Developing Comprehensive State Estimators for Robot Soccer

Thorsten Schmitt, Robert Hanek, and Michael Beetz

TU München, Institut für Informatik, 80290 München, Germany
{schmittt,hanek,beetzm}@in.tum.de
http://www9.in.tum.de/agilo

Abstract. This paper sketches and discusses design options for complex probabilistic state estimators and investigates their interactions and their impact on performance. We consider, as an example, the estimation of game states in autonomous robot soccer. We show that many factors other than the choice of algorithms determine the performance of the estimation systems. We propose empirical investigations and learning as necessary tools for the development of successful state estimation systems.

1 Introduction

Autonomous robots must have information about themselves and their environments that is sufficient and accurate enough for the robots 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 of their environments and their sensor measurements are inaccurate and noisy. Recent longterm experiments with mobile robots [11] 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.

One particularly promising method for accomplishing this is *probabilistic state estimation* [10]. Probabilistic state estimation modules 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 that is available to a robot. Successful state estimation systems have been implemented for a variety of tasks including the estimation of the robot's position in a known environment [2], the automatic learning of environment maps, the state estimation for objects with dynamic states (such as doors), for the tracking of people locations [9], and gesture recognition [11].

So far, research in state estimation for robots has focussed on the development on more reliable, more accurate, and faster algorithms [10]. With the services that autonomous robots are to provide becoming more demanding, the states that the robots have to estimate become more complex. Therefore, the choice of algorithms has become only one among many different factors that determine the performance of the state estimation systems. As a consequence, the application of state estimation algorithms becomes a difficult engineering effort. In this engineering effort, programmers

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 397–409, 2004.

must address questions such as (1) the specification of objectives in terms of cost functions; (2) the filtering of observations; (3) the design of action models; (4) the frequency of updates; and (5) the choice and combination of algorithms. We will show that the proper design of state estimators achieves both substantial and significant performance gains.

In this paper, we argue that developing high performance state estimation systems for difficult estimation problems requires careful system design and analysis. We propose empirical investigations and learning as necessary tools for the development of successful state estimation systems. In the remainder of the paper we proceed as follows. Section 2 describes and discusses the game state estimation problem, the design of cost functions for state estimation, and different design options for state estimation systems and their possible effects on robot performance. The subsequent section 3 describes the design of a state estimator for game situations in autonomous robot soccer and investigates the impact of design decisions on the perfomance of the estimator. We end with conclusions from our results.

2 Complex State Estimation

The task of state estimation is the computation the robot's beliefs about its own state and the state of its environment. The results of the state estimation contain information such as the robot's estimated state, the accuracy and a measure of the ambiguity of the state estimate, the state of other objects.

Approached probabilistically, the state estimation problem can be considered as a density estimation problem, where a robot estimates a posterior distribution over the space of its states and the states of other objects conditioned on the available data. Denoting the state at time t by s_t and the data up to time t by d_t, we write the posterior as $p(s_t|d_t, ..., d_0; m)$. Here m is the world model (e.g., a map). We will also refer to this posterior as $b_t(s_t)$, the robot's belief state at time t.

2.1 Game State Estimation in Robot Soccer

In this paper we apply probabilistic state estimation to the assessment of game situations in autonomous robot soccer. In robot soccer (mid-size league) two teams of four autonomous robots — one goal keeper and three field players — play soccer against each other. The robots of our team are equipped with, among other components, a wireless Ethernet for communication (1), an onboard computer (2), a fixed color CCD camera (3), a guide rail for the ball (4), and a kicker (5). All sensing and all action selection is done on board of the individual robots.

The game state estimators provide the robots' action selection routines with estimates of the positions and may be even the dynamic states of each player and the ball. This estimation problem confronts probabilistic state estimation methods with several challenges: limitations of the robots as well as environmental conditions make the reasoning task difficult to perform. The camera system with an opening angle of 90^o and pointed to the front gives an individual robot only a very restricted view of the game situation. In addition, relevant visual information may be occluded by other robots. Vibrations of the camera, spot light effects, specularity, and shadows cause substantial

inaccuracies. Even small vibrations that cause jumps of only two or three pixel lines cause deviations of more than half a meter in the depth estimation, if the objects are several meters away. Also, the positions of the robots are uncertain and inaccurate. In addition, the robots change their direction and speed very abruptly and therefore the models of the dynamic states of the robots of the other team can only be very crude and uncertain. Finally, vast amounts of data must be processed in real-time.

In our case, the estimated game state consists of the compound state variables $Robot^1$, ..., $Robot^4$, $Ball$, $Opponent^1$, ..., $Opponent^n$. The compound state variable $Robot^i = \langle x^i, y^i, \theta^i, vt^i, vr^i \rangle$ comprises the position (x^i, y^i) and orientation θ^i of robot i and its translational (vt^i) and rotational velocity (vr^i) and $Robot^i_t$ refers to the value of these variables at time step t. Analogously, $Ball = \langle x^{ball}, y^{ball}, v^{ball} \rangle$ denotes the position and velocity of the ball, where the ball velocity is interpolated from the last ball position estimates. Finally, $Opponent^j = \langle x^j, y^j, v^j \rangle$ where v^j is again interpolated from previous estimates.

Unfortunately, the compound state variables are not independent of each others. Inaccuracies in the estimation of a robot's position causes even higher inaccuracies in the estimation of the observed robots. Also, since robots often go for the ball, the ball position often influences the movements of the robots.

2.2 Objectives of State Estimation

For the purpose of this paper, let us consider state estimation tasks that include the detection of objects, the recognition of their identity, and the inference of their states. In this setting performance aspects of state estimation include, whether the state estimation process hallucinates objects (false positive), overlooks objects (false negative), and the expected accuracy of their estimated states.

To capture these concepts we first introduce the notion of an object hypothesis. An ***object hypothesis*** h is a data structure in the robot's belief state that represents an object that the robot believes to exist. An object hypothesis h contains the most likely position $\langle x_h, y_h \rangle$ and a region of possible position r_h. The second concept that we need is the notion of ***designation*** (grounding), which is a mapping of an object hypothesis to an entity in the real world that caused the last observation supporting the hypothesis. This entity might also be a hallucination. To specify the designation function for a game episode a programmer has to step manually through the captured images and assign each observed image blob to an entity in the world.

Because this definition cannot be made operational, we will use a weaker notion of designation that can be fully automated *(this is closely related to the least square criterion)*. We say that an object hypothesis ***designates*** an object o at position $\langle x_o, y_o \rangle$ if (1) $\langle x_o, y_o \rangle$ lies within r_h and (2) there exists no other object hypothesis h' that does not already designate another object such that the distance between the most likely position $\langle x_{h'}, y_{h'} \rangle$ and $\langle x_o, y_o \rangle$ is smaller then the distance between $\langle x_h, y_h \rangle$ and $\langle x_o, y_o \rangle$.

Using our notion of designation we can now state what we mean by overlooking and hallucinating an object. We say a state estimator ***overlooks*** an object o if there exists no hypothesis h in the belief state that designates o. A state estimator ***hallucinates*** an object if its belief state contains an object hypothesis that does not designate any object in the world.

In general, the design of a state estimator is part of the design of a robot controller. Thus, the goal in the design of the robot controller is design a state estimator *se* and an action selector *as* such that taken together they will achieve the optimal performance of the robot. More formally, we intend to find a pair $\langle se, as \rangle$ such that

$$\textit{argmax}_{\langle se,as \rangle}\ \textit{expected-performance(behavior(}\langle se, as \rangle\textit{))}.$$

The drawback of this approach is that the design of the state estimator interacts with the design of the action selector.

We can abstract away from action selection by stating the objective of state estimation as a cost function [3]. In this approach we have to design a state estimator that minimizes the given cost function $argmin_{se} cost(se)$.

So far most state estimators have been designed without specifying sophisticated cost functions. In the museum tour guide projects, some of the most successful applications of state estimation in autonomous robotics, the robot mainly estimated its own position. Therefore, the performance factors are simply whether or not the robot was lost and the average accuracy over the episodes in which it was not lost [4, 5].

In a nutshell, the cost of game state estimation in a soccer situation could be stated as *cost(belief-state,situation)* = w_1 * *hallucinations* + w_2 * *overlooked objects* + w_3 * *avg accuracy*, where w_1, w_2, and w_3 are weights that assess the relative importance of hallucinating and overlooking and the accuracy of observations. In robot soccer it is more important not to overlook objects than to hallucinate them. Overlooking opponents could result in collisions for which the robots might be sent off the field or not knowing where the ball is and therefore not being able to issue goal-directed actions. Hallucinations, on the other hand would mainly cause the robots to follow suboptimal trajectories, which is less critical.

The design of informative cost functions for game state estimation is much more subtle than suggested above and therefore the weights have to be set in situation specific ways. It is more important not to overlook the ball than the opponent players because knowing the ball position is necessary for focussed play. Overlooking team mates is not important at all because the team mates broadcast their own position estimates. It is also much more important to have accurate estimates in the area around the ball and for the ball handling robot than for objects that are not in the focus of the play or for players that only perform backup roles. Stating such informed cost functions is important because they also allow us to exploit task specific simplifications. For example, through the nature of soccer the ball handling robot is typically the one closest to the ball and facing the ball. It is therefore automatically the one that has the most accurate and reliable observations of the area around the ball.

2.3 Design Dimensions of State Estimation

When designing state estimation systems there are many design decisions to make including the ones listed below.

Decision 1: The Form of Probability Densities. One of the most critical decisions to make are the assumptions about the form of the probability densities. In many robot applications, this density is assumed to be unimodal and Gaussian distributed. Here,

the distribution can be represented by the mean value and an associated covariance matrix. This has the main disadvantage that we cannot represent ambiguities in position estimates. These ambiguities can be represented and reasoned about in the Markov state estimation framework. The increased expressiveness must be paid for with higher computational cost. Particle filter are an alternative method in which probability distributions are approximated by sample sets. Particle filter have the advantage that they can be run in resource adaptive fashion. A particularity of particle filter is that they perform worse as data get very accurate.

Decision 2: State Estimation Algorithms. The choice of state estimation algorithms is constrained by the representation of the probability distributions. The most common approaches are Kalman filters, Markov localization algorithms, and particle filters. If states to be estimated involve multiple objects then observations have to be associated with object hypotheses. This is done, for example, in Reid's multiple hypothesis tracking algorithm or in Joint Probabilistic Data association filters. Recently, researchers have been proposed hybrid state estimation mechanisms, in which they combine multiple representations in parallel.

Decision 3: Decomposition and Simplification. A key problem in solving difficult state estimation problems is the complexity of the joint probability density and the huge amount of data that the probability density is conditioned on. This requires us to factorize, approximate, simplify (by making assumptions, such as the Markov assumption), and decompose the probability density [10]. State estimation problems can also be simplified based on their usage. In robot soccer, for example, instead of track exactly each opponent we can confine the estimator to track a superset of opponents. Less pressure to integrate observations of different robots. When making simplifications, assumptions, or approximations it is often possible to provide routines that monitor them.

Decision 4: Probabilistic Models. The update rules for probability densities often use parameter that are to be supplied by the programmer. These parameters can often be derived from problem specific probabilities, such as sensor or action models. There are two ways in which we can set these parameters in more informed ways. First, we can learn the values of these parameters from experience. Second, we can supply the algorithm with situation specific probabilities instead of general ones that average over all possible situations. For examples, in situations where objects become occluded, the estimator should assume a longer lifetime for object hypotheses even without supporting observations.

Decision 5: Observation Filtering. Another important design dimension is how many and which observations to take to maintain the belief state. If the estimator takes too many it might consume too much computational resources. On the other hand, it can often get away with less informative predictive models if it updates its belief with higher frequency.

Another issue is that observations are often corrupted and integrating a corrupted observation causes a less accurate belief state. For example, if a robot turns quickly the odometric data and image data do not correspond well and tracking opponents while

turning quickly is therefore very unreliable. These problems can be mitigated by simply ignoring observations that are probably inaccurate and unreliable.

Decision 6: Managing Computational Resources. Yet another issue is what to spend the computational resources for. Typically, in autonomous robot control the robot has a limited amount of computational resources available that can be assigned to the different computational tasks. For example, the programmer can spend resources for a more accurate interpretation of sensory data, it can apply more sophisticated probabilistic models for predicting the resulting state (eg, ones that allow for predicting nonlinear movements), or it can use crude and cheap computation mechanisms and rather run the iterative state estimation with a higher frequency.

Discussion. We have sketched in this section a number of dimensions for the design of complex state estimators. Unfortunately, there is no general model of the influence of these design decisions on the performance of a state estimation system. Even worse, the different design decisions interact with each other in very subtle and application specific ways. Therefore, it is necessary to design state estimators specifically for the application at hand and to empirically evaluate the design decisions by comparing the expected performance with respect to the specified cost function.

3 Empirical Investigation

We will now sketch the design and the main parameters of the game state estimator and then investigate the effects of design choices on its performance.

3.1 The Game State Estimator

The game state estimation subsystem consists of the perception subsystem, the state estimator itself, and the belief state. The perception subsystem 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 position estimate for each dynamic task-relevant object.

The state estimation subsystem consists of three interacting estimators: the self localization system, the ball estimator, and the opponents estimator. The self localization estimates the probability density of the robot's own position based on environment features, the ball position, and its predicted position. The ball localizer estimates the probability density for the ball position given the robot's position, its observation, and the ball estimates of team mates. Finally, a robot estimates the positions of the opponents, based on its own position, the robots' appearances in the images, and the estimations of the team mates.

The decomposition of game state estimation reduces the overall complexity of the estimation problem and enables the robots to exploit the structures and assumptions underlying the different subtasks. However, as stated in the previous section, the different estimation problems are not independent.

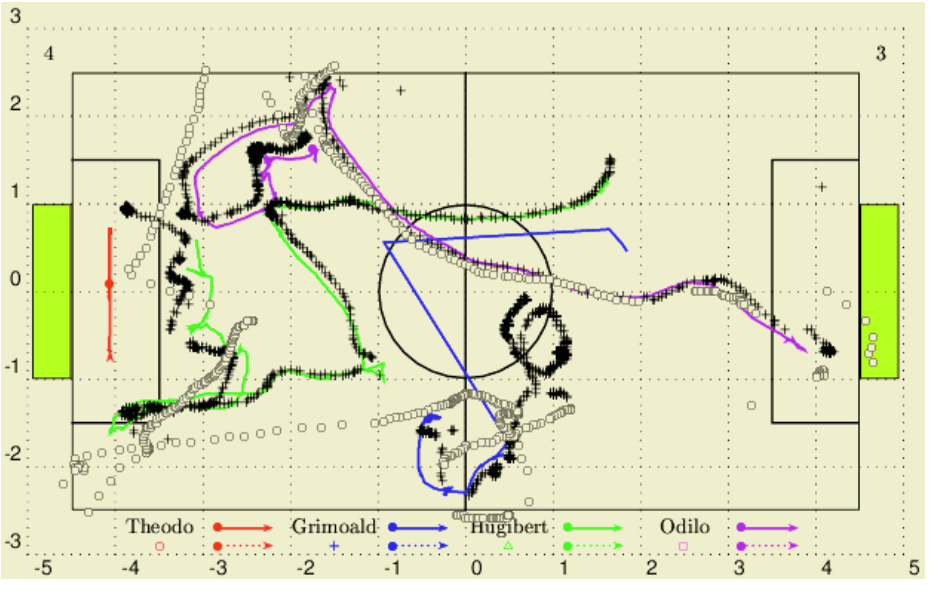
Self Localization. State estimation is an iterative process where each iteration is triggered by the arrival of a new piece of evidence, a captured image, an odometry reading, or a partial state estimate broadcasted by another robot. These data are integrated over time to a maximum a posteriori (MAP) estimate of the robot's pose. The MAP estimate $\widehat{\mathbf{\Phi}}$ is given by $\widehat{\mathbf{\Phi}} = \operatorname{argmax}_{\mathbf{\Phi}}\ p(\mathbf{\Phi}) \cdot p(data|\mathbf{\Phi})$ where $p(\mathbf{\Phi})$ is the prior of the pose $\mathbf{\Phi}$ summarizing all evidence gathered in the past. The prior at the current time step is obtained by predicting the pose distribution estimated for the previous time step. The second term $p(data|\mathbf{\Phi})$ is the likelihood of receiving $data$ given the robot's pose $\mathbf{\Phi}$.

Self localization runs a fast Kalman filter for tracking the position of the robot and thereby makes the assumption that the probability density is Gaussian. To detect situations where this assumption yields localization failures a particle filter with a lower frequency is run concurrently. In cases where evidence implies multi modal probability densities the particle filter detects that the robot gets lost and provides the different local maxima for reinitializing the Kalman filter.

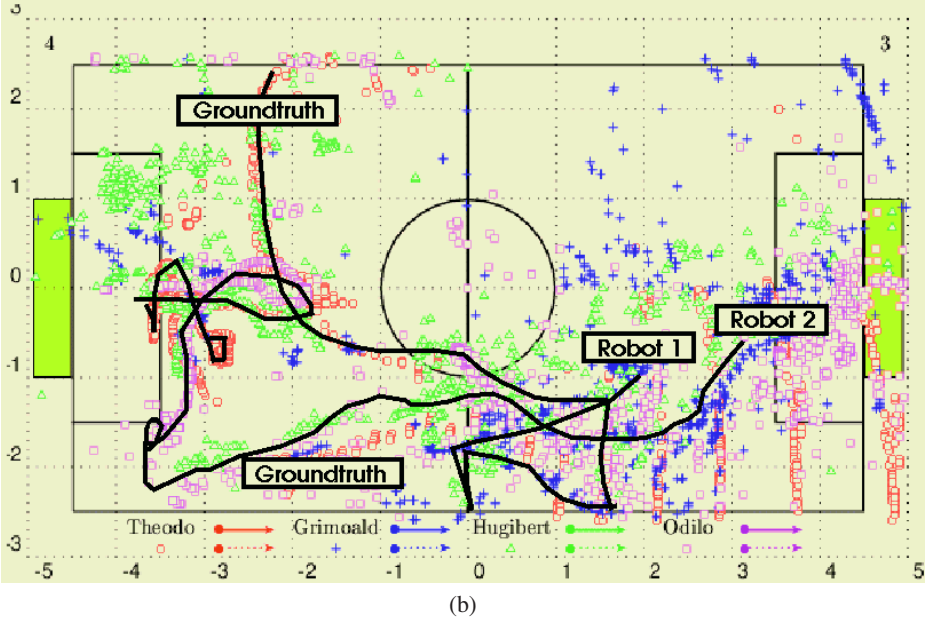
Parameters and Assumptions. The parameters of the self localization module that can be set include the the probabilistic models $p(data|\mathbf{\Phi})$ for the different kinds of data. The programmer can also decide on the mechanisms applied to image interpretation, the frequency at which to run the algorithms, and which observations to take for state estimation.

Opponent and Ball Tracking. For opponent tracking a variant of Reid's Multiple Hypothesis Tracking (MHT) algorithm [6] is used. The objective of the MHT algorithm is to maintain a set of object hypotheses, each describing a unique real object and its position and estimate the likelihood of the individual hypotheses. The MHT algorithm deals with two kinds of uncertainties. The first one is the inaccuracy of the robot's sensors and is represented using a Gaussian probability density. The second kind of uncertainty is introduced by the data association problem, i.e. assigning feature blobs to object hypotheses. It is represented by a hypotheses tree where nodes represent the association of a feature blob with an object hypothesis. A node h_j^k is a son of the node h_i^{k-1} if h_j^k results from the assignment of an observed feature blob with a predicted state $\tilde{h}_i^{k-1}$ of the hypothesis h_i^{k-1}. Computing the association probability $P(h_{ij}^{k+1}|Z(k))$, which indicates the likelihood that observed object and object hypothesis refer to the same object given $Z(k)$, the sequence of all measurements up to time k, is the heart of the MHT algorithm [1, 7].

Parameters and Assumptions. The opponents tracking system can be parameterized as follows. First, we can set the minimum likelihood of a hypothesis to be kept in the hypotheses tree. If the threshold is set higher then the tree is pruned more aggressively. Smaller trees require fewer computational resources but have a higher risk of deleting correct hypotheses from the tree. Other parameters include the average time of detecting spurious features and of keeping hypotheses without supporting observations, which is needed to deal with occlusions. Additional parameters are the motion models for opponents, the routine for deciding whether an observation is informative enough to improve the belief state.

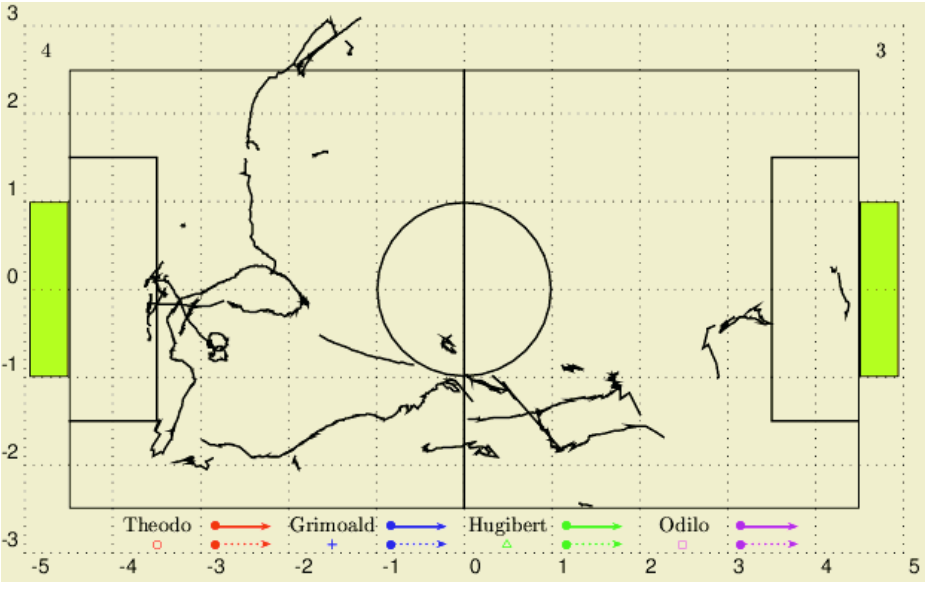


(a)

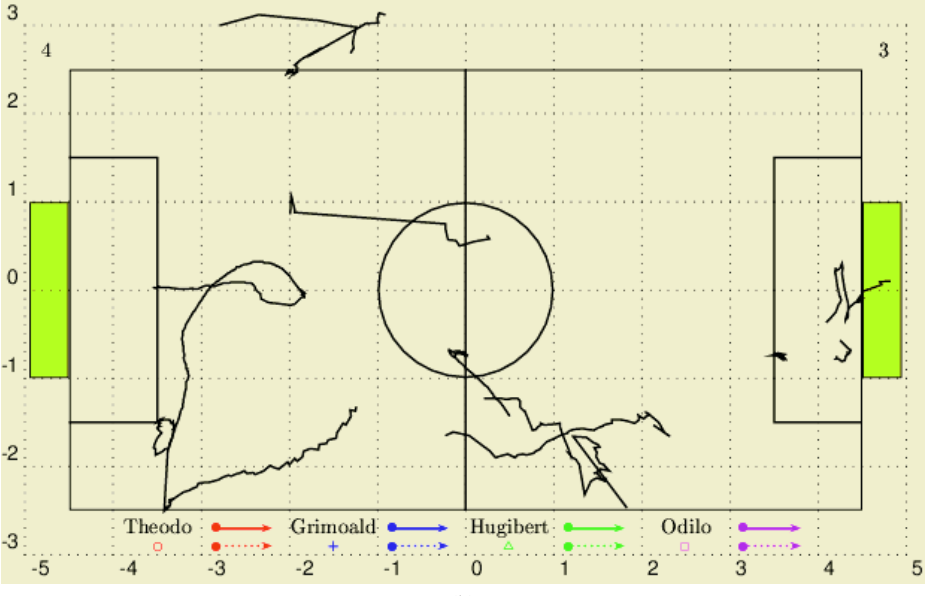


(b)

Fig. 1. Game episode (100 seconds long). Subfigure (a) shows the results of self localization. The trajectories of the robots are plotted as solid lines. Ground truth position data for robots is indicated with '+'. Ground truth position data for the ball is indicated with 'o'. Subfigure (b) shows the individual observations of an opponent where the color indicates which one of the robots made the observation. The overlying black lines are the ground truth data.



(a)



(b)

Fig. 2. Game episode (100 seconds long). Subfigure (a) shows the tracks of opponent robots exploiting observations made by all team mates. Subfigure (b) shows the tracking of opponents without cooperation.

3.2 Setup of the Experiment

To get a better understanding of the design options, we have recorded robot soccer games to evaluate system performance more carefully. To do so, we have mounted a ceiling camera into our robot lab in order to record a bird's eye view from the soccer games that will serve as the ground truth for our experiments. The state estimator for the ceiling camera has an average accuracy of 10-15cm depending on the situation and the position on the field. For evaluating the results of game state estimation we use the criteria of hallucinations and overlooking that we have stated in section 2.

To acquire the data for our empirical studies we have played friendly games against the Ulm Sparrows for a total net playing time of more than two hours in the setup described above. The following subsection will describe some of our first preliminary findings about the operation of the game state and the effects of setting certain parameters.

3.3 Experiments and Experiences

Fig. 1 shows a typical result of game state estimation with standard parameterization of the system. The Kalman filter self localization runs with a frequency of 20-30 Hz (frame rate), the particle filter self localization at about 15Hz and with 1000 samples. The MHT algorithm runs at 4*20 Hz. The action models for opponent tracking assume constant speed with a process variance of 0.03. The criterion for associating an observation with a hypothesis is a Mahalanobis distance of less than 5.9, which means that the observation lies within the 95% confidence interval of the prediction. Key performance criteria of the opponent tracking system are listed in the subsequent table.

	Robot 1	Robot 2	Robot 3	Robot 4	cooperation	1/4
Self-Loc. accuracy	11 cm	48 cm	28 cm	19 cm	—	—
Tracks	1038 51%	992 49%	1067 53%	1162 58%	1278 64%	635 31%
Tracks correct	820 41%	720 36%	893 44%	1002 50%	1087 55%	528 26%
Tracks hallucinated	218 10%	272 13%	174 9%	160 8%	191 9%	107 5%
Tracks omitted	913 59%	1008 51%	933 47%	838 42%	722 36%	1365 69%
Tracks accuracy	33 cm	37 cm	36 cm	31 cm	23 cm	37 cm

Self localization performs worse than expected [8]. The reason is that the soccer field has changed due to rule changes. The field has become larger which causes higher inaccuracies in vision-based depth estimation. Also, the field provides fewer features that can be used for self localization and the lack of a surrounding wall yields observations outside the field. Qualitatively, the robots loose track of their positions much more often and jump to alternative position estimates because of the particle filter. When the robots roughly know where they are then the accuracy is still sufficiently high (around 10-20cm).

The table lists the results for a robot with cooperation and the individual robots as columns. The rows represent the different performance measures. The maximal number of base points for the opponent tracks (given omnidirectional view and no occlusion) is 2000 for the individual robots and the cooperating robots. From those a robot with cooperation was able to observe 1278 tracks (64%), 9% of those where hallucinated track

points. In the table these numbers are listed as observed tracks, correct designations, hallucinated tracks, and average accuracy.

Cooperation Helps – Does It? We can see that in our episode cooperation increases the number of detected tracks as well as increases the accuracy, which is typical. These results are depicted in Fig. 2. Fig. 2 (a) shows the opponents tracked with cooperative tracking and Fig. 2 (b) the same estimation made without cooperation. You can see that gaps in the tracks that are mainly caused by occlusions can often be closed by using the observations of team mates. Accuracy can be substantially increased by fusing the observations of different robots because the depth estimate of positions are much more inaccurate than the lateral positions in the image. This can be accomplished through the Kalman filter's property to optimally fuse observations from different robots into global hypotheses with smaller covariances.

However, cooperation is not always a good idea. We have experienced this when running the game state estimation under extremely poor lighting conditions. Under these conditions self localization could not well discriminate between some symmetric positions on the field. Therefore, integrating observations from dislocalized robots caused incoherent hypotheses and we were forced to disable the cooperation to get more stable estimates.

Specify Cost Functions Explicitly! If we add a notion of relevance to the opponent players depending how important they are for the game state then the percentage of coverage of the *relevant* opponents is much higher. This is because the robots often look into the direction of the ball and therefore see the opponents that are close to the ball more often. It would be inadvantageous to design the state estimator such that it has a uniform coverage of all positions on the field.

Tune the Parameters of Estimation Algorithms. Tuning of algorithm parameters has an enormous impact on the performance of the systems. The most promising approach seems to be to learn these parameters from the experimental data. Due to space limitation we cannot discuss the issues here.

If You Can't Do It Accurately Then Do It Fast! The column *1/4* in our table shows the performance of the system if we run state estimation at 0.25 of its normal frequency. This causes less accurate position estimation as well losing more than half of the tracks because of less redundant information and stronger dependency on action models. Recall that we assumed motion to be constant whereas motions in robot soccer are very often changed abruptly. This suggests that we can deal with worse motion models by estimating with a higher frequency.

Implement Estimators That Check Their Work! We have pointed out in section 2 that Kalman filter localization is fast but relies on the assumption that the probability distribution for the robots is Gaussian distributed. We use a particle filter in parallel that runs at a third of the frequency of the Kalman filter as a monitor that detects ambiguous position estimates and that initializes the Kalman filter localization after the position track is lost. We didn't use a particle filter on its own because we couldn't run it fast enough

and because Gaussian distributions can be communicated much more compactly and used as evidences in the other state estimation problems. Our findings are that the time needed for relocalization could be reduced from about 10 seconds to about 2 seconds and that lost position tracks could be detected earlier.

Lesson. Applying multiple state estimation techniques with different strength and weaknesses in parallel is a viable design option for state estimators. In particular, if the computationally more expensive method can be used to monitor the faster but less reliable method.

Learn to Look Away! In state estimation it is often assumed that every observation provides the robot with additional information that can be used to improve the state estimate. In reality, however, this is often not true. We have used Quinlan's C4.5 decision tree learning algorithm to learn predictive rules as to whether or not to integrate an observation into state estimation considering the current situation. The robot learned rules that state, an observation is likely to be informative (within 30cm of the ground truth position) if

1. the observed distance D is less than 3.76086 and the robot has not lost track of its position
2. the robot is not turning, the angle ϕ between the observation and the direction the camera is pointed to is less than 22.91° and $D \leq 5.4253$
3. the robot is turning, $D \leq 6.87$, and $\phi \leq 43.6$

Applying these rules to filtering observations we could reduce the hallucinated tracks by one third, and further improved the track accuracy, and overlooked fewer tracks (which get otherwise corrupted by the noisy observations).

Lesson. What is interesting here is not the specific feature language and rules. They are different for different robots, environments, and cost functions. What is important is that we can try to learn predictive models as to whether or not an observation can be expected to improve the state estimate. These rules can be used to filter out worthless observations.

4 Conclusions

In our view, research in state estimation is focusing too much on algorithm design and analysis and too little on system design. As we will apply state estimation in very complex autonomous robot applications, such as household robotics, where the estimated states are extremely high dimensional we won't be successful unless we know how to parameterize the systems and how to provide them with the necessary probabilistic models. Therefore, our most important conclusion is an obvious one: the development of complex high-performance state estimation systems is a complex design problem with many design options. The choice of estimation algorithms is only one of these options but many other design dimensions have an equally large impact on system performance. The design has to be tailored to the particular application at hand and the different design option interact with each other in obscure and opaque ways. We propose empirical investigations and learning based on ground truth data as necessary tools

for the development of successful state estimation systems. We have illustrated these issues using a probabilistic game state estimation in autonomous robot soccer. Our results are preliminary and a lot more has to be done to understand the design of complex state estimators properly.

References

1. Y. Bar-Shalom and T. Fortmann. Tracking and data association. Academic Press., 1988.
2. D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.
3. Z. Ghahramani, D. M. Wolpert, and M. I. Jordan. Computational models of sensorimotor organization. In Morasso and V. Sanguineti, editors, *Self-Organization Computational Maps and Motor Control*, Amsterdam, North-Holland, 1997.
4. J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.
5. J.-S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.
6. D. Reid. An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control, 24(6):843–854, 1979.
7. T. Schmitt, M. Beetz, R. Hanek, and S. Buck. Watch their moves: Applying probabilistic multiple object tracking to autonomous robot soccer. In *AAAI National Conference on Artificial Intelligence*, pages 599–604, Edmonton, Canada, 2002.
8. T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Trans. on Robotics and Automation*, 18(10):–, 2002.
9. D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. Multiple object tracking with a mobile robot. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 371–377, Kauai, Hawaii, 2001.
10. S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.
11. S. Thrun, M. Beetz, M. Bennewitz, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.