goal keeper. His job is to protect the goal. In comparison to his teammates this robot is quite stationary as he usually does not leave the penalty area.

Robot #2 (Grimoald), robot #3 (Hugibert) and robot #4 (Odilo) are the field players of the team. Role and action selection is performed dynamically and completely autonomously by each robot (see (Buck, Beetz, and Schmitt, 2001; Buck, Schmitt, and Beetz, 2002) for further details). At any one time, two robots choose the role as defender and one robot takes the role as striker. At the game start Hugibert's and Grimoald's starting positions are close to the left and right corner of the penalty area. Odilo's starting position is either near the centre circle or at the kick-off point. Due to these starting positions Odilo finds itself more often in the role of the striker, than Hugibert or Grimoald, and as a consequence Odilo travels longer distances across the field than his teammates.

Table 7.3 characterises both sets of log files by providing some fundamental statistics. For each set of log files the statistics are subdivided into the two main stages of a match, *break* and *match*. Break statistics are collected during periods of time where the referee has stopped the match due to a foul or a goal. Their figures may be a result of manual human interference and interaction with the robots, such as a human entering the field and pushing a robot back to its starting position. A running match is characterised by the match statistics. These are collected during a period of time with complete autonomy of the robots and with almost no human interference.

The first column of Table 7.3 contains the robot number. The second column displays the total distance a robot has travelled during the match and break intervals of all log files. The percentage of time spent in the own half (Own %) is displayed in column three. Column four and five contain the mean, minimum and maximum translational (v) and rotational (ω) velocity of the robots, respectively. The localisation accuracy is assessed in column six. It contains the number of images the CIIL algorithm did not converge for and had to be reinitialised and the mean localisation accuracy. Unfortunately the mean localisation accuracy can only be assessed for the second set of log files, as the ground truth camera system was not available in 2001.

7.6.1 Self-Localisation

This section investigates the capabilities of the CIIL algorithm during real match situations. It shows that the accuracy of the self-localisation in real match situations is in the same order of magnitude as during the experiments performed in Section 7.4. Furthermore it is demonstrated that the self-localisation is quite robust towards a set of wrong intrinsic camera parameters and noise inflicted by a wrong colour classifier.

The analysis of the statistics reveals that the CIIL algorithm worked very well during RoboCup 2001, while the field was surrounded by a white wall. This is not amazing, since the walls are visible all across the field and can be used for localisation from almost any position on the field. The removal of the wall in 2002 made the CIIL algorithm fail more often by one and two orders of magnitude. This result was expected, since the field lines that replaced the walls are properly visible only from within a distance of up to two meters. Consequently, they cannot be used for localisation as often as the walls were used in previous years. Furthermore, the spectators surrounding the field introduce more noise and can lead to the establishment of false correspondences. Nevertheless, the localisation worked very well for the goalkeeper (#1) and Odilo (#4). Their mean localisation accuracies are estimated to be 12 and 19 cm, respectively. This is not astonishing for the goalkeeper, since it is quite

	Self-Localisation		Ball Observations		Opponent Observations	
Robot	RMSE (m)	std.dev. (m)	RMSE (m)	std.dev. (m)	RMSE (m)	std.dev. (m)
1. Match (break / **match**)						
#1	0.14 / **0.12**	0.09 / **0.07**	0.24 / **0.31**	0.25 / **0.23**	0.35 / **0.38**	0.27 / **0.26**
#2	0.31 / **0.33**	0.13 / **0.15**	0.46 / **0.38**	0.39 / **0.25**	0.52 / **0.50**	0.29 / **0.26**
#3	0.30 / **0.24**	0.12 / **0.11**	0.36 / **0.24**	0.24 / **0.22**	0.58 / **0.46**	0.25 / **0.26**
#4	0.21 / **0.19**	0.06 / **0.09**	0.43 / **0.25**	0.30 / **0.23**	0.41 / **0.37**	0.27 / **0.25**
2. Match (break / **match**)						
#1	0.15 / **0.11**	0.11 / **0.06**	0.28 / **0.33**	0.20 / **0.22**	0.37 / **0.42**	0.23 / **0.26**
#2	0.44 / **0.33**	0.29 / **0.19**	0.49 / **0.35**	0.21 / **0.25**	0.56 / **0.48**	0.24 / **0.27**
#3	0.25 / **0.21**	0.11 / **0.10**	0.44 / **0.28**	0.24 / **0.24**	0.54 / **0.40**	0.26 / **0.24**
#4	0.25 / **0.20**	0.08 / **0.10**	0.34 / **0.25**	0.21 / **0.26**	0.36 / **0.35**	0.23 / **0.24**
3. Match (break / **match**)						
#1	0.13 / **0.12**	0.10 / **0.09**	0.23 / **0.27**	0.19 / **0.22**	0.41 / **0.40**	0.25 / **0.26**
#2	0.36 / **0.37**	0.21 / **0.21**	0.45 / **0.34**	0.26 / **0.26**	0.56 / **0.51**	0.27 / **0.26**
#3	0.24 / **0.23**	0.11 / **0.11**	0.44 / **0.26**	0.18 / **0.22**	0.50 / **0.44**	0.25 / **0.25**
#4	0.21 / **0.19**	0.10 / **0.10**	0.28 / **0.18**	0.21 / **0.20**	0.42 / **0.38**	0.23 / **0.25**
Mean of all matches (break / **match**)						
#1	0.14 / **0.12**	0.10 / **0.08**	0.25 / **0.29**	0.21 / **0.22**	0.38 / **0.40**	0.25 / **0.26**
#2	0.37 / **0.34**	0.21 / **0.18**	0.46 / **0.36**	0.32 / **0.25**	0.54 / **0.50**	0.27 / **0.26**
#3	0.26 / **0.23**	0.11 / **0.11**	0.43 / **0.26**	0.23 / **0.23**	0.55 / **0.44**	0.26 / **0.25**
#4	0.22 / **0.19**	0.08 / **0.10**	0.33 / **0.21**	0.25 / **0.22**	0.40 / **0.37**	0.25 / **0.25**

Table 7.4: Accuracies achieved for self-, ball-, and opponent localisation.

stationary and can observe the penalty area lines most of the time very well and use them for precise localisation. The accuracy achieved by Odilo is quite remarkable since it travelled long distances across the field and scored several goals. The inferior accuracies of Grimoald (#2) and Hugibert (#3) lead to further investigations and it was found, that both robots were using a set of wrong camera parameters. Furthermore, Grimoald (#2) was equipped with a suboptimal colour lookup table, and as such failed to produce good classification results for a wide range of images. As a consequence, the CIIL algorithm failed more often and the achieved accuracy was less good than for the other two robots. However, the achieved accuracies are still quite good and prove that the CIIL algorithm is robust up to a certain

	Ball track				Opponent tracks			
	Corr.	Incorr.	RMSE	std. dev.	Corr.	Incorr.	RMSE	std. dev.
Robot	(%)	(%)	(m)	(m)	(%)	(%)	(m)	(m)
1. Match (break / **match**)								
#1	7 / **21**	9 / **8**	0.28 / **0.32**	0.28 / **0.23**	24 / **25**	14 / **13**	0.50 / **0.43**	0.30 / **0.26**
#2	47 / **17**	22 / **11**	0.47 / **0.37**	0.40 / **0.26**	32 / **21**	13 / **20**	0.54 / **0.50**	0.29 / **0.24**
#3	5 / **18**	23 / **14**	0.32 / **0.23**	0.19 / **0.23**	27 / **27**	19 / **19**	0.55 / **0.45**	0.26 / **0.25**
#4	28 / **31**	15 / **9**	0.46 / **0.27**	0.32 / **0.24**	29 / **34**	34 / **13**	0.47 / **0.38**	0.28 / **0.24**
Coop.	38 / **46**	17 / **10**	0.21 / **0.23**	0.25 / **0.21**	46 / **57**	12 / **10**	0.41 / **0.35**	0.30 / **0.19**
2. Match (break / **match**)								
#1	13 / **12**	12 / **7**	0.31 / **0.33**	0.22 / **0.21**	23 / **20**	13 / **8**	0.42 / **0.46**	0.26 / **0.25**
#2	32 / **16**	19 / **15**	0.49 / **0.31**	0.22 / **0.25**	18 / **19**	14 / **12**	0.60 / **0.46**	0.22 / **0.27**
#3	50 / **20**	23 / **8**	0.46 / **0.32**	0.24 / **0.26**	34 / **28**	15 / **16**	0.52 / **0.40**	0.25 / **0.24**
#4	21 / **28**	12 / **13**	0.35 / **0.29**	0.21 / **0.27**	30 / **31**	14 / **11**	0.43 / **0.36**	0.24 / **0.23**
Coop.	49 / **49**	14 / **13**	0.40 / **0.23**	0.25 / **0.22**	42 / **50**	11 / **9**	0.39 / **0.34**	0.28 / **0.21**
3. Match (break / **match**)								
#1	20 / **24**	12 / **9**	0.23 / **0.28**	0.19 / **0.22**	19 / **20**	9 / **11**	0.47 / **0.46**	0.26 / **0.25**
#2	27 / **10**	39 / **9**	0.44 / **0.37**	0.24 / **0.26**	11 / **16**	11 / **11**	0.56 / **0.50**	0.25 / **0.25**
#3	26 / **18**	29 / **12**	0.45 / **0.27**	0.17 / **0.22**	18 / **21**	16 / **13**	0.50 / **0.44**	0.24 / **0.25**
#4	67 / **34**	5 / **17**	0.26 / **0.22**	0.20 / **0.22**	38 / **24**	6 / **9**	0.41 / **0.39**	0.23 / **0.25**
Coop.	53 / **57**	12 / **16**	0.26 / **0.19**	0.21 / **0.18**	32 / **46**	7 / **8**	0.37 / **0.38**	0.24 / **0.21**
Mean of all matches (break / **match**)								
Coop.	46 / **51**	14 / **13**	0.29 / **0.21**	0.23 / **0.20**	40 / **51**	10 / **9**	0.39 / **0.35**	0.27 / **0.20**

Table 7.5: Accuracies and coverage achieved by the CODT algorithm for ball and opponent tracking.

degree of noise and the use of suboptimal camera parameters. The addition of a suboptimal colour classifier causes the CIIL algorithm to be unstable and fail more often by two orders of magnitude.

7.6.2 Cooperative Object Detection and Tracking

This section evaluates the capabilities of the CODT algorithm. It demonstrates that cooperative state estimation increases both, the accuracy and the completeness of objects tracks.

A detailed analysis of the CODT algorithm is only performed for the 2002 matches. Table 7.4 summarises the input data used to test the CODT algorithm. This data was acquired during three matches. The left, centre and right column display the accuracies (RMSE) and the standard deviation of the self-localisation, of the ball observations and of the opponent observations of the individual robots. Every entry consists of an appropriate value for break (left) and match periods (right). During a break the match is intermitted and the human team members enter the field and repair robots and drag them to their starting position, etc. This additional noise explains why the accuracies for all three kinds of observations are worse during the break than during the match. It is clearly obvious that self-localisation errors have an impact on errors for ball and opponent observations. As a rule the errors for opponent observations are usually greater than the errors for ball observations. This is due to the unique circular shape of a ball. Arbitrary robot shapes hamper the opponent detection routines and as such add an indirect level of noise. Unfortunately the influence of the wrong intrinsic camera parameters of Grimoald and Hugibert on the observations is clearly visible.

The results of the CODT algorithm for all three matches are displayed in Table 7.5. The column ball and opponent tracks reveal the achieved statistics for each object class. Every column displays the percentage of correct tracks, the percentage of incorrect tracks, the track accuracies, and the standard deviation. Again, every entry consists of an appropriate value for break (left) and match periods (right). For every match, the statistics for individual as well as for cooperative perceptions are given. In order to generate the statistics for the individual observations the CODT was run four times only using the observations performed by one robot.

Cooperative perception increases both the percentage of the correctly determined tracks and the accuracy of these tracks. During the match, between 46 to 57 % of the ball's trajectory was detected with an accuracy of 0.19 to 0.23 m. This is a good result, since the ball is often occluded by robots, lifted up by the referee and moved to a new location or shot off the pitch by one of the robots. Opponent tracking worked equivalently well. In average, 51 % of the opponent tracks were determined correctly by the CODT algorithm. The number of false tracks was reduced to an average of 9 % and the mean track accuracy was 0.35 m. This is also a good result, since broken robots are regularly moved off the field and repaired outside. Furthermore, the opponent goal keeper is usually only observed during an attack.

7.6.3 Detailed Analysis of Match Scenes

This section performs a detailed analysis of scenes from two matches played in 2001 and 2002. It demonstrates how observations from different teammates are integrated into tracks. Though the robots perform more false positive observations in 2002 than in 2001, after the removal of the wall, the quality of the trajectories is only slightly decreased.

Scene from a match in 2001 with walls surrounding the field

Figure 7.10 depicts a two minute scene from the second half of a match against *The Ulm Sparrows* in 2001. It shows the sixth goal of *The AGILO RoboCuppers*. The final score of this match was 7:0. Figures 7.10(a) and (b) show the trajectories of the AGILO robots. Ball and opponent observations performed by the AGILO players are displayed in Figures 7.10(c,d) and (e,f), respectively. The tracks generated by the CODT algorithm are presented in Figures 7.10(g,h). Black tracks indicate the trajectories of opponent robots and red tracks depict the trajectory of the ball. It is interesting to note how the observations performed by different robots are integrated into the tracks. Several false observations are correctly identified by the CODT algorithm and are discarded. The accumulation of two opponent robots and one AGILO player at the bottom of Figure 7.10(g) was successfully resolved. Figure 7.10(g) reveals relatively long tracks. The track on the left comes from the same opponent player. It is interrupted at the top left, since the opponent player was not observed by any AGILO player for about 10 s. The tracks on the right are originated from two other opponent players. It is particularly interesting to note that both tracks were successfully handed over from one observing AGILO robot to another. The ball was observed most of the time and its trajectory was resolved correctly. All tracks were qualitatively verified with a video taken from the match.

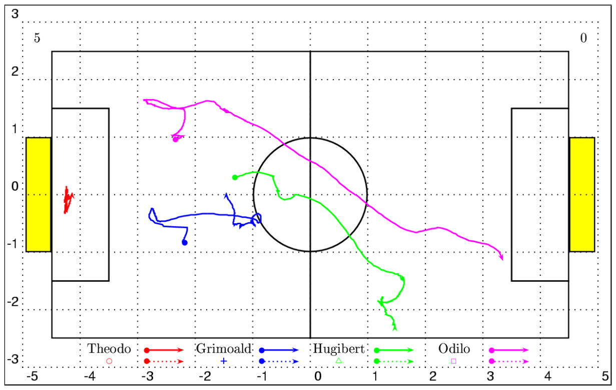
Scene from a match in 2002 without walls surrounding the field

Figure 7.11 depicts the last two minutes of the second match against *The Ulm Sparrows* in 2002. It shows the final attack of *The AGILO RoboCuppers*. Odilo tries to score but unfortunately he looses control of the ball in the last seconds and the match ends with a final score of 4:3. Figure 7.11(a) shows the trajectories of the AGILO robots. Ball and opponent observations performed by the AGILO players are displayed in Figures 7.11(b) and (c), respectively. The tracks generated by the CODT algorithm are presented in Figures 7.11(d). Black tracks indicate the trajectories of opponent robots and red tracks depict the trajectory of the ball. Figures 7.11(e) to (h) depict the same match scene without cooperative perception only based on the perceptions of the individual robots. Figure 7.11(e) is only based on Theodo's perceptions, Figure 7.11(f) on Grimoald's, Figure 7.11(g) on Hugibert's and Figure 7.11(h) on Odilo's ball and opponent observations. This example demonstrates clearly that the game states, estimated on the basis of the individual perceptions only, are erroneous and less complete. Theodo, the goal keeper hallucinates some tracks in the opponent's half and the field players only observe small parts of the opponent and ball trajectories. This is particularly obstructive since the coordination of the team and the behaviour of the individual robots is based on the ball's position.

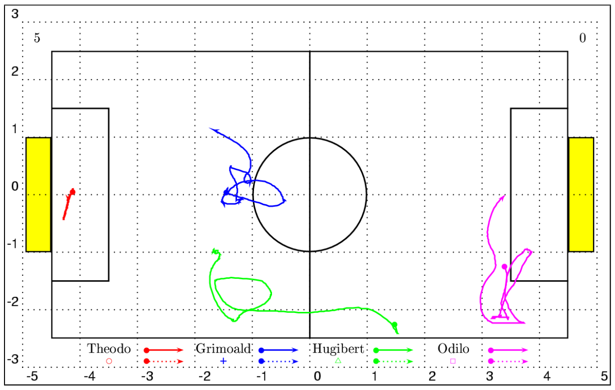
7.7 Conclusions

This chapter has investigated the capabilities of the CIIL and CODT algorithms in a variety of experiments, including several matches performed under ordinary RoboCup match conditions. Both algorithms can run at frame rate and achieve estimates with good accuracy. This was proven through groundtruth data provided by a ceiling camera system.

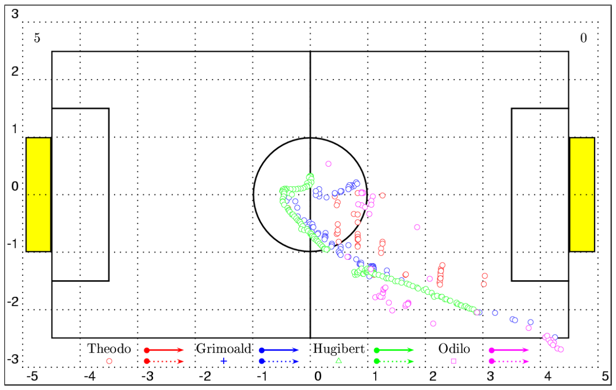
It was demonstrated that cooperative state estimation enables robots to determine their poses and the positions of further dynamic objects more accurately, to integrate individual observations into one common view of the world, to track temporarily occluded objects successfully, and to obtain a more complete view of the surrounding environment.



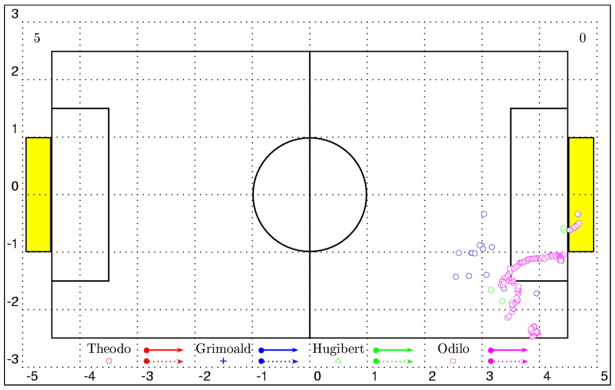
(a)



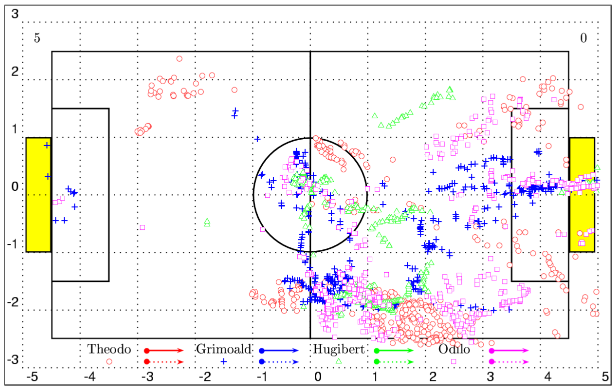
(b)



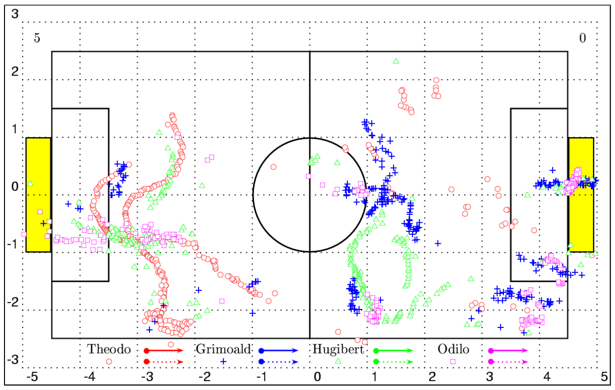
(c)



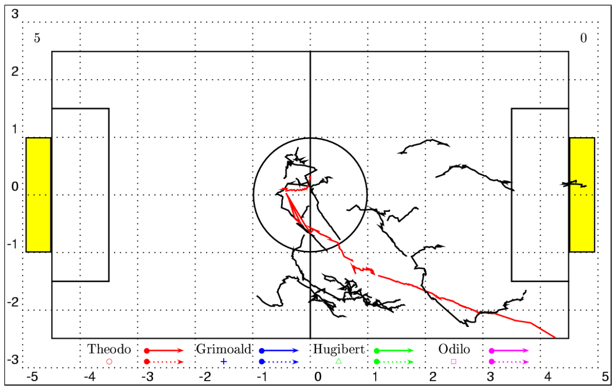
(d)



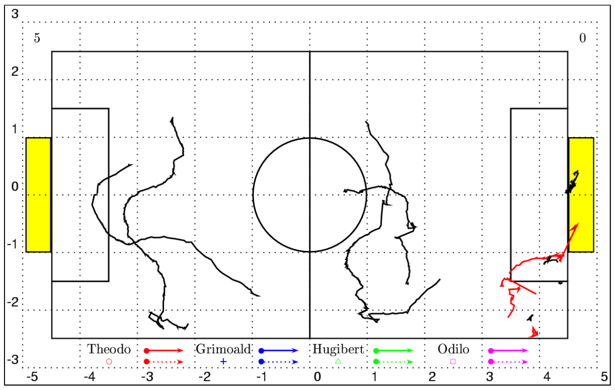
(e)



(f)

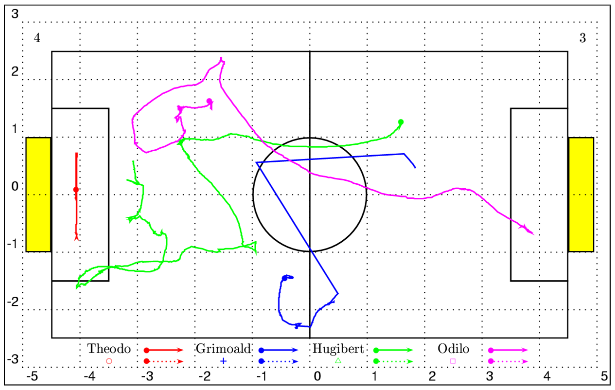


(g)

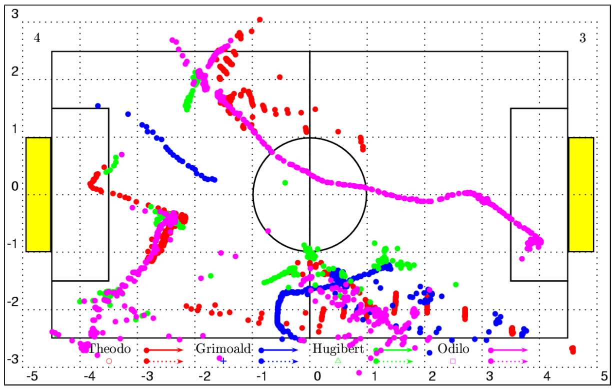


(h)

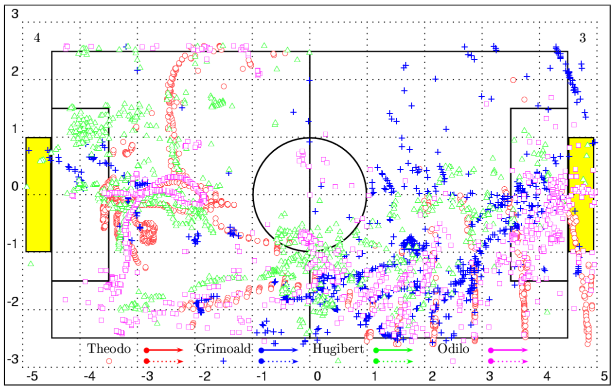
Figure 7.10: (a,b) Trajectories of *The AGILO RoboCuppers*, (c,d) ball observations of *The AGILO RoboCuppers*, (d,e) opponent observations of *The AGILO RoboCuppers*, (f,g) estimated trajectories of the ball and *The Ulm Sparrows*.



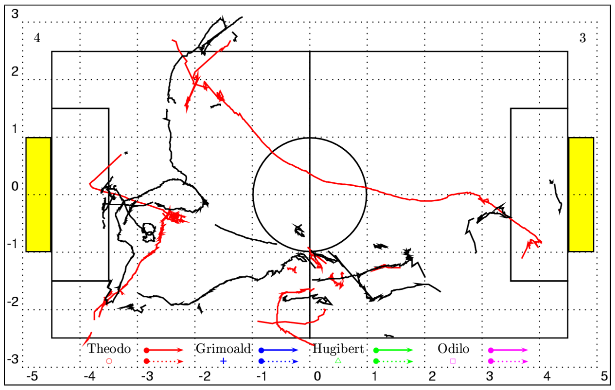
(a)



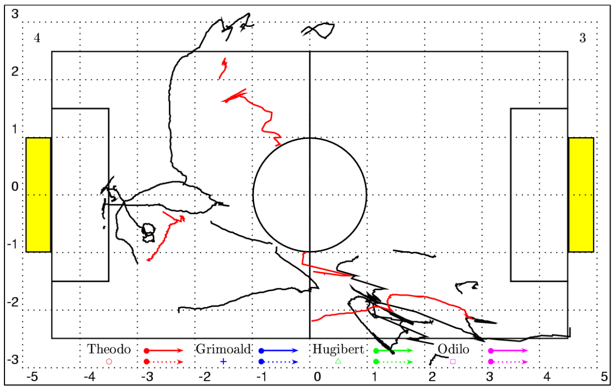
(b)



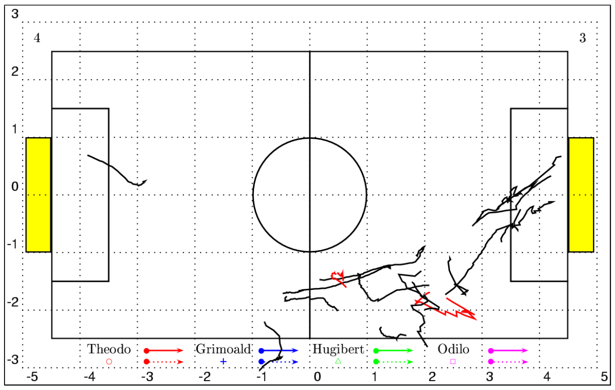
(c)



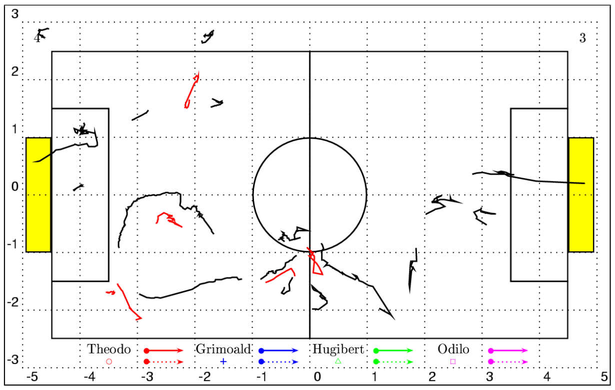
(d)



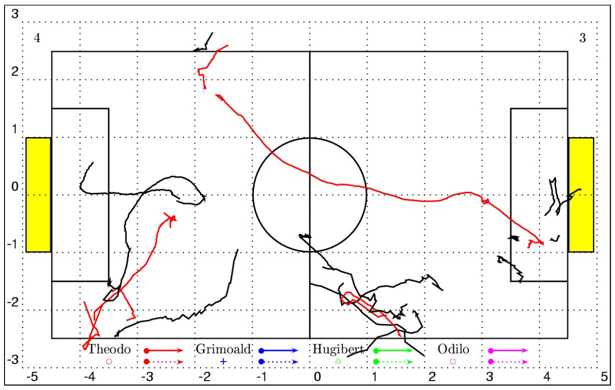
(e)



(f)



(g)



(h)

Figure 7.11: (a) Trajectories of *The AGILO RoboCuppers*, (b) ball observations of *The AGILO RoboCuppers*, (c) opponent observations of *The AGILO RoboCuppers*, (d) cooperatively estimated trajectories of the ball and *The Ulm Sparrows*, (e)-(f) individually estimated trajectories.

Chapter 8

Conclusions

In this thesis the computational problem of perceiving states of complex dynamic environments with a set of mobile, vision-based, and cooperating sensor systems has been investigated. The results of this investigation are a suite of novel and powerful techniques and an integrated system that can perform this perception task both accurately and reliably. To account for restricted views of individual sensors, the unreliability and inaccuracy of the data provided by the sensors, and the uncertainty about the evolution of the environment the problem has been solved as a complex probabilistic state estimation problem.

To develop and evaluate these techniques autonomous robot soccer was chosen as primary application scenario. Robot soccer provides a challenging and realistic testbed for cooperative state estimation in complex and dynamically changing environments. These challenges include: (1) a competitive, highly dynamic, and fast changing environment, (2) a changing number of opponent robots with unknown identity, (3) the use of an inaccurate and noisy vision sensors and (4) independently moving sensors with inaccurately known positions.

Thus, the robots of *The AGILO RoboCuppers* were equipped with standard video cameras, some limited communication capabilities, and with the developed state estimation system to provide them with accurate and reliable information about the situation on the soccer field. The state estimation system including the techniques that it has applied were thoroughly tested and evaluated in the course of four robot soccer world championships. The presented results showed, that (1) purely image-based probabilistic estimation of complex game states is feasible in real time even in complex and fast changing environments, and that (2) cooperative perception increases both the accuracy and the completeness of the estimated states.

The research described in this thesis makes three key scientific and technical contributions to the fields of vision-based perception and autonomous robot control:

1. *Estimation of Large Scale and Complex States* - The approach to state estimation presented in this thesis is able to solve large scale and complex state estimation problems in the robot soccer domain. The estimated state includes, the pose of the own robots, the position of the ball, and the positions of the opponent players. The overall state consists of a vector with more than 60 parameters (see Chapter 1). In order to reduce the overall complexity of the estimation problem, it is decomposed into several loosely coupled subproblems. Each subproblem is solved by a task specific state estimator employing a specific state estimation technique (see Chapters 3 and 4). The state estimators interact in order to derive more accurate and complete game state estimates. This is

made possible by a compact and uniform representation of uncertainties, consisting of a mean value and a covariance matrix, that can concisely be communicated between and easily be processed by the state estimation algorithms. This approach allows a team of robots to make use of all available information and to perform vision-based self-localisation, to track the ball, and to track the opponent robots at a frame rate of 30 Hz with off-the-shelf computing hardware and sensory equipment.

2. *Vision-based Self-Localisation, Ball- and, Opponent-Tracking* - A fast and accurate algorithm for model driven vision-based self- and ball-localisation is presented (see Chapter 5). The proposed algorithm is capable of estimating the pose of a robot in 3D (2D position and 1D orientation) or 6D (3D position and 3D orientation) and can be employed on various robotic platforms. Fast response times are achieved through the use of a known environment model consisting of curve features, an accurate and universal model of an image sensor, and an extremely efficient feature projection and extraction process. Per image less than 10% of the pixels are accessed and mean position and orientation accuracies of less than 10 cm and 3^o are achieved, respectively. A particular feature of this algorithm is its capability to uniquely solve ambiguous localisation problems through the use of teammate observations of dynamic objects with known shapes. Furthermore, an efficient algorithm for vision-based object detection and cooperative tracking is presented (see Chapter 6). This algorithm is especially suited for mobile sensors with uncertain position. The observations of dynamic objects with unknown shape performed by all members of the robot team are integrated into one single view of the world. In particular, it is demonstrated that this enables a team to track temporarily occluded dynamic objects and to enlarge the common field of view.

3. *Cooperative Perception Enhances the State Estimates* - It is demonstrated that cooperative perception enhances the accuracy, completeness and reliability of the estimated states. This evidence is derived from a series of real world experiments, with a physical team of robots in real match situations. Using ground truth data, provided by an overhead camera system it is shown that cooperative state estimation increases the accuracy as well as the coverage of ball and opponent tracks both substantially and significantly (see Chapter 7). The mean accuracies for cooperative ball and opponent tracking are 0.21 and 0.35 m, respectively. Cooperative ball and opponent tracking recovered more the 50 % of the balls and all opponent tracks with the standard control programme for RoboCup robots. It is expected that active observation routines would further enhance the achieved results.

Bibliography

Adorni, G., S. Cagnoni, S. Enderle, G.K. Kraetzschmar, M. Mordonini, M. Plagge, M. Ritter, S. Sablatnög, and A. Zell (2001). Vision-based localization for mobile robots. *Robotics and Autonomous Systems* 0(36): 103–119. 82

Akashi, H. and H. Kumamoto (1977). Random sampling approach to state estimation in switching environments. *Automatica* 13: 429–434. 46

Aström, K. J. (1965). Optimal control of markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications* 10: 174–205. 9, 10, 33, 34, 55, 61

Baker, S. and S. Nayar (1998a). A theory of catadioptric image formation. In *9th International Conference on Computer Vision (ICCV)*, pp. 35–58. IEEE Computer Society Press. 81

Baker, S. and S. Nayar (1998b). A tutorial on catadioptirc image formation. http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/BAKER/main.html. 81

Bandlow, T., R. Hanek, M. Klupsch, and T. Schmitt (1999a). AGILO RoboCuppers: RoboCup Team Description. In *Third International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 55

Bandlow, T., M. Klupsch, R. Hanek, and T. Schmitt (1999b). Fast image segmentation, object recognition and localization in a RoboCup scenario. In *Third International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 60

Bar-Shalom, Y. and T. Fortmann (1988). Tracking and data association. Academic Press. 19, 40, 45, 70, 88, 96, 97, 101

Beetz, M. (2001). Structured Reactive Controllers. *Journal of Autonomous Agents and Multi-Agent Systems* 4: 25–55. 63

Beetz, M., S. Buck, R. Hanek, A. Hofhauser, and T. Schmitt (2002a). AGILO RoboCuppers 2002: Applying cooperative game state estimation experience-based learning, and plan-based control to autonomous robot soccer. In Stone, P., T. Balch, and G. Kraetzschmar, editors, *6th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 55

Beetz, M., S. Buck, R. Hanek, T. Schmitt, and B. Radig (2002b). The AGILO autonomous robot soccer team: Computational principles, experiences, and perspectives. In *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pp. 805–812, Bologna, Italy. 55

Beetz, M., T. Schmitt, R. Hanek, S. Buck, F. Stulp, D. Schröter, and B. Radig (2004). The agilo robot soccer team — experience-based learning and probabilistic reasoning in autonomous robot control. *Autonomous Robots Journal* 2(2): –. 55, 60

Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distributions. In *Bull.*, number 35 in Calcutta Math. Soc., pp. 99–109. 20

Blake, A. and M. Isard (1998). *Active Contours*. Springer-Verlag, Berlin Heidelberg New York. 46, 73, 103

Bonarini, A., G. Kraetzschmar, P. Lima, T. Nakamura, F. Ribeiro, and T. Schmitt (2003). Robocup 2003 - middle size robot league – rules and regulations. http://wwwradig.in.tum.de/MSL-2003/rules2003/. 105

Buck, S., M. Beetz, and T. Schmitt (2001). Planning and Executing Joint Navigation Tasks in Autonomous Robot Soccer. In *RoboCup 2001: Robot Soccer World Cup V, Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag. 61, 122

Buck, S., M. Beetz, and T. Schmitt (2002a). Approximating the Value Function for Continuous Space Reinforcement Learning in Robot Control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2002*, Lausanne, pp. 1062–1067. 62

Buck, S., M. Beetz, and T. Schmitt (2002b). M-ROSE: A Multi Robot Simulation Environment for Learning Cooperative Behavior. In *Distributed Autonomous Robotic Systems 5, Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag. 61, 62

Buck, S., R. Hanek, M. Klupsch, and T. Schmitt (2000). AGILO RoboCuppers: RoboCup Team Description. In Stone, P., T. Balch, and G. Kraetzschmar, editors, *4th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 55

Buck, S., T. Schmitt, and M. Beetz (2002). Reliable Multi Robot Coordination Using Minimal Communication and Neural Prediction. In *Advances in Plan-based Control of Autonomous Robots. Selected Contributions of the Dagstuhl Seminar Plan-based Control of Robotic Agents, Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag. 61, 62, 122

Buck, S., F. Stulp, M. Beetz, and T. Schmitt (2002). Machine Control Using Radial Basis Value Functions and Inverse State Projection. In *IEEE International Conference on Robotics, Control, Automation, and Vision (ICARVC) 2002*, Singapore. 62

Buck, S., U. Weber, M. Beetz, and T. Schmitt (2001). Multi robot path planning for dynamic environments: A case study. In *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems*, pp. 1245–1250. IEEE/RSJ. 62

Burgard, W., A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun (1998). The interactive museum tour-guide robot. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98).* 83

Burgard, W., A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun (2000). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence* 114(1-2). 9

Burgard, W., A. Derr, D. Fox, and A. Cremers (1998). Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In *Proc. of the IEEE/RSJ International Conference ond Intelligent Robots and Systems (IROS'98).* 83

Burgard, W., D. Fox, D. Henning, and T. Schmidt (1996). Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'96)*, pp. 896–901, Portland, Oregon, USA. 45, 83

Burkhard, H.-D., I. Dahm, S. Deutsch, M. Hebbel, M. Osterhues, A. Jüngel, J. Hoffmann, M. Lötzsch, J. Bach, and T. Röfer (2003). GermanTeam 2003. http://www.robocup.de/germanteam/#publications. 118

Burkhard, H.-D., U. Düffert, M. Jüngel, M. Lötzsch, N. Koschmieder, T. Laue, T. Röfer, K. Spiess, A. Sztybryc, R. Brunn, M. Risler, and O. Stryk (2001). GermanTeam 2001. http://www.robocup.de/germanteam/#publications. 118

Clarke, J., S. Carlsson, and A. Zisserman (1996). Detecting and tracking linear features efficiently. In *British Machine Vision Conference (BMVC)*, pp. 415–424. 85

Comanicu, D., V. Ramesh, and P. Meer (2003). Kernel-based object tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 25(5): 564–575. 103

Cox, I.J. (1990). Blanche: Position estimation for an autonomous robot vehicle. In *(Cox and Wilfong, 1990)*, pp. 221–228. 78

Cox, I.J. (1991). Blanche – an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. on Robotics and Automation* 7(2): 193–204. 78, 82

Cox, I.J. (1993). A review of statistical data association techniques for motion correspondence. *Int. J. Computer Vision* 10(1): 53–65. 101

Cox, I.J. and S.L. Hingorani (1996). An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 18(2): 138–150. 103

Cox, I.J. and J. Leonard (1994). Modeling a dynamic environment using a bayesian multiple hypothesis approach. Artificial Intelligence, 66:311–344. 45, 95, 98, 103

Cox, I.J., J.M. Rehg, and S. Hingorani (1993). A bayesian multiple-hypothesis approach to edge grouping and contour segmentation. *Int. J. Computer Vision* 11(1): 5–24. 103

Cox, I.J. and G.T. Wilfong, editors (1990). *Autonomous robot vehicles.* Springer-Verlag. 133

Crisman, Z., E. Curre, C. Kwok, N. Ratliff, L. Tsybert, and D. Fox (2002). Uw huskies-02. In *7th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences).* 48

Crowley, J.L. (1989). World modeling and position estimation for a mobile robot using ultrasonic ranging. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 674–680. 37

Crowley, J.L., F. Wallner, and B. Schiele (1998). Position estimation using principal components of range data. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3121–3128, Leuven, Belgium. 37

Dean, T. and M. Wellmann (1991). *Planning and Control.* Morgan Kaufmann Publishers. 31

Dellaert, F., W. Burgard, D. Fox, and S. Thrun (1999a). Using the condensation algorithm for robust, vision-based mobile robot localization. In *Computer Vision and Pattern Recognition (CVPR).* 48

Dellaert, F., D. Fox, W. Burgard, and S. Thrun (1999b). Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1322–1328, Detroit, Michigan, USA. 48

Dietl, M., J.-S. Gutmann, and B. Nebel (2001). Cooperative sensing in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1706–1713, Maui, Hawaii. 100

Doucet, A. (1998). On sequential simulation-based methods for bayesian filtering. Technical Report CUED/F-INFENG/TR 310, Department of Engineering, Cambridge University. 46

Doucet, A, J.F.G. Freitas, and N.J. Gordon (2000). *Sequential Monte Carlo Medthods in Practice.* Springer-Verlag. 47

Doucet, A, N. Freitas, K. Murphy, and S. Russel (2000). *Roa-Blackwellised Particle Filtering for Dynamic Bayesian Networks.* Morgan Kaufmann. 102

Faugeras, O.D. (1993). Three-dimensional computer vision: A geometric viewpoint. *MIT Press* p. 302. 75

Fischler, M. A. and R. C. Bolles (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, Vol. 24, pp. 381–395. 85

Forsberg, J., U. Larsson, P. Ahman, and A. Wernersson (1993). Navigation in cluttered rooms using a range measuring laser and the hough transform. In *International Conference on Intelligent Autonomous Systems*, pp. 248–257. 37

Fox, D. (2001). Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems*. MIT Press. 48, 49

Fox, D. (2003). Adapting the sample size in particle filters through kld-sampling. *Int. J. of Robotics Research (IJRR)* . 48, 84

Fox, D., W. Burgard, F. Dellaert, and S. Thrun (1999). Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Orlando, FL. 48

Fox, D., W. Burgard, H. Kruppa, and S. Thrun (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots* 8(3): 325–344. 84

Fox, D., W. Burgard, S. Thrun, , and A.B. Cremers (1998). Position estimation for mobile robots in dynamic environments. In *AAAI National Conference on Artificial Intelligence*. 11

Fox, D., W. Burgard, and S. Thrun (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research* 11: 391–427. 45

Fox, D., W. Burgard, S. Thrun, and A.B. Cremers (1998). Position estimation for mobile robots in dynamic environments. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, Madison, Wisconsin. 83

Fox, D., S. Thrun, W. Burgard, and F. Dellaert (2000). Particle filters for mobile robot localization. In Doucet, A., N. Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. 33

Gordon, N.J., D.J. Salmond, and A.F.M. Smith (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings-F* 140(2): 107–113. 46

Gutmann, J. and C. Schlegel (1996). Amos: Comparison of scan matching approaches for self-localization in indoor environments. 78, 79

Gutmann, J.-S. (2000). Robuste Navigation autonomer mobiler Systeme. Ph.D. diss., Albert-Ludwigs-Universitaet Freiburg. 23, 30, 79

Gutmann, J.-S (2002). Markov-kalman localization for mobile robots. In *International Conference on Pattern Recognition (ICPR)*, Quebec, Canada. 84, 100

Gutmann, J.-S., W. Burgard, D. Fox, and K. Konolige (1998). An experimental comparison of localization methods. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Victoria, Canada. 79

Gutmann, J.-S., W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, T. Weigel, and B. Welsch (1999). The CS Freiburg Robotic Soccer Team: Reliable Self-Localization, Multirobot Sensor Integration, and Basic Soccer Skills. In *Second International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 100

Gutmann, J.-S., T. Weigel, and B. Nebel (2001). A fast, accurate, and robust method for self-localization in polygonal environments using laser-range-finders. *Advanced Robotics Journal* 14(8): 651–668. 37, 79, 82

Handschin, J.E. and D.Q. Mayne (1969). Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control* 5(5): 547–559. 46

Hanek, R., T. Schmitt, M. Klupsch, and S. Buck (2000). From multiple images to a consistent view. In *4th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Vol. 4 of *Lecture Notes in Computer Science*. Springer-Verlag. 60

Hartley, R.I. and A. Zissermann (2000). *Multiple View Geometry in Computer Vision*. Cambridge University Press, UK. 85

Hue, C., J.-P. Le Cadre, and P. Perez (2000). Tracking multiple objects with particle filtering. Technical report 1361, IRISA. 102

Hue, C., J.-P. Le Cadre, and P. Perez (2001). The (mr)mtpf: particle filters to track multiple targets using multiple receivers. 102

Iocchi, L. and D. Nardi (1999). Self-Localization in the RoboCup Environment. In *Third International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 80

Isard, M. and A. Blake (1996a). Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision (ECCV)*, pp. 343–356, Cambridge, UK. 46

Isard, M. and A. Blake (1996b). Contour tracking by stochastic propagation of conditional density. In *Proc. European Conference on Computer Vision*, pp. I:343–356. 103

Isard, M. and A. Blake (1998). Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision* 29(1): 5–28. 103

Jazwinski, A.H. (1970). Stochastic processes and filtering theory. In *Vol. 64 of Mathematics in Science and Engineering*. Academic Press. 38

Jensfelt, P. (2001). Approaches to Mobile Robot Localization in Indoor Environments. Ph.D. diss., Signal, Sensors and Systems (S3), Royal Institute of Technology, SE-100 44 Stockholm, Sweden. 83

Jensfelt, P. and S. Kristensen (2001). Active global localisation for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation* 17(5): 748–760. 44, 83

Jonker, P., J. Caarls, and W. Bokhove (2000). Fast and Accurate Robot Vision for Vision based Motion. In Stone, P., T. Balch, and G. Kraetzschmar, editors, *4th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science, pp. 72–82. Springer-Verlag. 80

Julier, S. and J. Uhlmann (1997). A new extension of the kalman filter to nonlinear systems. The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls. 22, 41, 93

Kaelbling, L., A. Cassandra, and J. Kurien (1996). Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.* 83

Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering* 82(Series D): 35–45. 36

Kanazawa, K., D. Koller, and S. Russel (1995). Stochastic algorithms for dynamic probabilistic networks. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 346–351. Morgen Kaufmann. 46

Kitagawa, G. (1996). Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics* 5(1): 1–25. 46, 48

Klupsch, M. (1998). Object-Oriented Representation of Time-Varying Data Sequences in Multiagent Systems. In Callaos, N.C., editor, *International Conference on Information Systems Analysis and Synthesis (ISAS'98)*, pp. 833–839, Orlando, FL, USA. International Institute of Informatics and Systemics (IIIS). 58

Klupsch, M. (2001). Objektorienterte Daten- und Zeitmodelle für die Echtzeitbildfolgenauswertung. Ph.D. diss., Technische Universität München, Institut für Informatik IX. 58

Klupsch, M., T. Bandlow, M. Grimme, I. Kellerer, M. Lückenhaus, F. Schwarzer, and C. Zierl (1998). AGILO RoboCuppers: RoboCup Team Dscription. In *Second International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 55

Kristensen, S. and P. Jensfelt (2003). An experimental comparison of localisation methods, the mhl sessions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 83

Kumar, R. and A. R. Hanson (1994). Robust Methods for Estimating Pose and a Sensitivity Analysis. *CVGIP: Image Understanding* . 82

Kwok, C., D. Fox, and M. Meila (2003). Adaptive real-time particle filters for robot localization. In *IEEE International Conference on Robotics and Automation (ICRA).* 48, 49

Lanser, S. (1997). Modellbasierte Lokalisation gestützt auf monokulare Videobilder. Ph.D. diss., TU München, Fakultät für Informatik. 18, 23, 25, 30, 79

Lenser, S. and M. Veloso (2000). Sensor resetting localization for poorly modelled mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1225–1232. 48

Lenz, R. (1987). Linsenfehlerkorrigierte Eichung von Halbleiterkameras mit Standardobjektiven für hochgenaue 3D-Messungen in Echtzeit. In Paulus, E., editor, *Mustererkennung*, Informatik-Fachberichte 149, pp. 212–216. Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM), Springer-Verlag. 24

Lenz, R. and R. Y. Tsai (1986). Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3D Machine Vision Metrology. Technical Report RC 54867, IBM. 26

Lenz, R. and R. Y. Tsai (1988). Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 10(5): 713–720. 26

Leonard, J.J. and H.F. Durrant-Whyte (1991). Mobile robot localization by tracking geometric beacons. *IEEE Trans. on Robotics and Automation* 7(3): 376–382. 37

Leonard, J.J. and H.F. Durrant-Whyte (1992). *Directed Sonar Sensing for Mobile Robot Navigation.* Kluwer Academic Publisher, Boston. 37

Leonardis, Ale and Horst Bischof (2000). Robust recognition using eigenimages. *Computer Vision and Image Understanding: CVIU* 78(1): 99–118. 85

Lima, P., A. Bonarini, C. Machado, F. Marchese, C. Marques, F. Ribeiro, and D. Sorrenti (2001). Omni-directional catadioptric vision for soccer robots. *Special issue on the Euro-RoboCup of the Journal of Robotics and Autonomous Systems* 36(2): –. 82

Lowe, D. G. (1987). Three-Dimensional Object Recognition from Single Two-Dimensional Images. *Artificial Intelligence* 31: 355–395. 79

Lowe, D. G. (1991). Fitting Parameterized Three-Dimensional Models to Images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 13(5): 441–450. 41, 79

Lowe, D. G. (1992). Robust Model-based Motion Tracking Through the Integration of Search and Estimation. *Int. J. Computer Vision* 8(2): 113–122. 79

Lu, F. and E. Milios (1994). Robot pose estimation in unknown environments by matching 2d range scans. In *CVPR94*, pp. 935–938. 79

Lu, F. and E. Milios (1997). Globally consistent range scan alignment for environment mapping. 79

Marchese, F. and D.G. Sorrenti (2000). Omni-directional vision with a multi-part mirror. In *4th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Vol. 4 of *Lecture Notes in Computer Science*. Springer-Verlag. 81

Marques, C. and P. Lima (2000a). A localization method for a soccer robot using a vision-based omni directional sensor. In *4th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Vol. 4 of *Lecture Notes in Computer Science*. Springer-Verlag. 82

Marques, C. and P. Lima (2000b). Vision-Based Self-Localization for Soccer Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1193–1198, Takamatsu, Japan. 80, 82

McDermott, D. (1991). A reactive plan language. Research Report YALEU/DCS/RR-864, Yale University. 63

McLauchlan, P. and A. Jaenicke (2000). Image mosaicing using sequential bundle adjustment. In *British Machine Vision Conference (BMVC)*, pp. 616–626. 85

Melen, T. (1994). Geometrical Modelling and Calibration of Video Cameras for Underwater Navigation. Ph.D. diss., University of Trondheim, Norway, ITK-rapport 1994:103-W. 17

Nakamura, T., A. Ebina, M. Imai, T. Ogasawara, and H. Ishiguro (2000). Real-time Estimating Spatial Configurations between Multiple Robots by Triangle and Enumeration Constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 82

Nayar, S. (1997). Catadioptric omnidirectional camera. In *8th International Conference on Computer Vision (ICCV)*, pp. 482–488. IEEE Computer Society Press. 81

Neumann, D. (2003). Bayes pose estimation by modeling colour distributions. Masterthesis, Munich University of Technology, Department of Computer Science. 61, 68

Nourbakhsh, I., R. Powers, and S. Birchfield (1995). DERVISH an office-navigating robot. *AI Magazine* 16(2): 53–60. 83

Olson, C. (2000). Probabilistic self-localization for mobile robots. *IEEE Trans. on Robotics and Automation* 16(1): 55–66. 80

Plagge, M., R. Güther, J. Ihlenburg, D. Jung, and A. Zell (1999). The attempto robocup robot team. In *Third International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Vol. 1856 of *Lecture Notes in Artificial Intelligence*, pp. 424–433. Springer. 82

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1996). *Numerical Recipes in C*. Cambridge University Press, Cambridge. 70, 71, 75

Pritchett, P. and A. Zissermann (1998). Wide baseline stereo matching. In *International Conference on Computer Vision (ICCV)*, pp. 754–760. 85

Rasmussen, Christopher and Gregory D. Hager (2001). Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(6): 560–576. 103

Reid, D. (1979). An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control, 24(6):843–854. 43, 94, 95, 98

Rencken, W. (1994). Autonomous sonar navigation in indoor, unkown and unstructured environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 431–438. 37

Rosencrantz, M., G. Gordon, and S. Thrun (2003). Locating moving entities in indoor environments with teams of mobile robots. In *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, Melbourne, Australia. 101, 102

Rousseeuw, P.J. and A.M. Leroy (1987). *Robust Regression and Outlier Detection*. Wiley. 85

Rull, P. H. (1993). BARRY: An Autonomous Train-Spotter. In *Proc. Image and Vision Computing*, pp. 1–7. 103

Schaffalitzky, F. and A. Zisserman (2001). Viewpoint invariant texture matching and wide baseline stereo. In *8th International Conference on Computer Vision (ICCV)*, Cancouver, Canada. 85

Schmitt, T. and M. Beetz (2003). Designing probabilistic state estimators for autonomous robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3823–3829, Las Vegas, NV. 60

Schmitt, T., S. Buck, and M. Beetz (2001). AGILO RoboCuppers 2001: Utility- and plan-based action selection based on probabilistically estimated game situations. In Stone, P., T. Balch, and G. Kraetzschmar, editors, *5th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 55

Schmitt, T., R. Hanek, and M. Beetz (2003). Developing comprehensive state estimators for robot soccer. In *7th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Padova, Italy. 60

Schmitt, T., R. Hanek, M. Beetz, S. Buck, and B. Radig (2002). Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Trans. on Robotics and Automation* 18(5): 670–684. 55, 60

Schmitt, T., R. Hanek, S. Buck, and M. Beetz (2001). Cooperative probabilistic state estimation for vision-based autonomous soccer robots. In *5th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Seattle, USA. 60

Schmitt, T., R. Hanek, S. Buck, and M. Beetz (2002). Probabilistic vision-based opponent tracking in robot soccer. In *6th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Fukuoka, Japan. 60

Schulenburg, E., T. Weigel, and A. Kleiner (2003). Self-localization in dynamic environments based on laser and vision data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. –, Las Vegas, NV. 81, 82

Schulz, D., D. Burgard, D. Fox, and A.B. Cremers (2003). People tracking with a mobile robot using sample-based joint probabilistic data association filters. *Int. J. of Robotics Research (IJRR)* 22(2). 102

Schulz, D., W. Burgard, D. Fox, and A.B. Cremers (2001a). Multiple object tracking with a mobile robot. In *Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 371–377, Kauai, Hawaii. 102

Schulz, D., W. Burgard, D. Fox, and A.B. Cremers (2001b). Tracking Multiple Moving Targets with a Mobile Robot using Particle Filters and Statistical Data Association. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seoul, Korea. 102

Schulz, D., D. Fox, and J. Hightower (2003). People tracking with anonymous and id-sensors using rao-blackwellised particle filters. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico. 102

Se, S., D.G. Lowe, and J. Little (2002a). Global localization using distinctive visual features. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 226–231, Lausanne, Switzerland. IEEE Computer Society Press. 80

Se, S., D.G. Lowe, and J. Little (2002b). Local and global localization for mobile robots using visual landmarks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 414–420, Maui, Hawaii. IEEE Computer Society Press. 80

Se, S., D.G. Lowe, and J. Little (2002c). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Int. J. of Robotics Research (IJRR)* 21(8): 753–758. 80

Simmons, R., R. Goodwin, K. Haigh, S. Koenig, J. O'Sullivan, and M. Veloso (1997). Xavier: Experience with a layered robot architecture. *ACM magazine* Intelligence . 9

Simmons, R. and S. Koenig (1995). Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1080–1087. 45, 83

Stone, P., M. Asada, T. Balch, R. D'Andrea, M. Fujita, B. Hengst, G. Kraetzschmar, P. Lima, N. Lau, H. Lund, D. Polani, P. Scerri, S. Tadokoro, T. Weigel, and G. Wyeth (2000). Robocup-2000: The fourth robotic soccer world championships. *AI Magazine* pp. 495–508. 51

Stroupe, A.W., M.C. Martin, and T. Balch (2001). Distributed sensor fusion for object position estimation by multi-roobot systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1092–1098, Seoul, Korea. 100

Tanner, M.A. (1993). *Tools for Statistical Inference.* Springer. 49

Thrun, S. (1998). Bayesian landmark learning for mobile robot localization. *Machine Learning* . 83

Thrun, S. (2000). Probabilistic algorithms in robotics. *AI Magazine* 21(4): 93–109. 9

Thrun, S., M. Beetz, M. Bennewitz, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz (2000). Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research* 19(11): 972–999. 9

Thrun, S., D. Fox, and W. Burgard (2000). Monte carlo localization with mixture proposal distribution. In Kortenkamp, D., R.P. Bonasso, and R. Murphy, editors, *AAAI National Conference on Artificial Intelligence*, Austin, TX. 48, 49

Thrun, S., D. Fox, W. Burgard, and F. Dellaert (2000). Robust monte carlo localization for mobile robots. *Artificial Intelligence* 128(1-2): 99–141. 33

Torr, P. H. S. (1995). Outlier Detection and Motion Segmentation. Ph.D. diss., Dept. of Engineering Science, University of Oxford. 85

Torr, P. H. S. and A. Zisserman (2000). Mlesac: A new robust estimator with application to estimating image geometry. In *Computer Vision and Image Understanding*, number 78 in ., pp. 138–156. 85

Torr, P. H. S., A. Zisserman, and S. J. Maybank (1995). Robust Detection of Degenerate Configurations for the Fundamental Matrix. In *5th International Conference on Computer Vision (ICCV)*, pp. 1037–1042. IEEE Computer Society Press. 85

Tsai, R. Y. (1986). An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In *Computer Vision and Pattern Recognition (CVPR)*, pp. 364–374. IEEE Computer Society Press. 26

Tuytelaars, T. and L. Van Gool (2000). Wide baseline stereo matching based on local affinely invariant regions. In *British Machine Vision Conference (BMVC)*. 85

Uhlmann, J.K. (1995). Dynamic Map Building and Localization: New Theoretical Foundations. Ph.D. diss., Department of Engineering Science, University of Oxford. 22, 23, 30

Veenman, C.J., M.J.T. Reinders, and E. Backer (2001). Resolving motion correspondence for densely moving points. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 23(1): 54–72. 103

Veenman, C.J., M.J.T. Reinders, and E. Backer (2003). Establishing motion correspondence using extended temporal scope. *Artificial Intelligence* 145(1-2): 227–243. 103

Veloso, M., M. Bowling, S. Achim, K. Han, and P. Stone (1998). The cmunited-98 champion small robot team. In Asada, M. and H. Kitano, editors, *Second International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science, pp. 77–92. Springer-Verlag. 100

Vestli, S. J. and N. Tschichold-Gürman (1996). MoPS, a System for Mail Distribution in Office-Type Buildings. *Service Robot: An International Journal* 2(2): 29–35. 37

Wan, E. and R. Merwe (2000). The unscented kalman filter for nonlinear estimation. 41

Wan, E. and R. Merwe (2001). *Kalman Filtering and Neural Networks*, chapter 7. Wiley. 41

Wang, C. C. (1992). Extrinsic Calibration of a Vision Sensor Mounted on a Robot. *IEEE Trans. on Robotics and Automation* 8(2): 161–175. 26

Weigel, T., J.-S. Gutmann, M. Dietl, A. Kleiner, and B. Nebel (2002). CS Freiburg: Coordinating Robots for Successful Soccer Playing. *IEEE Transactions on Robotics and Automation* 18(5): 685–699. 79

Weigel, T., A. Kleiner, F. Diesch, M. Dietl, J.-S Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski (2001). CS Freiburg 2001. In *5th International RoboCup Symposium (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag. 79

Weiss, G. and E. Puttkamer (1995). A map based on laserscans without geometric interpretation. 78

Weng, J., P. Cohen, and M. Herniou (1992). Camera Calibration with Distortion Models and Accuracy Evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 14(10): 965–980. 25

West, M. (1993). Mixture models, monte carlo, bayesian updateing and dynamic models. *Computer Science and Statistics* 24: 325–333. 46

Yagi, Y. (1999). Omnidirectional sensing and its applications. *ICICE Transactions on Information and Systems* E82-D(3): 568–579. 81

Index