Handling Diverse Information Sources: Prioritized Multi-Hypothesis World Modeling

Paul E. Rybski **Manuela M. Veloso**

December 2006
CMU-CS-06-182

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Keywords: world modeling, situation awareness

Abstract

To act intelligently in complex and dynamic environments, mobile robots must estimate the position of objects by using information obtained from a wide variety of sources, including different sensors, kinematic models, and communication from teammate robots. For any problem of reasonable complexity, mobile robots do not have the sensing capabilities necessary to simultaneously perceive all aspects of a dynamic environment, nor can they correct for possible unmodeled dynamics and noise. The ”view” of the robot's sensors is hence narrow compared to the size of the environment. While the state of a single object is being updated, the evolving state of all other non-sensed objects must be predicted from models. In this paper, we formally describe the problem of estimating the state of objects in the environment where the robot can only task its sensors to view on object at a time. We contribute an object tracking method that generates, maintains, and selectively uses a disjoint space of hypotheses, whereby each hypothesis consists of a probabilistic state estimate that is generated by the individual sources of information. The priorities are set by the expected uncertainty in the object's motion/process model, as well as the uncertainties in the sources of information used to track their positions. Each individual robot guides its actions to track an object by selectively iterating through the multiple hypotheses sorted in terms of their expected state information. Our approach has been fully implemented in a team of AIBO soccer robots and used successfully in the RoboCup soccer competition. We describe the algorithm in detail and show extensive empirical results in simulation that demonstrate the effectiveness of our approach as well as an illustrative example from our actual robots.

1 Introduction

Robot perception processing consists of a mapping from sensory data to an estimate of the state of the elements of the environment that are of relevance to the task under execution. For example, a robot traversing a maze needs to estimate the area and position of open space and walls from its sensory data. The complexity of state estimation greatly increases with the task, the dynamics of the environment, and the sensing capabilities of the robots.

In our work, we assume that the individual robots do not have the sensor capacity necessary to perceive the multiple elements of a complex task under execution in a dynamic environment. Concretely, we investigate the problem when robots have limited and narrow perceptual scope, such that they are only capable of observing a single object (or a reduced set of objects) at a time with their sensors. Thus, the relative size of the robot's sensor scope is so small compared to the environment that while the state of a single object is being updated by the sensors, the evolving state of all other non-sensed objects must be predicted from models learned from observations or provided *a priori*. Adding to the complexity of the problem, not all sources of information about a single object can and should be handled equally, as in the traditional sense of weighting those estimates by their covariance. There are times when empirical evidence has proven that some modalities must be ignored as they are unreliable in certain circumstances. Additionally, non-deterministic effects of actuators can create several distinctly different potential outcomes, each of which must be tracked and reasoned about separately.

To address this challenge, we define a method for reasoning over a disjoint hypothesis space whereby high-level domain knowledge is used to impose a strict ordering on estimates created by different sources of information. By segmenting the sources of information used to reason about the state of environmental quantities into different classes, each with different state dynamics and expected effect of robot actions, a prioritized hierarchy of state estimates can be inferred. Additionally, when tracking multiple objects simultaneously, the evolving states of those objects must be considered carefully when deciding where to task the robot's sensors.

We describe a hybrid state estimation algorithm that attempts to reduce the complexity of the generated probability density functions over a quantity of interest by factoring the problem into a series of small estimation problems that are tied to the different sources of model world information possessed by the robot. A high-level policy is used to determine where to task the robot's sensors to best track the objects in the environment. Such policies for creating hierarchies can be defined *a priori*, or they could potentially be learned from data. Using this policy, the decision process that governs each individual robot's actions can easily select the most informative state estimate to use as its input. The priorities are set by the expected uncertainty in the object's motion model, as well as the uncertainties in the sources of information used to track their positions. A robot's actions directly affect its perception of the environment as well as the environment itself, and the best estimate is often one that will allow the robot to obtain more information about its surroundings to further clarify its estimate of quantities of interest. This in turn provides more information to the robot that further updates the ordered hierarchy of possible estimates.

This paper describes an *active* state estimation algorithm, as applied to a real-time adversarial multi-robot domain, which combines action policies determined from high-level domain knowledge with multi-modal probabilistic state estimators. We have successfully applied this approach

to the RoboCup 4-Legged league where a team of Sony AIBO robots autonomously play soccer against another team of AIBO robots, as shown in Figure 1.

Figure 1: Sony AIBO robots preparing to play soccer at a RoboCup competition.

2 Related Work

We discuss some related work along the three main aspects of our work: (i) probabilistic state estimation; (ii) multi-object targeting; and (iii) reasoning about multiple hypotheses from multiple sensing sources.

Most probabilistic estimation techniques follow a Bayesian filtering approach [10] and have been successfully applied to robot state estimation e.g. [19]. Object tracking using a Bayesian filter formalism relies on an *a priori* model of the object's motion that allows the algorithm to predict the object motion given noisy observations. One of the most widely used methods for state estimation is the Kalman filter [11], in which the system model is assumed linear and the noise is assumed Gaussian. When the linearity assumption becomes a limitation, the dimension of the state vector can be changed as the tracked object changes its perceived dynamics, such as with a

variable state dimension (VSD) filter [4]. We also consider the object dynamics, but our approach changes the number of hypotheses, while the specific dimensions of those hypotheses' estimates do not change. Furthermore, we maintain multiple hypotheses independently as potential object locations.

An approach to reasoning about a complex motion model consists of maintaining multiple models. The interacting multiple models (IMM) filter [3] uses a weighted mixture of different process models. Our approach differs in that it maintains a disjoint set of hypotheses which are not merged or fused [7], but are prioritized and visited according to a specific policy. Similar approaches maintain separate estimations based on subjective sensing and other sources, e.g., sensing from robot teammates [13, 17].

A more general approach is the Switching Kalman filter model [14], which represents multiple independent system state dynamics models and switches between them (or linearly combines them) to best fit the observed (or predicted) non-linear dynamics of the system being modeled. Our approach creates multiple independent belief states (or hypotheses) rather than a single state with multiple potential models.

A Multiple Hypothesis Tracking (MHT) [16, 6] approach uses multiple independent state estimators to estimate a multi-modal probability density. This approach has been used successfully for challenging mobile robot localization problems [18], where non-parametric distributions are estimated through sampling techniques, such as the particle filtering [2]. The number of particles used can be dynamically adjusted as computational resources become available or are needed elsewhere [8]. Approaches that factor in a joint state estimation have been used successfully for tracking an object with a mobile robot [12], where the actions of the robot change the process characteristics of the tracked object. Our approach extends the MHT paradigm by reasoning about the different hypotheses as a function of the source of information that generated them.

Finally, multi-object tracking is a complex problem addressed by different approaches that capture connected dynamics of the multiple objects. We address the problem of multi-object tracking from a different perspective, namely in terms of sensor planning [1].

We consider that the robot has a narrow sensor scope incapable of capturing more than one object at a time. Our algorithm includes a policy for directing the sensor machinery towards multiple objects. Furthermore, we consider different types of objects with different motion models which are used to update the confidence on the state estimation of each individual object.

3 Challenges of Dynamic World Modeling

We are interested in problems associated with having a robot autonomously build and maintain accurate world models in dynamic environments where the states of many objects must be estimated simultaneously. A robot will typically be able to make use of multiple sources of information that can describe the motion of objects in the environment. In any environment of reasonable complexity, a robot is incapable of viewing the entire environment at a single time with its sensors. In the extreme case, the robot can only track a single object at a time with its sensors. Figure 2 illustrates the general class of world modeling issues addressed in this paper. We are primarily concerned with the issues involved with *object tracking* rather than issues involved with the complementary

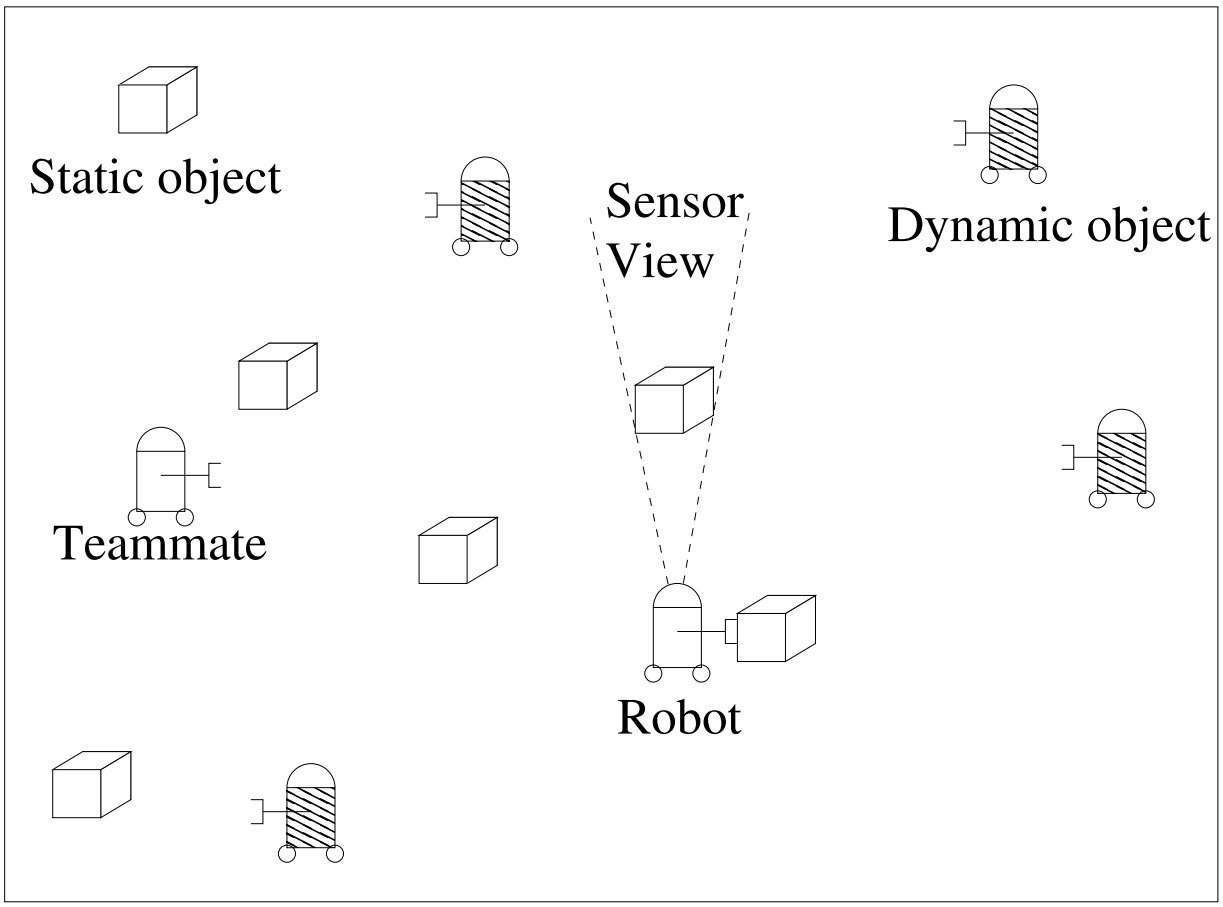


Figure 2: The general world modeling problem in a dynamic environment includes requiring that a robot use a narrow scope sensor to track the positions of multiple (static and dynamic) objects in an environment. Determining when and how to use additional sources of information, such as from the effects of actuation, and teammate sensor information is a non-trivial task.

field of *map building* which is not part of our discussion.

We consider the challenges of tracking multiple objects, where each object has multiple sources of sensor and model information that are available as a combined problem. In order to keep track of the positions of all objects in the environment, the robot must continually re-task its sensors to refresh the models with more accurate position data. Deciding which object to track next is dependent on the expected uncertainty in the motion model for that object as well as the availability and quality of the different sources of information that can provide estimates for the expected position of the object.

- A : the set of all actions $a(t) \in A$, including the null action, that the robot is capable of performing at time t.
- O : the set of all objects in the environment where O_j is the jth object of which the robot

must keep track. These include moving objects of which the robot must maintain an accurate estimate as well as stationary objects with which the robot must maintain periodic contact (such as landmarks for localization).

- $X_{O_j}(t)$: the estimated position of object O_j at time t.
- L_{O_j} : a sensor observation of object O_j which can be null in the case that the robot does not perceive object O_j.
- M_{O_j} : the motion model as a function over objects and robot actions. Defines the expected change in object position over time regardless of whether the robot has obtained a sensor observation.

$$M_{O_j} : (X_{O_j}(t), a(t)) \rightarrow X_{O_j}(t+1) \tag{1}$$

- $S^r_{O_j}$: the sensor model of robot r as a function over objects and sensor observations. Defines the updated object position at the current time.

$$S^r_{O_j} : (X_{O_j}(t), L_{O_j}) \rightarrow X_{O_j}(t+1) \tag{2}$$

Note that in the case of no observations of object O_j at time t, the output position is identical to the input position:

$$S^r_{O_j} : (X_{O_j}(t), NULL) = X_{O_j}(t) \tag{3}$$

- $E^{r_\alpha}_{O_j}$: a sensor observation from an external source, such as another robot r_α:

$$E^{r_\alpha}_{O_j} = S^r_{O_j}, \text{ where } r_\alpha \neq r \tag{4}$$

For the problems in which we are interested, we identify three different classes of objects that have distinct motion model dynamics.

1. **Static:** objects that do not move on their own, such as landmarks used for localization. Even though these landmarks do not move, the robot's own position estimate over time is uncertain. This causes the expected position of this object to drift as the robot's own state changes.
2. **Quasi-Dynamic:** objects which do not move on their own, but which move by being manipulated or pushed by a robot. The motion model for this kind of object encapsulates the actuation dynamics from manipulation when the robot manipulates it, but also must take into account that the object can move unexpectedly when another robot makes contact with it.
3. **Dynamic:** objects, such as other robots, which can move under their own power and control. The internal state of these robots is unobservable and so their motion can be difficult to predict.

4 Multi-Object Tracking

For any environment of interest, a robot's sensors will not have the capability to view all aspects of an environment at the same time. Thus, the robot must change the direction that its sensors are pointing in order to continually update its world model with new readings of the objects that it is tracking. In the most difficult case, the robot can only track a single object at a time and must predict the positions of the other objects with their motion models. The longer an object is not visible, the less accurate the robot's model will be due to noise and unmodeled dynamic changes in the object's motion. Deciding how and when to re-task the robot's sensors depends highly on the objects being tracked as well as the environment in which they exist.

Our solution is to define a policy over all objects which describes when the robot should point its camera from one object to the next. A formal description follows:

- A_L : a subset of actions A which cause the robot to change the angle of its sensors in order to gather a new observation of an object.
- π_{ob} : a policy over the position of objects X_{O_j} which decides which object the robot should track next.

$$\pi_{ob} : (\vec{X_{O_j}}) \rightarrow a_j \tag{5}$$

  π_{ob} takes as input the vector of all estimated object positions X_O and computes the best action $a_j \in A_L$ (possibly NULL if no best action exists) that will move the robot's sensors to track an object O_j.

Two functions for π_{ob} are considered in this paper:

1. **Naive** : takes no notion of object uncertainty into account. Cycles robot's sensor between all objects equally.
2. **Greedy** : selects the object with the greatest uncertainty to track. Expected uncertainty is derived from the motion models for the object.

5 Prioritized Multi-Hypothesis Model Tracking

To effectively estimate the state of objects in the environment, sensor observations $S^r_{O_j}$ must be obtained which provide some update as to the position of the object. In the absence of good sensor readings, models M_{O_j} of the expected motion of the objects must be used to predict the change in the object's state. In all but the most degenerate cases, such models will not be able to completely describe the motion of the object. Noise and unexpected changes in the dynamics of the object will cause the robot's estimate to rapidly diverge from the object's true position.

Multiple sources of information exist that a robot can use to search for an object that is not visible in its sensors. Each source represents a potential hypothesis on the location of the object. For example, non-deterministic effects of actuators can create several distinctly different potential outcomes, each of which should be tracked and reasoned about separately. Similarly, teammates

may provide some information about the state of an object, but the quality of this information could be quite poor if the position estimate of the teammates is erroneous due to localization errors.

Our approach to the problem of object estimation, where multiple sources of information about the objects are available, is to define a policy over the set of objects which prioritizes when and how the robot should task its sensors based on the kinds of objects being tracked, information returned from sensors about the object, and *a priori* models of how the robot interacts with the object (such as with its actuators).

- h_i : a hypothesis over the location of object O_j. Each hypothesis is defined as: $h_i : P(X_{O_j})$ where $P(X_{O_j})$ is a probability distribution over X_{O_j} consisting of the pose and uncertainty.
- H_{O_j} : the set of hypotheses h_i that represent the set of possible locations for object O_j.
- π^j_{mh} : a policy for a particular object O_j which describes a ranking of the different hypotheses h_i that could exist at any given time for it.
- f_{mh} : a function over the sources of information that can be used to predict object O_j's location at time t.

$$f_{mh} : (M_{O_j}, S^r_{O_j}, < E^{r_\alpha}_{O_j}, \ldots >, \pi^j_{mh}) \rightarrow X'_{O_j}(t) \tag{6}$$

  where $X'_{O_j}(t)$ is the most highly ranked pose given the set of available hypotheses, and $< E^{r_\alpha}_{O_j}, \ldots >$ is the set of all available observations from teammates. Table 1 illustrates the hypothesis ranking function.

The following list illustrates how information returned from the robot's sensors, model information from actuators, and teammate observations can be ranked:

1. Robot's own sensors
2. Successful actuation
3. Failed actuation
4. Teammate observations

In this case, the robot's own sensors, such as a camera, is trusted over all other sources of information. All other sources of information are not sensed directly but are instead obtained indirectly through models and teammate information. Actuation is assumed to be done blindly where the contact with the object is invisible to the robot's cameras. Actuator success is assumed over failure. Finally, because of the possibility of poor self-localization, teammate information is listed to only when no other sources of information are available.

Because all hypotheses are represented as probability distributions, their state can be estimated with appropriate probabilistic tracking algorithms, such as the Kalman Filter, Particle Filter, or other Bayesian filter-based approaches.

$f_{mh} : (M_{O_j}, S^r_{O_j}, < E^{r_\alpha}_{O_j}, \ldots >, \pi^j_{mh}) \rightarrow X'_{O_j}(t)$

- **Given:**
  1. H = list of hypotheses H_i (from last invocation)
  2. H' = [] (empty list of hypotheses)
- **Execute:**
  - Generate hypotheses h'_k from M_{O_j}, $S^r_{O_j}$, $< E^{r_\alpha}_{O_j}, \ldots >$, and add to H'
  - **for each** h'_k in H' **do**
    - **if** h'_k matches some h_i in H
      - Update h_i with data from h'_k
    - **else**
      - Add h'_k to H
  - Rank and sort hypotheses in H based on policy π^j_{mh}
  - Prune all h_n from H if uncertainty over threshold
  - **If** H is empty **return** NULL
  - **return** h_1 (first ranked hypothesis)

Table 1: Definition of Hypothesis Ranking Function

6 World Modeling in a Multi-Agent Dynamic Adversarial Domain

In the RoboCup 4-Legged league, two teams of four Sony AIBO robots autonomously play soccer against one another. While robots on the same team use 802.11b wireless Ethernet to communicate with each other, no additional off-board computation is allowed. A deployed team becomes a distributed sensor processing network. Several sources of information are available to each robot that allow it to build a model of its environment, including its sensors, kinematic models of its own body and actuators, and information communicated to it from its teammates. Additionally, each team possesses an *a priori* map of the field which gives the locations of the markers, goals, and field lines in a global reference frame.

In the RoboCup domain, knowing the location of the ball at all times is critical for successful play. The problem of knowing the ball's position is a challenging combination of active search and tracking. A number of specific factors serve to confound the modeling problem. Each of these factors contributes a quantity of error that introduces noise that must be contended with. Unfortunately, the full extent of some of the noise factors is extremely difficult to model. These factors include:

- **Inaccurate sensing:** Each robot is equipped with a color digital camera, located in the front of its head, that it uses to perceive the world. Because the robot is very low to the ground, its view can very easily and quickly be occluded by opponents and/or teammates. It is worth noting that visual observations of these environmental features are used to localize the robot on the field . When the robot is actively tracking the ball, it is typically unable to localize as often, which contributes to pose uncertainty error.

- **Interactions between the robot and target:** The four-legged chassis of the AIBO gives it a wide variety of motions that it can use to manipulate objects such as the ball. However, due to slippage of the joints and variability of the initial starting positions of robot and ball, the effects of these actions can vary considerably. Specifically, the effect of an action can have a single successful mode and multiple independent failure modes, each of which has its own dynamic characteristics.

- **Interactions between the robot and environment:** The four-legged chassis is also a large source of odometric noise as the complex physics of how the robots limbs strike the ground coupled with the fact that the robot is typically jostled heavily during game play means that the robot's confidence in its own position can very quickly become erroneous even if it had very recently correctly localized itself.

- **Erroneous information from teammates:** Using their wireless Ethernet, the robots can share local observations made about the environment with their teammates. Because the robots do not have a centralized server, they do not have a method of synchronizing their internal clocks. The lack of accurate timestamps on observations makes fusion of the sensor data much more challenging. Because of the positional uncertainty, global positions of objects reported by teammates can very easily be erroneous if the robot's position or (more importantly) its heading are estimated badly. This source of information is very likely the most problematic as a teammate can broadcast a very tight and accurate covariance estimate even though it has become very poorly localized due to an undetected collision with an opponent. These errors are highly non-linear as errors in robot orientation contribute greatly to errors in reported ball pose.

Attempting to reason effectively about each of these sources of error directly can be very challenging and difficult to do precisely. Because a robot's actions directly affect its position in the environment and the position of the ball, as well as the amount of information that the robot can obtain from its sensors, the algorithm for selecting the correct action to perform at a given time is extremely important.

6.1 Prioritized Multiple Hypothesis Object Tracking

In our group's long experience with the RoboCup legged league, we have observed many effects of noise on our robots that are caused by such a dynamic environment. In particular, we have identified a number of places where more abstract knowledge about the high-level domain can be helpful when estimating the position of the ball on the field.

1. Occlusions occur enough that the ball can often be in the robot's visual field even though it is behind another robot. Persistence in searching for the ball in an area last believed to be its location is preferable to immediately giving up the search and looking elsewhere on the field. Thus, the actions performed by the robot are highly dependent upon the source of information used to generate the hypothesis being tracked.

2. Ball estimates returned from teammates are never as accurate as the robot's own estimates. Our team uses a dual world model [17] where the robot's own perceptions build a model that is kept independent of the model built from its teammate's perceptions.

Our approach tags the contribution of each source of information to the state estimate. This allows additional information, such as the utility of the source of data on the estimate, to play a factor in the decision processes that the robot makes when solving its task. Concretely, the state vector to be estimated is segmented into a set of parallel and independent hypotheses, each of which represents a probability distribution over the state vector. These individual estimates are maintained in parallel and an external decision process chooses which ones to ignore and which one (or ones) to use.

6.2 RoboCup Hypothesis Selection Policy

In order to incorporate domain-specific data into the estimation algorithm that can be used in hypothesis ranking and persistence, we define a specific policy for reasoning about the specific sources of information. In the RoboCup 4-Legged league, there are multiple sources of information that must be accounted for when tracking the ball. The individual sources of information are used to generate a disjoint hypothesis space that is filtered for the most relevant information. The different sources include:

- **Vision:** The robot's own camera is the most reliable source of information that allows the robot to compute the ball's position by itself.

- **Game Manager:** When the ball goes out of bounds, it is immediately replaced in a fixed location depending on the offending team and the quadrant of the field where the ball went out. When the game manager reports a throw-in, the robot can be sure that the ball has jumped to a new location, as the referee will have moved it.

- **Actuation:** Kicks are performed blindly as the ball is usually under the robot's camera. A large library of pre-defined kicking motions is available to each robot on the team. The robot is typically unable to visually track the ball during a kick because the ball is under the chin and behind the camera when the kick is initiated. Models of the predicted position of the ball after the kick are learned empirically in the lab [5] and used by the estimator to re-acquire the ball after a kick has been performed. Because kicks are not always successful due to noise in the interaction between the robot, ball, and the rest of the environment, we distinguish between two effects of actuation, namely: *Kick Success* and *Kick Failure*.

- **Teammate:** Teammate ball information is typically the worst source because, while their tracked local information is accurate with respect to their local reference frame, the global position can be erroneous if they are mis-localized.

The different sources of information are ranked in order of expected quality and are used to guide the behaviors to search for the ball and track it when found. The hypothesis database encodes all of the relevant domain knowledge that is necessary for segmenting the hypothesis space into the relevant subsets so that the robot can use this information to act effectively. The specific policy defined for our AIBO team for deciding which source of information to use in the hypotheses returned from the estimator is summarized in Table 2.

7 Empirical Evaluation

Estimating the state of quantities in an environment is typically done through the generation of a complex probability density function. In typical real-world problems of interest, the density functions are typically highly multi-modal and can rapidly diverge from the true estimate due to noise. Our hybrid approach to state estimation factors the complete probability density function into smaller sub-problems based on the *a priori* policy function over the problem. We have applied this approach to the challenge of robot soccer in the RoboCup 4-Legged league by first making use of a robust probabilistic algorithm for solving the underlying state estimation problem of simultaneous self-localization and tracking of the ball. Our hypothesis selection algorithm then maintains multiple independent state estimators which are created, updated, or deleted as the robot interacts with its environment and gains information from its sensors and teammates.

7.1 1D Simulation Study

To analyze the prioritized multi-hypothesis multi-object algorithm described in this paper in a statistically significant fashion, a simple one-dimensional version of the tracking problem is implemented in simulation. The simulation contains the following elements:

- A robot capable of self-locomotion, manipulation (pushing) of object, and tracking different objects one at a time. The robot uses a Kalman Filter for tracking the multiple objects.
- Several objects that exhibit stochastic motion models that can be classified as static, quasi-dynamic and dynamic. The class of motion to which each object belongs is known to the robot. All object motion is described by a noisy linear dynamical system.
- One or more "teammate" robots that can provide their own observations of objects to the primary robot. These teammates do not manipulate any objects or affect the environment in any way. Because of localization errors, the reported objects positions may be erroneous.

The task for the robot is to track the positions of all of the objects as closely as possible.

- Given:
  1. Set of hypotheses based on the game manager : H_g
  2. Set of hypotheses based on the robot's own sensors : H_r
  3. Set of hypotheses based on teammate sensors : H_t
- Select game manager, self, or teammate information
  - If H_g not empty
    - Select **game-hypothesis**
  - else if H_r not empty and H_t not empty
    - If vision data actively supports a hypothesis in H_r then select **self-hypothesis**
    - else If time_since_ball_viewed $<$ threshold then select **self-hypothesis**
    - else select **teammate-hypothesis**
  - else if H_r not empty then select **self-hypothesis**
  - else if H_t not empty then select **teammate-hypothesis**
  - else return
- Track hypothesis classes
  - If **game-hypothesis**
    - Track hypothesis H_g created by game manager
  - else if **self-hypothesis**
    - If ball is actively in view of the camera, filter self estimates with source vision and actively track the most likely one
    - else If ball is in possession, track possession estimates
    - else If ball was kicked, track the kick estimates starting with the *Kick Success* estimate and switching to the *Kick Failure* estimate when the former is pruned
    - else Track any estimates based on older vision information
  - else If **teammate-hypothesis**
    - Rank the teammate estimates based on current self-role
    - Track best ranked estimate based on teammate role and position

Table 2: Hypothesis Selection Policy for Ball Tracking in AIBO Robot Soccer

7.1.1 Multi-Object Tracking

Before introducing the concept of tracking multiple hypotheses per object, the utility of the described approach for deciding when to track a specific object is evaluated. In this experiment, three different objects with increasingly dynamic motion models are simulated. The robot's task is to maintain a good estimate of each of the three even though it is able to observe only a single object at a time. The robot is not to manipulate any objects and no teammates are present to assist the robot in the tracking problem.

The performance of the naive tracking policy is compared against the greedy tracking policy. In the naive case, the robot gives equal time to tracking each object regardless of that object's motion model and associated uncertainty. In the greedy case, the robot tracks the object with the greatest uncertainty at the time.

The simulation is run for 500 trials of 500 timesteps each. The sums of the errors between estimated position and ground truth across all three objects are computed. The average error across all trials for the greedy policy ($\mu = 19.617$, $\sigma = 1.584$) is less than the average error across all trials for the naive policy ($\mu = 22.300$, $\sigma = 2.141$). This result is statistically significant (one-tailed, two-sample t-test). Figure 3 illustrates an example run of the simulation.

7.1.2 Multi-Hypothesis Multi-Object Tracking

In this simulation, the robot's task is again to track three objects, but additionally, it must also move up to a quasi-static object and manipulate (push) it. After manipulation, the robot must re-acquire sensor contact with that object. Several hypotheses are generated after each manipulation: one which reflects a successful manipulation of the object, a second which reflects a failed manipulation, and a third which is the teammate estimate. The simulation is set up such that the actuation succeeds 90% of the time but fails 10% of the time. The physical modeling of the actuation is also corrupted by random noise. The teammate's localization estimate is corrupted by random noise as well as an offset bias which is randomized between trials to reflect the uncertainty in a teammate's localization. Because of the localization error, the positions reported by the teammate are nearly always worse than the robot's own estimates.

Several different multi-hypothesis tracking policies (shown in Table 3), which describe the order in which the hypotheses are visited by the robot's sensors, are evaluated as part of this experiment. Once again, the naive and greedy multi-object tracking policies are evaluated as part of this experiment. Figure 4 illustrates a sample run of the robot chasing the object it must actuate (for clarity, the other two objects are not shown). The simulation is run for 10,000 trials of 500 timesteps each. The sums of the errors between the estimated position and ground truth for all three objects are compared. Table 4 illustrates the results for all twelve policy configurations.

As expected, the policy configuration that performs the best over all other policy configurations is the greedy object tracking policy with hypothesis policy 1 (see Table 3 for an explanation). These results are statically significant over all other policy configurations (one-tailed, two-sample t-test).

The best hypothesis selection policy is the one which most closely matches the physics of the true environment. However, if the robot were to possess a damaged actuator which caused the actuation effect to fail more often, or if the teammate could be assured to be well localized

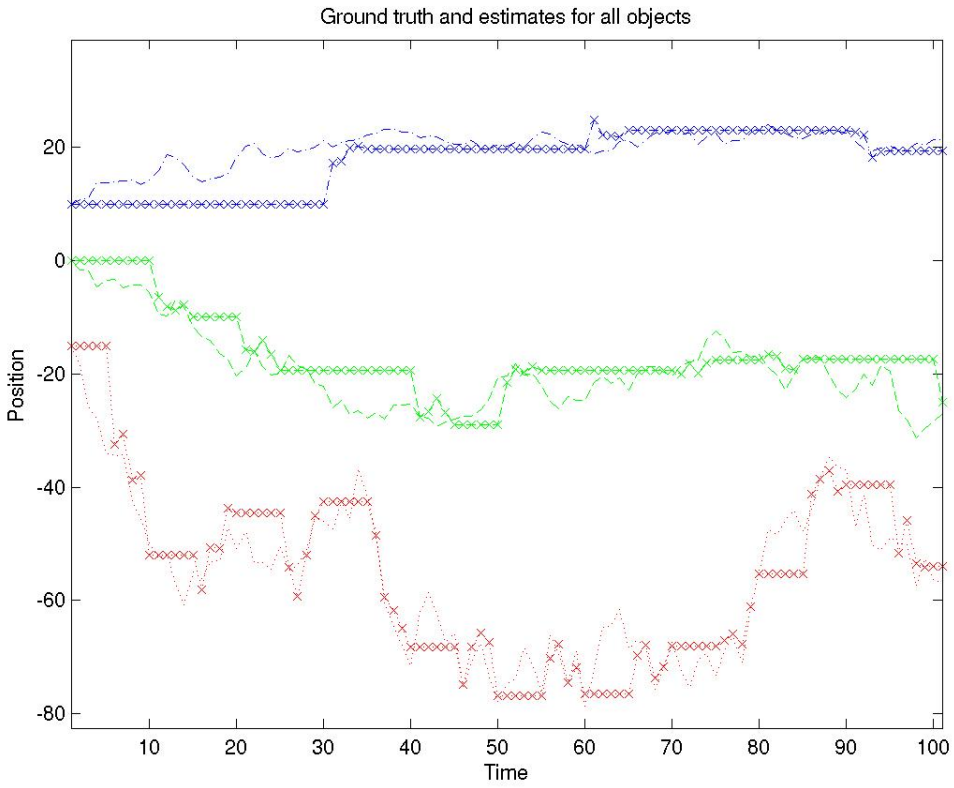


Figure 3: Example run of the one-dimensional simulation showing the positions of three objects being tracked with the greedy policy. Only 100 timesteps out of 500 are shown for clarity. For each object, 'x' marks the target's estimated position. The most dynamic object (bottom) is tracked 52% of the time, the second most dynamic (middle) is tracked 33% of the time, and the least dynamic (top) is tracked 15% of the time.

(such as a stationary teammate), policies 2 or 3 (respectively) would most likely be the superior choices. Thus, the selection of the specific hypothesis policy must be done with care after the robot's performance in its chosen environment has been observed and carefully measured.

7.2 2D Simulation Study

For initial testing of our algorithm, we have developed a robust simulation environment for running our algorithms in test soccer matches. Our simulator incorporates a basic dynamics engine allowing us to simulate the forces and accelerations applied to rigid bodies moving about (and colliding within) the environment. Actuation and sensor noise is also simulated with models measured from our real-world robots. Figure 5 shows a typical view from the simulator.

To systematically evaluate the effectiveness of the use of a high-level hypothesis policy to

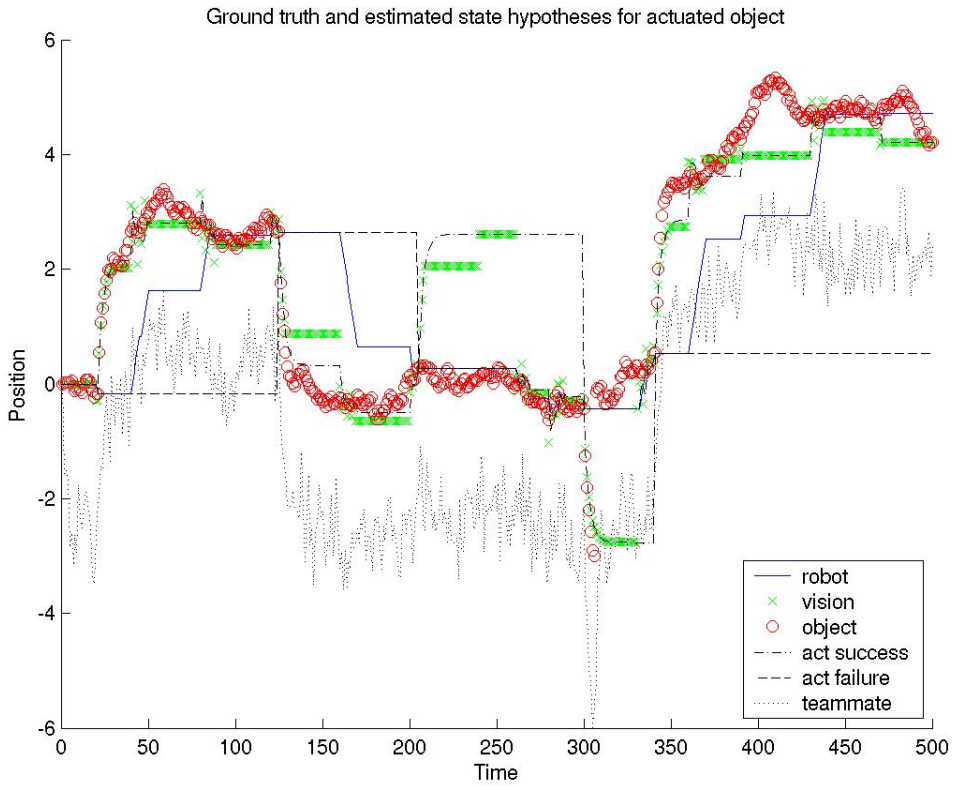


Figure 4: Example run of the one-dimensional simulation showing the robot chasing and actuating an object. After actuating the object, the robot maintains three separate hypotheses: one for actuation success, one for actuation failure, and one for the noisy external teammate observation. Actuation succeeds at times 21, 125, 300, and 341. Actuation fails at time 205. The object is moved by an external force at time 307 and the robot must use the teammate observation to re-localize it. Not shown are the other two objects that the robot is tracking.

factor a probabilistic state estimation problem into a more tractable form, several hundred robotic runs were performed with our simulation package. The underlying estimation algorithm used in this study was a Rao-Blackwellized particle filter (RBPF), similar to the algorithms reported in [12] and in [9], where the state of the ball as well as motion model of the ball are stochastically sampled based on the expected activities of the robot and its sensor information.

In these experiments, single robot kicks the ball between several different waypoints on the field. The robot's sensor readings as well as the actuation models are stochastically corrupted with noise. We compared the performance of the RBPF which estimated the full state of the ball and motion model against our hybrid approach where each motion model is given its own independent state estimate (also using a RBPF for each) and the robot chooses which model to track based on its actions and expected performances therein.

Policy	1st Hyp.	2nd Hyp.	3rd Hyp.
1	successful push	failed push	teammate
2	failed push	successful push	teammate
3	teammate	successful push	failed push
4	successful push	N/A	N/A
5	failed push	N/A	N/A
6	teammate	N/A	N/A

Table 3: Six multi-hypothesis tracking policies tested in simulation. Policies 4-6 only ever track a single hypothesis.

Hyp. Policy	Targ. Policy	μ	σ
1	Greedy	1.667	1.644
1	Naive	1.819	1.844
2	Greedy	1.891	1.731
6	Naive	2.028	2.226
3	Greedy	2.054	2.170
3	Naive	2.067	2.240
2	Naive	2.086	2.098
6	Greedy	2.105	2.160
4	Naive	2.122	2.400
4	Greedy	2.164	2.360
5	Naive	2.574	2.912
5	Greedy	2.648	2.764

Table 4: mean error and std dev for all 12 cases evaluated over 10,000 trials in simulation. The policy combinations are sorted with the least error on top and the largest error on bottom. The first column represents the hypothesis selection policy, as described in Table 3, and the second column represents the target tracking policy.

The ground truth of the ball's position on the field was recorded and compared to the robot's current estimate. After several hundred experiments, we found that the hybrid policy estimator outperformed the single estimator in a statistically significant fashion (0.732m error on average for the single RBPF vs 0.592m on average for the policy selection algorithm). We note that the parameters for all of the RBPFs were kept the same for both experiments. Figure 6 illustrates an example estimate from both approaches. Note in Figure 6(b) how the density from the policy selection algorithm is focused mainly around the areas from the expected outcomes of the actions where the density of the particles in Figure 6(a) is more spread out.

A series of simulation experiments were performed to evaluate the effectiveness of the policy selection algorithm on a variety of different environment. In this study, a single robot was required to find the soccer ball on the field, and manipulate it with its kicking mechanism through a series of waypoints. As with the 1D simulation study, a number of different policies were evaluated for

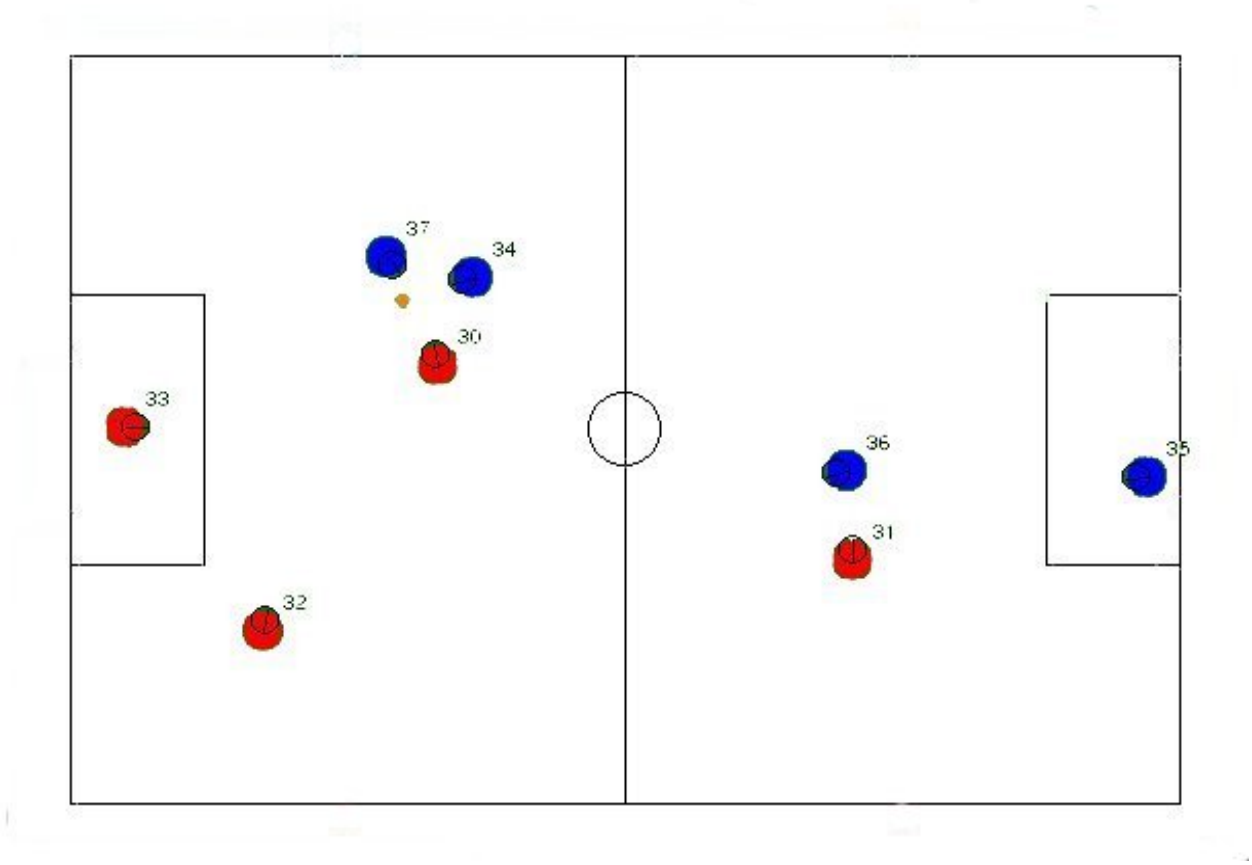


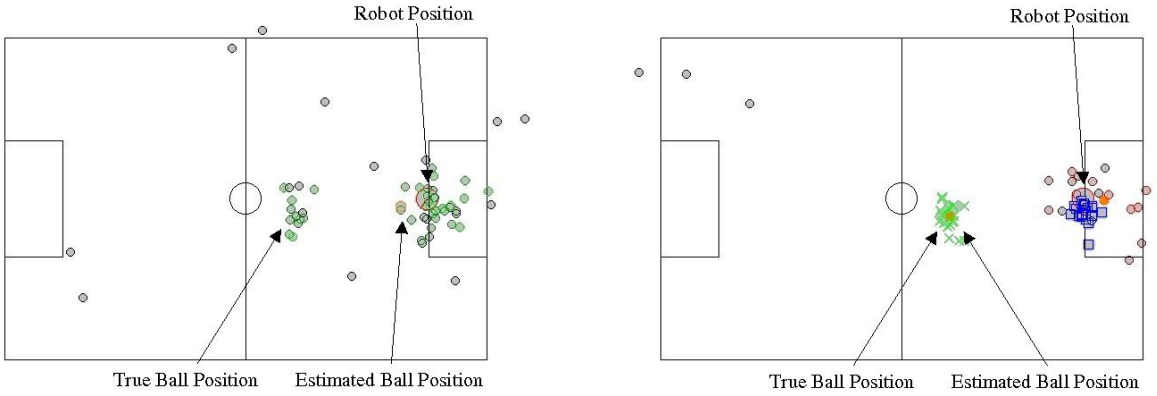
Figure 5: Simulation package used for evaluation of robot soccer algorithms.

where the robot should task its sensors in order to find the ball when it was not in view of the robot's sensors. Table 5 illustrates the list of different policies.

In the simulation study, four different environmental cases were studied. These included several different environmental cases that we have observed in real RoboCup soccer matches:

- Case A: High probability of kick success with bad teammate localization
- Case B: Low probability kick success with bad teammate loc localization
- Case C: High probability kick success with good teammate loc localization
- Case D: Low probability kick success with good teammate localization

The kick success is directly affected by the state of the environment whereby the texture, friction, and dampening of the soccer field will directly affect how well the kicking action works. The effects of teammate localization are also heavily dependent on the state of the lighting in the environment.



(a) (b)

Figure 6: Example illustration of the simulated world where the robot has just kicked the ball. The vanilla RBPF estimator tracking the ball (a). The hybrid bank of RBPF estimators tracking the ball in (b). In (b), the three different hypotheses are represented as different shaped particles (vision=circle, kick_success=cross, kick_failure=square).

Policy	1st Hyp.	2nd Hyp.	3rd Hyp.
1	successful kick	failed kick	teammate
2	failed kick	successful kick	teammate
3	teammate	successful kick	failed kick

Table 5: The multi-hypothesis tracking policies tested in 2D simulation.

Each trial of the simulation consisted of the robot approaching, grabbing, and kicking the ball such that it could manipulate it through a series of waypoints on the field continuously for 10 minutes. A stationary teammate robot tracked the ball and relayed its observations when the ball was in view. The results were evaluated on how well the kicking robot's state estimate matched the global ground truth of the world and by how much time the robot actually had the ball in its view. The results are summarized in Table 6 for the error in the robot's ball estimates and Table 7 for the amount of time that the robot had the ball in view of its sensors.

In general, the appropriate policies performed well in the environmental conditions where they were placed. We did not expect that the outcomes of the different policies would have the same ranking in performance for both the average error in the estimated ball position as well as the average time that the ball was visible in the robot's sensors. However, when looking at the results, it can be seen that the rankings of the 3 policies are the same for both metrics.

In cases A & B, the teammate robot was unable to localize itself very well and as a result, the policy that made use of that information first performed the most poorly in those cases. However, in cases C & D, the opposite was true. The challenge must be faced by any team of robots is to decide

Case A	Error in meters		Case B	Error in meters		Case C	Error in meters		Case D	Error in meters	
Policy	μ	σ	Policy	μ	σ	Policy	μ	σ	Policy	μ	σ
1	0.58	0.12	2	1.06	0.08	3	0.28	0.05	3	0.16	0.04
2	0.85	0.07	1	1.17	0.01	1	0.48	0.14	2	0.59	0.03
3	1.79	0.05	3	1.73	0.05	2	0.74	0.06	1	0.75	0.03

Table 6: Results of the 2D simulated soccer simulation experiment showing a list of the policies ordered from best (top) to worse (bottom) based on the estimator error for the four different experimental cases. Every result is statistically significant (based on t-test).

Case A	Frames visible		Case B	Frames visible		Case C	Frames visible		Case D	Frames visible	
Policy	μ	σ	Policy	μ	σ	Policy	μ	σ	Policy	μ	σ
1	1087	48.08	2	568	50.80	3	1248	19.30	3	965	51.96
2	925	27.26	1	477	68.50	1	1167	47.42	2	714	30.99
3	866	59.19	3	442	43.98	2	981	40.84	1	633	29.9

Table 7: Results of the 2D simulated soccer simulation experiment showing a list of the policies ordered from best (top) to worse (bottom) based on the time the ball is in view for the four different experimental cases. Time is measured in frames of video where the ball is visible (frame rate is 33Hz). Every result is statistically significant (based on t-test), except for the times of policies 1 and 3 in case B.

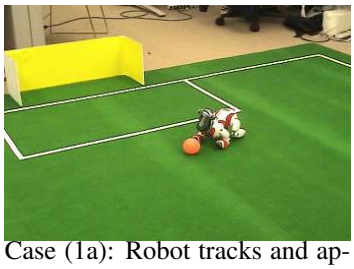
when to trust the information returned by their teammates. Teammate information, particularly in the RoboCup environment where individual robots are crowded and jostled by opponents, can very easily be corrupted without the teammate being aware of it until it attempted to re-localize itself. When the teammate's position is corrupted with error any information about tracked objects that are converted from the robot's egocentric coordinate systems to a global coordinate system will also be corrupted. This is a very serious problem because in addition to translational error, any error in the orientation will generate a significant additional error in the global pose of the object.

The only results that were not statistically significant were the times that the ball was visible in case B for policies 1 & 3. Case B was probably the hardest for the robot because its kicking actions were the most likely to fail and the teammate's reported ball position was very error-prone. Thus, if the robot did not first look to the kick failure hypothesis first, it would spend a lot of time chasing phantoms in either of those two cases.

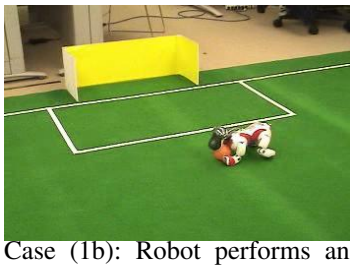
7.3 Real World Study

We have implemented our hybrid policy selection algorithm on our AIBO RoboCup team where the robots and algorithm have performed (and won) in competition. In the AIBO implementation, the underlying probabilistic state estimation algorithm for tracking the ball is a simplified Multi-Hypothesis Tracker using an Extended Kalman Filter. The deciding factor for the choice of this estimator was the need for computational efficiency on a very limited CPU budget. Other algo-

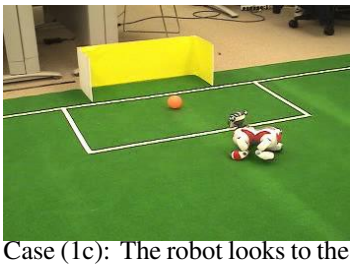
rithms, such as the computer vision and self-localization, require a large percentage of the available computation as well. Our hybrid hypothesis selection algorithm was implemented as described in the previous section. Each hypothesis estimate is allowed one or two Kalman Filters (merging or splitting as needed). An example of the hypothesis selection policy, as implemented on our AIBO robots, is illustrated on a simple example in Figure 7.



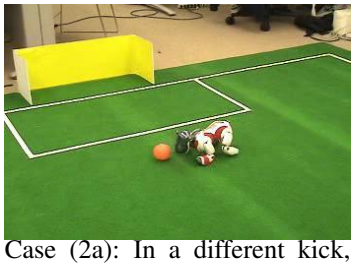
Case (1a): Robot tracks and approaches the ball for a kick.



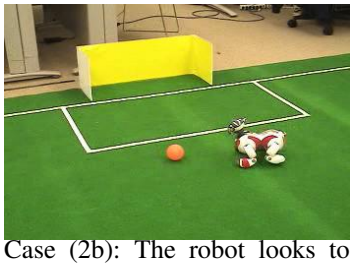
Case (1b): Robot performs an open-loop grab motion and kick which succeeds.



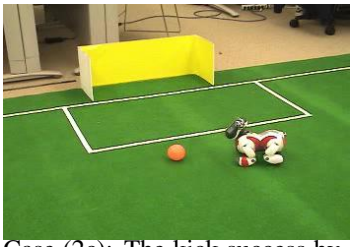
Case (1c): The robot looks to the location of the kick success hypothesis for the ball.



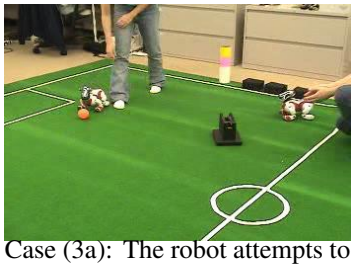
Case (2a): In a different kick, the open-loop grab and kick fails.



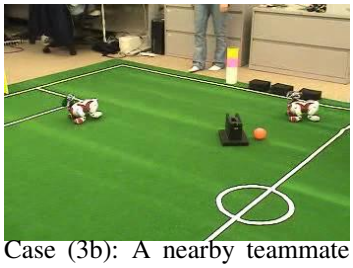
Case (2b): The robot looks to the location of the kick success hypothesis for the ball but doesn't find it.



Case (2c): The kick success hypothesis is pruned and the robot looks to the kick failure hypothesis.



Case (3a): The robot attempts to kick, but the ball is stolen by external forces.



Case (3b): A nearby teammate is shown the ball. Both the kick success and kick failure hypotheses are evaluated.

Case (3c): All of the kick hypotheses expire and the robot tracks the teammate's reported hypothesis (the ball is hidden from the first robot's view.)

Figure 7: An illustrative example of the prioritized Multiple Hypothesis algorithm for reasoning about possible locations for the ball.

The Kalman filter [11] is a Bayesian filtering algorithm which estimates the state of a system by modeling the process and sensor noise with zero-mean univariate Gaussian distributions. The Kalman filter estimates a quantity with a *propagation* step whereby the predicted state of the sys-

tem is computed according to a dynamics model, and a sensor *update* step, where a (noisy) sensor reading model corrects the predicted state. In both steps, the state estimate and the uncertainty associated with the state are updated. However, a shortcoming of the basic Kalman filter algorithm is that it assumes that all of the noise models can be estimated using white Gaussian noise. Additionally, the final state and uncertainty estimate are also represented as a single Gaussian distribution. Thus, our approach uses a variation on the Multiple Hypothesis Tracker (MHT) [16] Kalman filter algorithm where a multi-modal probability density is estimated by a bank of Kalman filters.

Interestingly enough, the deterministic approximation to the state estimation problem solved by the MHT paradigm can be considered analogous to approximate inference methods for performing stochastic inference in Switching Kalman Filter models via a Rao-Blackwellized particle filter [15]. At this time, it is not clear whether one approach is superior to the other. For efficiency purposes, the AIBOs use the Kalman filter to generate a probabilistic estimate for the position of the ball. Particle filters typically require greater computational power due to the large number of samples that must be maintained and updated. While significant for a robot its size, the AIBO's on-board computer, a 600MHz MIPS processor, must handle a great deal of additional processing, such as vision, localization (already using a particle filter), and kinematics. Computational issues of the robot aside, we assert that our proposed hypothesis selection algorithm is independent of the particular representation used for the state estimates. Instead of Kalman Filters for each estimate, independent sets of particle filters could be used to represent the different hypothesis classes. By keeping them disjoint, the robot can select the appropriate hypothesis to explore using the proposed algorithm.

Each element of the disjoint state estimate is represented using a bank of L Kalman filters. In this way, a multi-modal estimate generated by multiple potentially conflicting or ambiguous sensor readings can be maintained until additional sensor information removes one or more hypotheses that are inconsistent with new sensor data. When new sensor data arrives, a gating function is used to determine which filter should be updated with the new information. If no hypothesis matches the data, a new hypothesis will be initialized. All hypotheses have an uncertainty model which is represented as a covariance matrix P. As per the propagation algorithm, the uncertainty of the covariance matrix will continuously grow if there is no sensor data. Eventually, a check is performed to determine whether the covariance of the estimate has grown too large to be practical. In this case, a particular filter is no longer informative (essentially a uniform density distribution) and is removed from consideration.

The sources of information that feed into this estimator can have distinctly different process models which describe how quickly the uncertainty grows in the model. Our approach makes use of this in order to exploit both positive and negative information returned from the sensors to adapt the process noise of the estimates. When the tracked object is observed by the sensors, the process noise is set to a model which best describes the dynamics of that object. Specific estimate process noise is based on the following states and is ranked from lowest (1) to highest (5):

1. **Visible:** The estimate is being actively observed and tracked with the camera.

2. **Possession:** The estimate is not seen, but the robot believes that the object is under its chin and can be manipulated.

$\hat{X}_{t/t}$	Estimated state at time t with accumulated sensor readings from time t
$\hat{X}_{t/t+1}$	Estimated state at time $t+1$ with accumulated sensor readings from time t. This occurs when the system dynamics are propagated but no sensor reading has yet been obtained at time $t+1$
F	State transition Jacobian matrix
B	Control matrix
u_t	Control input at time t
$P_{t/t}$	Covariance matrix at time t with accumulated sensor readings from time t
$P_{t/t+1}$	Covariance matrix at time $t+1$ with accumulated sensor readings from time t. See definition of $\hat{X}_{t/t+1}$ above.
G	Process noise Jacobian matrix
Q_t	Process noise covariance matrix at time t
z_t	Sensor reading at time t
$\hat{z}_t$	Estimated value of sensor reading at time t
H	Sensor Jacobian matrix
r_t	Residual between expected and actual sensor readings
R	Sensor noise covariance
S_t	Computed covariance of sensor reading at time t
K_t	Kalman gain at time t

Table 8: Description of terms used in the Multiple Hypothesis Kalman filter algorithm

3. **Not in camera view:** The estimate is not within the expected field of view of the camera as the robot's sensors have been directed elsewhere.

4. **In camera view but occluded:** The estimate is expected to be visible in the calculated camera view but currently is not. However, occluding objects (such as other robots) are also present in the image, so the object could still be present.

5. **In camera view but not visible:** The estimate is expected to be visible in the calculated camera view but it is not. No additional occluding objects are present.

As the process noise increases, the estimate uncertainty will increase and decrease the likelihood that it will be selected as the next hypothesis to explore by the robot. Thus, when the sensors view an area where the tracked object is *expected*, but no readings are found, the process noise increases drastically to reflect the notion that the object has moved.

The specific notation for these algorithms is described in Table 8. The propagation algorithm for our disjoint Multiple Hypothesis tracker is shown Table 9, and the sensor update algorithm is shown in Table 10.

Directly evaluating this algorithm on real robots is much more difficult due to the challenge of obtaining the ground truth of the ball and the robot in the environment. However, we have conducted controlled experiments where we have measured the time that it takes for the robot to maintain visual contact with the ball with the policy algorithm vs. a straight estimator with a naive search. The mean times for visually re-acquiring the ball after losing track of it are statistically

Multiple Hypothesis Kalman Filter propagation algorithm

- Given:
  1. A list of Kalman filters L
  2. Sensor poses and expected fields of view
- **Propagate**
  - For each filter l (with state estimate $\hat{X}_{t/t}$ and covariance matrix $P_{t/t}$) **do**
    - Propagate the state and covariance matrix from time $t-1$ to time t

$$\begin{aligned} \hat{X}_{t+1/t} &= F\hat{X}_{t/t} + Bu_t \\ P_{t+1/t} &= FP_{t/t}F^T + GQ_tG^T \end{aligned}$$

    - Update the process noise matrix Q of the filter based on the expected readings of the sensor
    - If likelihood of $P_{t+1/t}$ is less than a threshold, delete filter l from the list

Table 9: Multiple Hypothesis state estimation propagation algorithm

significant on the order of several seconds. This time to re-acquire the ball is even more significant when dealing with reported teammate estimates. Due to the difficulty of localizing the robot in the dynamic RoboCup environment, teammates can potentially broadcast very inaccurate information. In actual competition games where the teammate information was given higher priority, the robots tended to be more lost than in games when they used their own models first before listening to teammates.

Figure 8 illustrates how the multiple disjoint hypothesis tracking algorithm steps through the different hypothesis classes in an attempt to drive the robot towards the correct estimated ball position. In this example, two AIBOs are tracking two different balls on the field. The AIBO in the center is actively attempting to score a goal with its ball. The stationary AIBO in the upper right corner of the field tracks a ball that is occluded from the first AIBO. The moving AIBO continuously receives a global position estimate for the ball from the stationary one.

In Figure 8(a), an AIBO observes the ball on and moves towards it in an attempt to kick it into the goal. In Figure 8(b), the robot performs a side kick that uses its head and the current hypothesis changes to the class of kicks and is split into two cases. The first case is the success case, which models the kinematics of the kick and predicts the motion. The second case is a failure case which models the situation where the AIBO failed to kick the ball. The kick success case is initially higher priority, and so the robot attempts to track its position. The kick success hypothesis estimates the ball's new position at each timestep by modeling the velocity of the ball after the kick, as shown in Figure 8(c). Because the robot missed the ball, the successful kick hypothesis is not valid and when the robot aims its camera towards it, no ball is observed. This negative information

Multiple Hypothesis Kalman Filter sensor update algorithm

- Given:
  1. A list of Kalman filters L
  2. Reading from a sensor z
- **Update**
  - If Kalman filter list L is empty, initialize a new filter based on the sensor reading and exit
  - else For each Kalman filter l **do**
    * Compute the Mahalanobis distance for the filter l and the sensor reading

$$\begin{aligned} \hat{z}_{t+1} &= H\hat{X}_{t+1/t} \\ r_{t+1} &= z_{t+1} - \hat{z}_{t+1} \\ S_{t+1} &= HP_{t+1/t}H^T + R \\ M &= r_{t+1}S_{t+1}^{-1}r_{t+1}^T \end{aligned}$$

  - Select the Kalman filter l_i with the smallest Mahalanobis distance M_i
  - If $M_i <= thresh$, apply the sensor estimate to that filter

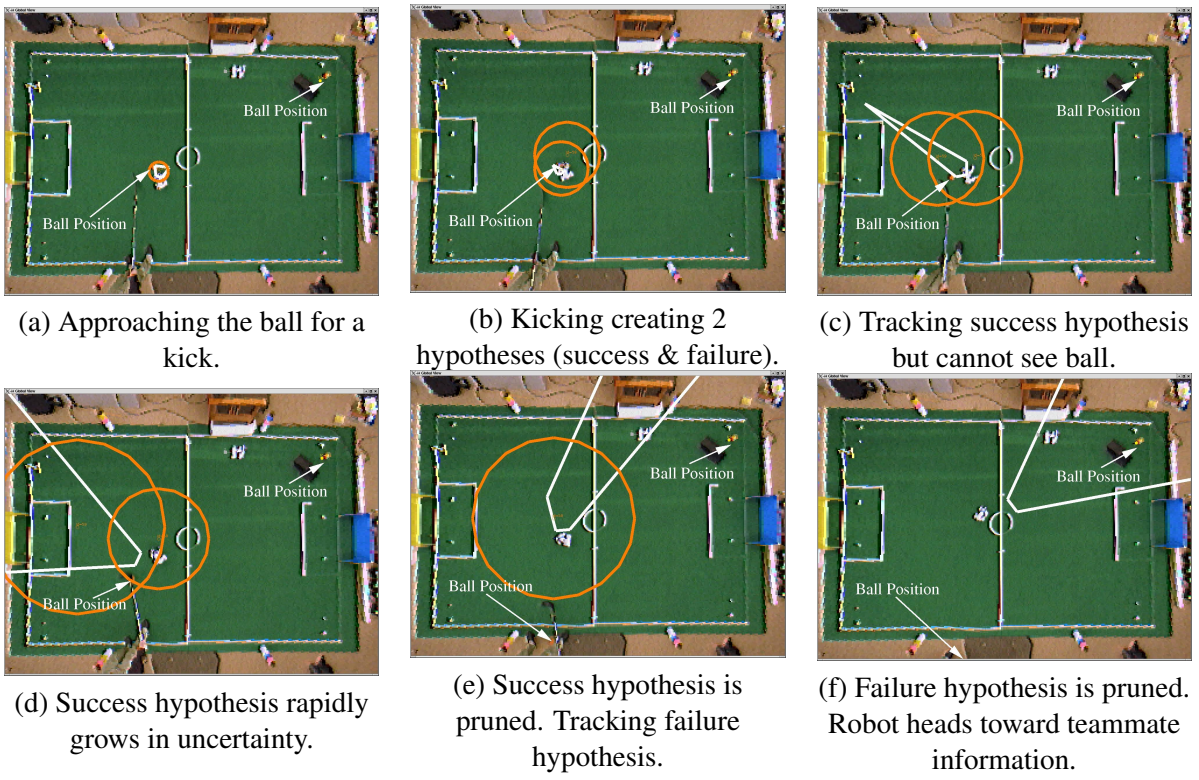
$$\begin{aligned} K_{t+1} &= P_{t+1/t}H^T S_{t+1}^{-1} \\ \hat{X}_{t+1/t+1} &= \hat{X}_{t+1/t} + K_{t+1}r_{t+1} \\ P_{t+1/t+1} &= P_{t+1/t} - P_{t+1/t}H_{t+1}^T S_{t+1}^{-1} H_{t+1} P_{t+1/t} \end{aligned}$$

  - else Initialize a new filter based on the sensor reading and add it to the list.

Table 10: Multiple Hypothesis state estimation sensor update algorithm

greatly increases the process noise of the kick success model and the uncertainty grows quickly, as shown in Figure 8(d). The success hypothesis quickly expires and the robot brings its attention to the kick failure hypothesis. As shown in Figure 8(e), once again, because the ball is not there, the negative information causes the uncertainty of that hypothesis to grow until it expires as well. In Figure 8(f), the kick failure hypothesis also expires and the robot finally uses the teammate observations to direct its motion to the upper corner of the field.

We have conducted controlled experiments on the real robots where we have measured the time that it takes for the robot to maintain visual contact with the ball with our proposed search policy algorithm vs. a standard MHT-EKF state estimator coupled with a naive search. We have run a set of experiments to compare the performance of the two different estimators. The naive search is considered to be a policy where the robot always tracks the estimated position of the ball



(a) Approaching the ball for a kick.

(b) Kicking creating 2 hypotheses (success & failure).

(c) Tracking success hypothesis but cannot see ball.

(d) Success hypothesis rapidly grows in uncertainty.

(e) Success hypothesis is pruned. Tracking failure hypothesis.

(f) Failure hypothesis is pruned. Robot heads toward teammate information.

Figure 8: A top-down view of the multi-hypothesis algorithm running on the robot which demonstrates how the algorithm directs the robot's actions. The field of view of the robot's camera is shown with white lines while the circles represent the uncertainty in the tracked object position.

assuming a successful kick. In both cases, after the tracked hypothesis expires due to exceeding an uncertainty threshold, the robot will revert back to a more expensive generic ball search which constitutes spinning in place while scanning its camera to exhaustively search at different distances. In a real game situation, minimizing the time to find the ball is critical as the longer the robots search for the ball, the greater the chance that the other team will find and control the ball.

In these experiments, similar to the simulation experiments, the robots were required to locate the ball and kick it to a specific position on the field. This emulates normal game behavior where the robots will attempt to move the ball up the field (and potentially near teammates). The time between when each kick was performed and the ball was re-acquired by the vision system was recorded. A set of ten experiments was performed where the times to find the ball after the first ten successful kicks as well as the first ten failed kicks were recorded. Kicks can fail due to misalignment of the ball to the robot's head and legs. In a game situation, this happens very frequently due to the robot being jostled by opponent robots. The results are illustrated in Table 11.

On an empty field, the kicks fail approximately 10% of the time. However, as mentioned previously, when jostled in a real game situation, this kick failure can be in excess of 50% and is often much higher in crowded situations. Thus, it is very important that the robots reason effectively about the potential outcomes of their actions at a high level in order to more rapidly re-acquire the

	PMHWM EKF	Simple EKF
Successful kick	$\mu = 0.98, \sigma = 0.01$	$\mu = 0.97, \sigma = 0.01$
Failed kick	$\mu = 3.07, \sigma = 0.42$	$\mu = 5.00, \sigma = 0.52$

Table 11: Mean times (in seconds) and standard deviations to find the ball after successful and unsuccessful kicks. Finding the ball after a successful kick was statistically the same time while finding the ball after an unsuccessful kick was statistically much faster with the PMHWM approach.

target.

This time to re-acquire the ball can be even more significant when dealing with reported teammate estimates. Due to the difficulty of localizing the robot in the dynamic RoboCup environment, teammates can potentially broadcast very inaccurate information. In actual competition games where the teammate information was given higher priority, the robots tended to be more lost than in games when they used their own models first before listening to teammates. Because the robots have no sense of touch, if they are knocked off course due to a collision, they do not know that their localization estimate has become inaccurate until they attempt to re-localize themselves based on the nearly markers. However, as all robot are attempting to track the position of the ball as much as possible, teammates can potentially transmit very poor ball information for long periods of time.

8 Summary and Conclusions

In this paper, we describe an approach for multi-hypothesis state estimation in a dynamic environment where a robot must contend with uncertain sensor readings and incomplete models of objects. We formally describe the problem of tracking objects with multiple hypotheses based on models and other information sources. For any environment of reasonable complexity, a robot is incapable of simultaneously tracking all objects of interest with its sensors as the scope of the robot's sensor field is simply too narrow (or otherwise limited) compared to the size of the environment.

Probabilistic state estimates are powerful mechanisms for representing the uncertainty in a robot's state estimate. However, due to the multi-modal nature and potentially high dimensionality of these estimates, estimating the complete density function can be exceedingly challenging, particularly when the noise models are not known exactly. More importantly, in many applications, maintaining an accurate estimating of the density is not as important as choosing an action quickly in a dynamic environment. We believe that the fusion of a high-level policy-based approach with effective probabilistic state estimation algorithms will allow robots to maintain better estimates of their world by combining effective action selection with robust state estimation.

We describe a mechanism by which the robot can intelligently decide how best to aim its sensors to maintain an accurate estimate of the state of all objects. When an object is not in view, its position must be predicted from analyzing multiple sources of model information that can include models for the success or failure of actuation as well as external observations from teammates. We describe a formal policy mechanism by which the robot can select appropriately

among multiple hypotheses based on domain information in order to augment a traditional state estimation algorithm to allow the robot quickly re-acquire the object. Deciding on the correct policy for the robot can be done *a priori* and can rapidly be changed if the situation warrants. Our approach was developed for and successfully applied to several real multi-robot systems. We have validated it through an extensive empirical simulation study and have used it successfully in competition on our real robots. Our current work is to analyze how we can learn these policies in real-time on the robots as they perform their tasks.

Acknowledgements

We would like to thank the members of the CMDash'05 and CMDash'06 RoboCup 4-Legged league team for their help and support. In particular, we'd like to thank Sonia Chernova and Colin McMillen for their contributions to individual robot behaviors and team coordination, respectively.

References

[1] Steven Abrams, Peter Allen, and Konstantinos Tarabanis. Dynamic sensor planning. In *Proceedings of the International Conference on Robotics and Automation*, 1993.

[2] Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for on-line non-linear / non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[3] Y. Bar-Shalom, K. C. Chang, and H. A. P. Blom. Tracking a maneuvering target using input estimation vs. the interacting multiple model algorithm. *IEEE Transactions on Aerospace and Electronic Systems*, 25:296–300, March 1989.

[4] Y. Bar-Shalom and K. Kirmiwal. Variable dimension filter for maneuvering target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 18(5):621–629, September 1982.

[5] Sonia Chernova and Manuela Veloso. Learning and using models of kicking motions for legged robots. In *In Proceedings of the International Conference on Robotics and Automation (ICRA'04)*, New Orleans, LA, May 2004.

[6] I. J. Cox and S. L. Hingorani. An efficient implementation and evaluation of reid's multiple hypothesis tracking algorithm for visual tracking. In *International Conference on Pattern Recognition*, pages 437–442, 1994.

[7] H. Durrant-Whyte. Sensor models and multisensor integration. *The International Journal of Robotics Research*, 7:97–113, December 1988.

[8] Dieter Fox. Adapting the sample size in particle filters through kld-sampling. *International Journal of Robotics Research (IJRR)*, 22, 2003.

[9] Yang Gu. Tactic-based motion modeling and multi-sensor tracking. In *In Proceedings of the American Association for Artificial Intelligence (AAAI)*, pages 1274–1279, Pittsburgh, PA, July 2005.

[10] A.M. Jazwinsky. *Stochastic Processes and Filtering Theory*. Academic, New York, 1970.

[11] R. E. Kalman and R. Bucy. New results in linear filtering and prediction theory. *Transactions of ASME, Journal of Basic Engineering*, 83:95–108, 1961.

[12] Cody Kwok and Dieter Fox. Map-based multiple model tracking of a moving object. In *In Proceedings of the RoboCup 2004 Symposium*, Lisbon, Portugal, July 2004.

[13] Noriaki Mitsunaga, Taku Izumi, and Minoru Asada. Cooperative behavior based on a subjective map with shared information in a dynamic environment. In *In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 291–296, Las Vegas, October 2003.

[14] Kevin Murphy. Learning switching kalman filter models. Technical Report 98-10, Compaq Cambridge Research Lab, 1998.

[15] Kevin Murphy. *Dynamic Bayesian Networks: Represenation, Inference, and Learning*. PhD thesis, University of California, Berkeley, 2002.

[16] D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transaction on Automatic Control*, 24(6):843–854, December 1979.

[17] Maayan Roth, Douglas Vail, and Manuela Veloso. A real-time world model for multi-robot teams with high-latency communication. In *In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2494–2499, Las Vegas, NV, October 2004.

[18] Stergios I. Roumeliotis and George Bekey. Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. In *In Proceedings of the International Conference on Robotics and Automation (ICRA'00)*, San Francisco, CA, April 2000.

[19] Sebastian Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.