From Pixels to Multi-Robot Decision-Making: A Study in Uncertainty

Peter Stone, Mohan Sridharan, Daniel Stronger, Gregory Kuhlmann [1,2,1,1]
Nate Kohl, Peggy Fidelman, Nicholas K. Jong [1,1,1]

The University of Texas at Austin
Austin, TX 78712

Abstract

Mobile robots must cope with uncertainty from many sources along the path from interpreting raw sensor inputs to behavior selection to execution of the resulting primitive actions. This article identifies several such sources and introduces methods for i) reducing uncertainty and ii) making decisions in the face of uncertainty. We present a complete vision-based robotic system that includes several algorithms for learning models that are useful and necessary for planning, and then place particular emphasis on the planning and decision-making capabilities of the robot. Specifically, we present models for autonomous color calibration, autonomous sensor and actuator modeling, and an adaptation of particle filtering for improved localization on legged robots. These contributions enable effective planning under uncertainty for robots engaged in goal-oriented behavior within a dynamic, collaborative and adversarial environment. Each of our algorithms is fully implemented and tested on a commercial off-the-shelf vision-based quadruped robot.

Key words: Robotics, planning under uncertainty, robot vision, localization, multi-robot behavior

1 Introduction

Autonomous robots encounter uncertainty in many forms along the path from interpreting their raw sensor data to generating executable actions. There is uncertainty manifest in decoding inevitably noisy sensor readings; there is uncertainty in the effects of the robots' actions, both past and future; and (in part as a result of the former) there is uncertainty reflected in the robot's tracking of the world state. In multi-robot settings, there is further uncertainty in the world knowledge of the other robots, both with regard to their

[1] Department of Computer Sciences
[2] Department of Electrical and Computer Engineering

relative accuracies, and with regard to consistency among the robots. However, in a team setting, teammates may be able to mitigate this uncertainty by communicating with one another.

Robots must plan their actions in spite of all this uncertainty, and in some cases may select actions specifically to reduce uncertainty. This article identifies methods for coping with uncertainty towards action planning on a vision-based mobile robot. Specifically, on such a robot, there is uncertainty in:

- the colors of the objects observed by the robot, for example as a result of changing illumination conditions;
- the robot's sensor model, for example mapping the observed height of an object (in pixels) to its distance;
- the robot's action model, for example determining how quickly it moves as a result of its specific movement actions; and
- the robot's location in its environment.

We summarize our novel algorithms for dealing with each of these forms of uncertainty, initially on individual robots, and ultimately taking advantage of collaborative multi-robot interactions. Specifically, we present algorithms for autonomous color calibration, illumination invariance, autonomous sensor and actuator modeling, and an adaptation of particle filtering for improved localization on legged robots. Each of these technical contributions comes from a detailed and independent research thread. This article synthesizes them within the context of creating models for robot planning under uncertainty.

Collectively, these algorithms produce models that are necessary for effective planning under uncertainty for robots engaged in goal-oriented behavior within a dynamic, collaborative and adversarial environment. We then place particular emphasis on the ways in which our robots reach action decisions based on these models. In particular, we focus on the robots' ability to i) interleave planning, action, and information-gathering; ii) execute consistent actions over time; iii) behave reactively when appropriate and iv) share and merge local perceptual information among teammates as a way of accurately tracking the world state.

As the concrete substrate for our research, we implement all of our contributions on a team of commercial off-the-shelf robots, namely Sony ERS-7 Aibo robots. Each contribution is validated individually in a controlled setting. Taken together, some of these contributions enable effective execution of the RoboCup robot soccer task following the rules of the four-legged soccer league, while the others are forward-looking towards operating in more uncontrolled environments.

The remainder of this article is organized as follows. Section 2 provides some background information on the test platform and the application domain.

Sections 3, 4, and 5 describe our methods for reducing the uncertainty at the pixel level, in the action and sensor models, and in the robot's position, respectively. Then, Section 6 presents the methods incorporated to account for the uncertainty in planning and decision-making. Finally, we briefly discuss some related approaches in Section 7 and conclude in Section 8.

2 Background

Our focus is on developing efficient algorithms for reasoning under uncertainty in task-oriented scenarios. One such scenario is the RoboCup Robot Soccer Legged League [3] in which teams of fully autonomous robotic dogs manufactured by SONY play a game of soccer on an indoor field.

In our experiments, we used the standard Legged League robot, the Sony Aibo ERS-7 [1]. It is equipped with a CMOS color camera at the tip of its nose with a horizontal field-of-view of 56.9^o and a vertical field-of-view of 45.2^o, providing the robot with a limited view of its environment from which it has to extract the information needed for decision-making. The images are captured in the $YCbCr$ format at 30Hz and image resolution of 208×160 pixels. It has 20 degrees of freedom: 3 in its head, 3 in each leg, and 5 more in its mouth, ears and tail. It also has noisy touch sensors, IR sensors, and a wireless LAN card for inter-robot communication. All processing – for vision, localization, locomotion, and action-selection – is performed on-board the robot, using a 576MHz processor.

RoboCup Legged League games are played on a 4m $\times$ 6m green carpet pitch with white field lines, color-coded goals, and four color-coded cylindrical beacons used for localization. Additionally, the robot is able to perceive the orange ball and red or blue uniforms worn by the robot teams. As a result, some of the perceptual algorithms presented here are specific to color-coded environments. However, their uses for localization and decision-making generalize to any perceptual system capable of identifying objects.

Currently, RoboCup games are played under constant and reasonably uniform lighting conditions, but one research challenge is to enable the robots to play under varying illumination conditions. [4] Our team has participated in both the national (US-Open) and the international robot soccer competitions for the last three years and has consistently ranked among the top teams.

[3] http://www.tzi.de/4legged

[4] The stated ultimate goal of the RoboCup initiative is to create a team of humanoid robots that can beat the human soccer champions by the year 2050 on a real, outdoor soccer field [2].

3 Uncertain Object Colors

The first step towards planning on a mobile robot is gathering world state information. On a vision-based robot, interpreting raw sensor data is a formidable challenge. Furthermore, most of the previous work in machine vision assumes a stationary camera and/or relatively unconstrained computational resources. In contrast, the algorithms on vision-based robots must work within the constraints of their on-board processing capabilities, and be robust to mobile cameras.

Color is often (though certainly not always) one of the most informative visual cues in the environment. However, color segmentation is an inherently uncertain operation due to the fact that there are more pixel values than can possibly be labeled manually, thereby requiring error-prone generalization that is often brittle. Furthermore, under changing illumination conditions, the same pixel values may represent different colors. Finally, finding the desired objects in environments with other objects of similar colors can be particularly challenging. These difficulties in vision processing are addressed in this section. We begin with an overview of our baseline vision system, and then present our two approaches to mitigating the vast amount of uncertainty in vision: autonomous color calibration and a method aimed at directly achieving illumination invariance.

Our baseline vision system consists of two main components: color segmentation and object recognition.

First, in the color segmentation phase, the robot maps each pixel in the raw *YCbCr* input image to a color class label (one of nine different colors in our domain). To reduce the memory requirements, instead of generating this mapping for all possible *(Y, Cb, Cr)* combinations (0–255 along each dimension), we subsample the color space to have values ranging from 0–127 along each dimension. We represent this mapping as a *color map*, created off-board by hand-labeling a set of images captured using the robot's camera. To generalize from the hand-labeled data, which covers roughly 3% of the whole space, the color label assigned to each cell in the color map is modified to be the weighted average of the cells within a certain *Manhattan distance* (a form of *Nearest Neighbor*). The resulting color map ($\approx$ 2 megabytes) is loaded on the robot to segment its input images.

During segmentation, we find contiguous *regions* of constant colors by organizing the image pixels into run-lengths [3,4]. Adjacent run-lengths of the same color are merged using the Union-Find algorithm [5]. We then build rectangular boundaries around the merged regions, *bounding boxes*, which store properties corresponding to each region such as its dimensions.

In the object-recognition phase, we use these regions along with domain knowl-

edge to detect the color-coded objects in the environment. One challenge of this task is distinguishing the objects of interest from other objects surrounding the field that happen to be segmented as the same color. For instance, a person standing by the field in an orange shirt may be identified as the ball. By using heuristic constraints on the size, pixel density, and relative locations of the regions, we can successfully isolate the objects of interest. These properties are also used to estimate the uncertainty in each object's distance and angle measurements, based on how well they conform to the expected values. In addition to colored objects, we also recognize field lines by searching for linear green-white transitions. Figure 1 shows the results from these processing steps. More images and videos taken by the robot are available online.[5] Full details of this baseline vision system are available in [6].

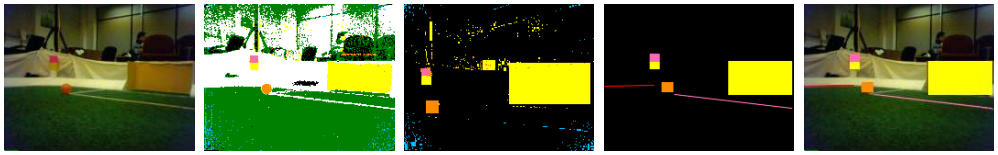

Fig. 1. Successive processing stages of the baseline vision system.

Other researchers working in the RoboCup domain have developed similar vision systems [3,7]. These systems as well as our own baseline implementation suffer from two major drawbacks: they require time-consuming manual color calibration and are highly sensitive to illumination changes. In almost all of these systems several ($\approx 20 - 30$) images need to be hand-labeled to generate the color map. Because this tedious process can take hours to complete, it is performed rather infrequently. This infrequent recalibration introduces great uncertainty into vision processing, because as conditions gradually change, the color map becomes increasingly obsolete.

To eliminate the time-consuming manual color calibration process, we developed an algorithm to enable the robot to autonomously learn the desired colors using the *structure* of the environment: known locations, shapes and colors of the objects in its world. Each color that the robot has to recognize is modeled as a three-dimensional Gaussian with mutually independent color channels. This algorithm requires that the robot have both training images and a model of its world with known locations of uniquely color-coded objects.

The robot starts at a known fixed initial position with an empty color map and traverses a specified sequence of positions on the field. At each such position it learns about one or more colors by looking for candidate image regions of unknown color that match the world model description of the objects. Note that the robot does not have any labeled data; it chooses appropriate pixels to learn the mean and variance of the Gaussians, which in turn are used to

[5] `http://www.cs.utexas.edu/users/AustinVilla/?p=research/robust_vision`

generate the final color map using the Bayesian decision rule. The color learned at each stage helps in the detection of the later colors by increasing the robot's ability to parse its environment. The effectiveness of the learned color map is demonstrated in Figure 2.

This algorithm works under different illumination conditions and different field settings. The segmentation performance of this color map, learned autonomously in less than five minutes, is comparable to that of the hand-labeled color map, which takes an hour or more to create [8]. Several sample images and a video of the the algorithm in action, as seen by the robot's camera, can be found online. [6]

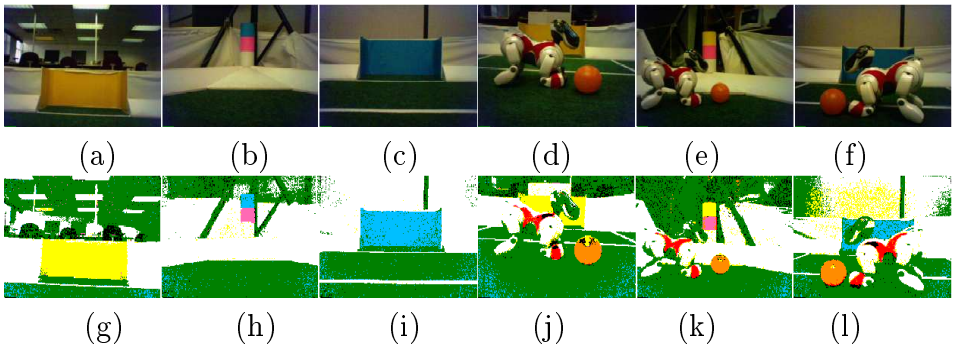

Fig. 2. Results from the autonomous color calibration algorithm. (a)-(f) Input, (g)-(l) Segmented with learned color map.

Although this color learning mechanism provides a means for autonomously recalibrating the color map, it does not provide a means for recognizing changes in illumination conditions. A color map trained under one illumination condition can become totally useless if the lighting conditions change, due to the non-linear shift in colors. To provide robustness to illumination changes, we hypothesized that different images from the same illumination condition would have measurably similar color space distributions, as compared to distributions from different illumination conditions.

We consider three discrete illumination conditions, *bright* ($\approx 1500lux$), *dark* ($\approx 400lux$), and *intermediate* (≈ 900). During the training phase, we train a color map and collected a set of sample images of the environment for each illumination condition. We use the *normalized RGB (rgb)* color space, which inherently provides some illumination insensitivity [9], and store the sample image distributions in *(r,g)*. For comparing two distributions we use the KL-divergence measure [10].

During its normal operation, the robot periodically samples an input image to generate the *(r,g)* distribution which is compared with the stored sample distributions. The sample image is assigned an illumination class label based

[6] `http://www.cs.utexas.edu/users/AustinVilla/?p=research/auto_vis`

on the training sample it is *most similar* to. If a sufficient number of sample images are classified as belonging to a particular illumination class, the robot considers itself to be in that illumination condition and uses the appropriate color map for subsequent operations.

This mechanism involves the experimental estimation of a set of parameters, which involves a trade-off between correctly identifying illumination changes as soon as possible and not interfering with the normal operation of the robot. With this approach, the robot performs efficiently and detects changes in illumination quickly. In addition, when faced with illumination conditions for which it has not been explicitly trained, the robot transitions to the *closest* illumination condition and, empirically, performs as efficiently as before. Therefore, it does not need to consider the continuous variation of illumination. Videos of this process [11] are available online. [7]

In this section, we have summarized three algorithms that enable the robot to deal with the uncertainty in its visual input. The baseline vision system performs color segmentation and object recognition in real-time under rapid camera motions, but it requires manual color calibration and is sensitive to illumination changes. The color learning approach autonomously learns the desired color distributions using the environmental structure. Robustness to illumination changes is achieved using color maps and sample image distributions over a few discrete illumination conditions.

4 Uncertain Sensor and Actuator Models

The algorithms described in Section 3 greatly reduce the uncertainty in the robot's visual output, i.e. the location and size of objects in the visual field. However, another significant source of uncertainty in the robot's processing comes from translating that visual data into useful information, such as the robot's actual distance to an object seen in an image. To accomplish this translation, the robot relies on a *sensor model* which defines the relationship between the object's properties in the image and its position relative to the robot on the field. Another model that the robot uses is an *action model*, which adjusts its estimate of the world state according to the actions it executes. Both the sensor and action models are inherently noisy. Inaccuracies in the action and sensor models inevitably lead to inaccuracy in the robot's location estimate. The resulting uncertainty can therefore be reduced by ensuring that the robot's action and sensor models are accurately calibrated.

Sensor and actuator models are typically calibrated manually: sensor readings are correlated with actual measured distances to objects, and robot actuator commands are measured with a stopwatch and a tape measure. However this type of approach has significant drawbacks. It is labor intensive, and the

[7] `http://www.cs.utexas.edu/~AustinVilla/legged/illumination`

model is necessarily tuned to a specific environment and may not apply more generally. A technique for *autonomously* calibrating both models simultaneously, called ASAMI (Autonomous Sensor and Actuator Model Induction), is presented in this section.

ASAMI explores the problem of autonomous model learning in the context of a specific, somewhat simplified, setting. The robot learns a sensor model and an action model, each represented by a calibration function. The sensor model function maps the various readings of a visual sensor to relative distances from a fixed landmark, and the action model function maps a range of action commands to the velocities of the corresponding movements. ASAMI is both autonomous and unsupervised, in that the robot never receives any feedback as to its actual location or velocity. ASAMI's goal is for the robot to learn action and sensor models that accurately reflect its distances and velocities.

ASAMI involves the robot performing the following three tasks simultaneously.

- Walking forwards and backwards while its visual sensor faces a fixed target, covering a range of relevant distances and velocities.
- Learning a function from action commands to actual velocities, assuming the distance calibration for the visual sensor is accurate.
- Learning a function from distance observation data to its distances from the target, assuming the robot has an accurate sense of its velocities.

This process successfully learns action and sensor models that closely approximate measurements made manually with a stopwatch and a tape measure.

The results reported in this section make use of the vision processing module described in Section 3 as well as a learned walking module [12]. To move forwards and backwards at different speeds, the robot interpolates between parameters for an idle walk, a fast forwards walk, and a fast backwards walk. As the experiments described below demonstrate, the resulting speed is a non-linear function of the parameters.

Meanwhile, the Aibo's visual sensor is based on its camera, which, as described in Section 3, is used to recognize objects including a colored cylindrical beacon that the robot can use to help it localize while on a playing field. The height of the beacon in the robot's image plane decreases with the robot's distance from the beacon; this observed height (in pixels) is the visual sensor reading used for the experiments reported in this article. A video of the Aibo performing its training behavior is available online.[8]

Because the robot is trying to learn two arbitrary continuous functions, it must represent them with a function approximator. Polynomial regression is

[8] http://www.cs.utexas.edu/~AustinVilla/?p=research/simultaneous_calibration

used for both functions. Furthermore, ASAMI learns the action and sensor models *from each other* in that it is not given any ground truth as to the robot's distance to the beacon or its speed. Therefore, it cannot learn the two models in any particular units. However, the learned action and sensor models are consistent with each other. Note that this property is sufficient for it to perform domain-specific tasks, such as predicting the amount of time a specific action command will take to yield a certain visual sensor reading.

Specifically, as the robot moves towards and away from the beacon, we denote its (actual) distance from the beacon at time t as $x(t)$. The robot's kth visual sensor observation occurs at time t_k and is denoted by obs_k. Each value reported by the visual sensor corresponds to a specific distance. This sensor model function is denoted by S, so that $x(t_k) = S(obs_k)$. The function S is one of the two functions that the robot is trying to learn. At the same time, the robot continuously executes an action command, $C(t)$, that varies with time. Each action command moves the robot at a specific velocity, and we denote the function from command to velocity by A. The robot learns this action model A along with the sensor model S. The action model also provides information about the robot's location: $x(t) = x(0) + \int_0^t A(C(s))\,ds$. ASAMI works by implicitly performing a continual comparison of these two sources of information. The robot knows the values of obs_k, t_k, and $C(t)$, and its task is to learn the functions A and S.

Note that the sensations and action effects are continually perturbed by zero-mean random noise, so that formally S and A represent the *average* distance or velocity corresponding to a given sensation or action selection. This noise represents an unavoidable source of uncertainty for the robot, but by estimating S and A as accurately as possible, the uncertainty is minimized.

ASAMI learns the action and sensor models simultaneously. To learn the sensor model, it assumes the action model is correct, and uses the resulting state estimate (the location estimate based on the action model), denoted by $x_a(t)$, as training data for the sensor model. Similarly, to learn the action model, ASAMI uses a location estimate based on the current sensor model, $x_s(t)$, to learn the action model.

Both models can be learned simultaneously because, even though the action (sensor) model learned from an inaccurate sensor (action) model will be inaccurate, it will be an improvement. As each model grows more accurate, its ability to help the other model improve grows. As this bootstrapping process continues, the two models converge to functions that accurately reflect what they are trying to model. Because both models grow in accuracy as time goes on, the regressions should give more weight to the more recent data points. Thus a weighted regression is used, where each data point has a weight that decreases over time [13].

After ASAMI has run for a pre-set amount of time (two and a half minutes), we consider its best estimates for A and S to be the models that it has learned at that point. The success of ASAMI is evaluated by comparing the learned action and sensor models to those measured with a stopwatch and a tape measure. A typical run is depicted in Figure 3a. Over the course of a trial, both models get progressively more accurate. The learning curves are depicted in Figure 3b. Both models' errors are shown, compared to the best possible error for the measured model and the degree of the polynomial being learned. The data is averaged over 15 trials [13].

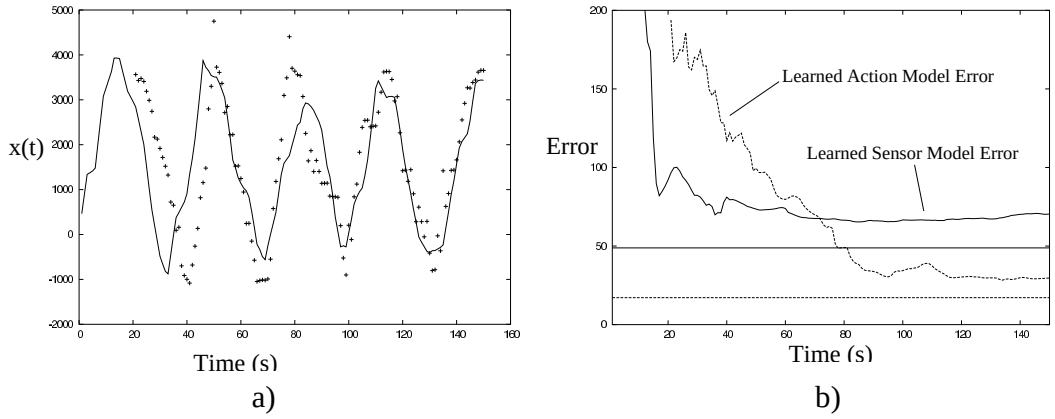

Fig. 3. a) In this example run, the +'s are values of $x_s(t)$, and the curve depicts $x_a(t)$. Over time, each model learns how to keep its estimate of the location close to the other model's estimate. b) The error for the action model is in mm/s, and for the sensor model in mm. The horizontal lines are at the minimum possible error to the measured models for a polynomial of the appropriate degree.

Inaccuracy in the robot's action and sensor models leads directly to uncertainty in its location estimates. By learning accurate accounts of its action and sensor models, the robot can minimize the corresponding uncertainty. The technique presented in this section, ASAMI, enables a mobile robot to autonomously learn its sensor and action models in an environment with a fixed landmark. The following section discusses how the robot can make use of accurate action and sensor models to reduce the uncertainty generated during Monte Carlo localization.

5 Uncertain Robot Localization

Typically, on mobile robots, the action and sensor models feed into a probabilistic localization algorithm that explicitly represents the robot's uncertainty in its own location in the world. One such algorithm is particle filtering, also known as Monte Carlo Localization (MCL) [14,15]. MCL has been shown to be a robust solution for mobile robot localization, particularly in the face of collisions and large, unexpected movements (e.g. the "kidnapped robot" problem [15]). It has been well-studied on wheeled robots with range-finding

sensors. This section summarizes extensions that increase its robustness and reduce uncertainty for vision-based legged robots [16].

In Monte Carlo Localization, a robot estimates its position using a set of samples called *particles*. Each particle represents a hypothesis about the robot's *pose*: its global location (x, y) and orientation (θ). The density of particle probabilities represents a probability distribution over the space of possible poses. Each operating cycle, the robot updates its pose estimate based on its action and sensor models. In the *motion update*, each particle's pose is moved according to the velocity reported by the action model. Random noise is added to account for the model's uncertainty. Next, during the *observation update*, the sensor model is used to update each particle's probability. The model predicts the likelihood of the robot's observations given the particle's pose, and adjusts the particle's probability accordingly.

Finally, particles are resampled in proportion to their probabilities. High probability particles are duplicated, replacing particles with low probability. In addition, to cope with unexpected movements, standard MCL approaches use *reseeding*; during the resampling step, a few of the particles with low probability are replaced by estimates obtained by triangulation of the landmarks seen in the current frame [17].

We have shown that a vision-based legged robot, operating in a world with unmodeled movements, encounters some particularly difficult types of uncertainty during localization. Our improvements on the basic Monte Carlo Localization algorithm allow this theoretically well-grounded approach to be practically deployed in this tricky setting. In particular, we have demonstrated significant increases in localization accuracy and certainty by i) overcoming biased distance estimate; ii) improving the robot's action model; and iii) maintaining landmark histories.

In the observation update step of MCL, the likelihood of an observation can be calculated from the perceived distance and angle to the observed landmark. Distance estimates computed analytically using geometric methods tend to be inaccurate and are hence not used in standard MCL methods, resulting in the exclusive use of angle information for probability updates [18].

In our approach, a distance function is constructed via cubic regression based on empirical data relating the size of a landmark in the image to its distance from the robot. Including the resulting distance estimates in the localization update decreased localization error by more than 30% when compared with using only angle information, and by almost 50% when compared with using an analytic distance model [16].

Causing the robot to shorten its step as it approaches a target point so as to take advantage of a more precise motion model during the motion update

reduced the localization error by an additional 40%. Finally, enabling the robot to remember *landmark histories* did not have any significant effect during smooth, unobstructed motion. But it enabled the robot to recover much more quickly from unmodeled movements, such as colliding with another robot or being picked up and moved [16].

After such unmodeled movements, it is important to quickly triangulate one's pose from fixed landmarks. To do so, either two or three landmarks must be seen, depending on whether or not distance information is used. A shortcoming of previous reseeding approaches is that they require the landmarks to be seen in the same camera frame, which may not occur very frequently. We contribute a concrete mechanism to enable reseeding even when two landmarks are never seen concurrently.

Observed distances and angles to landmarks are stored over successive frames in a landmark history. These stored values are adjusted each frame based on the robot's odometry, computed by its action model. Successive observations of the same landmark are averaged, weighted by their confidence, then given as input for reseeding. Because the robot's action model is uncertain, the confidence is decayed each cycle that the record stays in the history. The robot's high-level vision module occasionally mistakes one object for another, resulting in a false observation. To prevent these false observations from having long-term consequences, records that have been in the history for too long are thrown out.

The robot's final pose estimate is represented by the set of particles in MCL. When a single estimate is required for planning and decision-making, we use the weighted average of the particles. The robot's certainty in its pose is found by averaging the particle weights. This estimated pose and corresponding certainty, combined with knowledge of the relative positions of movable objects in the environment, constitute the robot's *world state*, upon which all planning decisions are made.

6 Planning and Decision-Making Under Uncertainty

To this point, we have introduced methods for reducing the uncertainty in the robot's world state estimate that results from its vision, motion, and localization processes. In this section we place particular emphasis on the robot's methods for dealing with uncertainty in planning and decision-making. We introduce algorithms by which our robots can i) interleave planning, action, and information-gathering so as to reduce localization uncertainty; ii) execute consistent actions over time so as to prevent oscillations due to uncertainty; iii) determine when to use reactive behaviors instead of deliberative ones; and iv) share and merge local perceptual information among teammates as a way of accurately tracking the world state and planning collaborative actions.

6.1 Interleaving Planning, Acting, and Information-Gathering

When planning under uncertainty, it may be possible to act explicitly so as to reduce uncertainty, perhaps at some cost. For example, a lost driver in a new city can stop to ask for directions, or can follow signs to a known landmark and replan from there. When such information-gathering actions are available, the agent can take one of three basic attitudes towards dealing with uncertainty. First, at the most passive extreme, the agent can neglect to explicitly gather information, instead planning based on whatever information happens to be available. Second, an agent can take a slightly more active role in its information-gathering by acquiring missing information on an as-needed basis. Third, at the most active extreme, the agent could treat information-gathering as a first class planning operator and deliberatively maintain its level of certainty in the world state over the course of its entire plan. Here, we provide an example of this third form of fully interleaved planning and information-gathering.

In RoboCup soccer, the robot's main focus is on the ball. It must constantly track the ball's position and act decisively as soon as it gains possession. At the same time, it must stay well-localized to make good planning decisions. Because it is often difficult for the robot to see landmarks when its head is pointed down at the ball, there is a trade-off between tracking moving targets (the ball and the opponents) and staying localized. In this context, information-gathering actions include communication with teammates and purposely looking for landmarks to improve localization accuracy. In a behavior called *active localization*, the robot occasionally shifts its focus from the ball to actively look for landmarks to improve its localization estimate.

Active localization is triggered when the uncertainty in the localization estimate becomes too large. If the localization certainty falls below a threshold, the robot uses its current pose estimate and the known geometry of the world to predict the relative positions of the various landmarks. It then uses this knowledge to plan the motion of its head (pan and tilt) that should allow it to see the closest markers. Because performing active localization could cause the robot to lose track of the ball's position, especially when the ball is nearby, active localization is performed only when the robot is a sufficient distance (more than 800mm) away from the ball.

The robot's objective in including active localization in its action plan is to arrive at the ball with high certainty in its location, so that it does not need to pause to localize *after* reaching the ball. To verify that active localization can achieve this objective, we performed the following experiment.

The robot starts at a fixed point slightly behind the center of the field with the ball near the edge of the opposite goal box. At the start of a trial, the

robot initiates its plan of walking to the ball and kicking it into the goal. Two seconds after the robot begins executing its plan, we impede its motion for four seconds by holding it still. While the robot attempts to walk towards the ball, the simulated collision disrupts the robot's localization estimate. We then release the robot and allow it to continue executing its goal-scoring behavior until it either successfully scores a goal or fails by kicking the ball out of bounds.

This experiment was performed with and without active localization and comprised 15 trials. For the successful trials, we recorded the number of attempted kicks before scoring as well as the total time taken. The results, shown in Table 1, show that active localization significantly improves the robot's ability to score quickly and consistently. The time results are statistically significant (p-value of 1.385×10^{-5} using a one-tailed t-test).

Active Localization	Avg. Time	Avg. Attempts	Success Rate
Without	26.11 ± 5.74	2.7 ± 1.16	20%
With	15.617 ± 6.33	1.2 ± 0.42	67%

Table 1
Time, number of attempts, and success rate for goal-scoring with and without active localization.

Without the active localization, the robot often ends up with a wrong pose estimate when it it gets to the ball. It kicks the ball in the wrong direction and then has to make more than one attempt before it gets the ball into the goal. When using active localization the robot almost always kicks the ball into the goal on its first attempt.

6.2 The Task Hierarchy

One common danger of planning under uncertainty is that fluctuations in a robot's estimated world state can cause the robot to vacillate among the behaviors planned from each perceived state. To counter this effect, the robot must be equipped with some form of hysteresis that biases it towards pursuing consistent subgoals over time [19]. This section presents our novel action selection paradigm designed for this purpose.

In the absence of uncertainty, a purely reactive architecture suffices to describe intelligent agent behaviors. One well-known such architecture is the production rule system, which consists of if-then rules that are evaluated at each action opportunity to map world states to action choices. These systems are often used to describe behaviors for agents in the RoboCup Simulated Soccer league, where agents have much better sensors than those that exist in the real world today [20]. In this simulation environment, the agents can trust their world state knowledge to be stable and reliable.

However, the highly noisy sensors used in the RoboCup Legged League prevent any teams known to the authors from using production rule systems. Instead, many teams use finite state machines (FSMs) to describe behaviors. Robots using this architecture switch behaviors only when their observations provide strong enough evidence that the current behavior is no longer appropriate.

Although FSMs are simple to implement, they can be hard to maintain, refine, and expand. For this reason, we designed a *task hierarchy* framework [21]. Instead of representing each behavior or activity as an atomic state, we create tasks that may recursively call other tasks. Like a subroutine call, the invocation of a task may persist for some time (throughout multiple low-level execution cycles) and maintain local state information. Unlike typical subroutines, each task in the stack of active subtasks continually monitors the world state and may switch to a new subtask in response. The stack thus corresponds to a consistent set of active subgoals, and the robot benefits from hysteresis at each level of the hierarchy. This framework thus provides more flexibility than FSMs while generalizing their ability to enable hysteresis, which is so important when acting under uncertainty.

6.3 Opportunistic Reactivity

As presented throughout this article, a large source of uncertainty in planning is the robot's localization estimate, which in turn comes from the robot's sensation and action *histories*. However, in certain circumstances, there may be enough information from the robot's instantaneous perceptions to make a *reactive* decision. That is, the correct action to take is the same, regardless of the details of the world state.

We take advantage of such opportunistic reactivity in our robots by enabling them to shoot directly towards the goal whenever it is close and visible. When the robot acquires the ball in the quarter of the field closest to the offensive goal, it first turns to the angle where the goal should be located, assuming its localization estimate is correct. However, once it reaches that angle, it makes a small adjustment to face the center of the largest region of goal-color that it has seen in the last few vision frames. After this adjustment, it kicks the ball.

The robot's objective in this situation is to kick toward the largest opening into the goal, avoiding all possible obstacles (including both things that are modeled by the world state, such as the position of the opponent goalie or other robots, and things which are entirely unmodeled, such as a referee's leg). Since the robot cannot accurately identify all possible obstacles, and since the robot's estimation of its own location is itself prone to uncertainty, the best information the robot has about the location of this opening is its immediate perception about regions of goal-color.

Note that this opportunistic reactivity contrasts with action architectures that

fully integrate reactive and deliberative reasoning [22,23]. Our robot acts entirely based on its world model except for during such exceptional circumstances when the immediate perceptions provide all the information necessary to act.

6.4 World State Representation and Communication

To this point, we have focused on how an *individual* robot can plan its actions in the face of uncertainty. A *multi-robot* environment introduces new opportunities and additional challenges with regard to acting under uncertainty. For instance, robots may share their own world state information with their teammates to improve the accuracy of each other's estimates. However, when there are large discrepancies in world state estimates between teammates, coordinating behaviors can be a challenge.

In our multiagent scenario, each robot maintains its own world state estimate. The robot tracks the ball and opponent positions using a Kalman filter-like representation [24]. When the robot sees the ball, the ball's relative distance and angle are represented as a two-dimensional Gaussian with variances computed from the uncertainty of the observation [21]. Each ball observation is merged with the previous estimate, which is first adjusted in accordance with the robot's motion. The merging process gives more weight to observations with lower variances. If the ball is not seen, the current estimate's certainty is degraded by increasing its variance. Opponent position estimates are maintained similarly. We also maintain an estimate of the relative velocity of the ball, based on the change in ball position estimates over a few frames. Velocity information can be used to update the ball's position estimate even when the ball is not seen, for example when performing active localization (Section 6.1).

Using this probabilistic framework to represent the various movable objects in the world, the robots are able to incorporate information communicated by teammates. To reliably merge teammate information with its own estimate, a robot must know that teammate's uncertainty in the information provided.

When each robot broadcasts its state information to its teammates, it must convert its egocentric representation to the global coordinate system using the robot's estimate of its own position. The uncertainties of the communicated information are therefore a function of the relative object uncertainty and the robot's own position uncertainty. A robot must be sufficiently certain of both estimates before it will communicate information about that object.

When merging the ball estimates from teammates, a robot primarily trusts what it sees over what is communicated, i.e. it considers the teammates' estimates of the ball only when the certainty of its own ball location estimate is low. It then merges the teammates' estimates and uses the result to decide the direction in which to start searching for the ball. Without the communicated

information, the robot typically spends much of its time recovering after losing sight of the ball, especially after sudden ball movements such as kicks. With communication. however, if one robot sees the ball, its teammates are able to turn in the most probable direction to recover the ball's position quickly.

To empirically test the advantage of information sharing between robots, we performed the following experiment. Two robots are placed on the field, one in the goalkeeper position at the center of the goal, the other near the far corner of the opposite goal. We place the ball directly in front of the first robot. In this position, the ball is in clear sight of the first robot but too far away to be seen by the second robot.

The second robot's goal is to find and approach the ball, which it initially cannot see. Our hypothesis was that the robot would perform best when selectively merging information from its teammate. That is, it could improve performance by listening to its teammate's ball information when it was uncertain itself. At the same time, we expected that if the robot continued to use the merged estimate even when the ball was in plain sight, its performance would degrade. The reasoning behind this hypothesis is that the robots' local, relative ball estimates are much more accurate than their global estimates, which must rely on both robots' estimates of their own poses. Thus for a robot that sees the ball, incorporating a teammate's global ball estimate is more likely to degrade the estimate quality.

Results verifying these effects over 15 trials are shown in Table 2. A successful trial is one in which the robot is able to touch the ball in less than one minute. The average time is calculated for successful trials only.

Merged Estimates	Average Time (s)	Success Rate
Never	28.72 ± 11.9	67%
When needed	15.87 ± 1.7	100%
Always	38.73 ± 11.86	60%

Table 2
Time taken to find the ball using different communication paradigms.

All timing results are statistically significant according to a one-tailed t-test. The standard deviation is higher when merged estimates are never used because the robot takes random walks across the field and manages to find the ball faster in some trials. Similarly, when information from teammates is always taken into account, the improper merging can cause the robot to wander off in random directions. Note that with excessive merging the robot actually performs worse than in the case with no communication.

In other work done on information sharing in this domain [25], only the ball estimates benefited from the combination of sensory and communicated infor-

mation. However, we found that it is better for the robot to know the likely locations on the field where it might be obstructed. Therefore, in our case, robots always merge communicated opponent estimates. Once again, the estimates with higher certainty are given proportionately more importance. This information is used mainly to avoid the opponents. We observe that using the merged opponent estimates provides a significant improvement in the robot's behavior. For example, when the robot is close to an opponent (even one it cannot see), it is able to determine that it is necessary to clear the ball quickly.

Although all of the behaviors described above are planned based on a single averaged pose estimate derived from the localization algorithm, it would be possible to choose the action that is most effective for the range of possible locations in which the robot might be [26]. But overall, these behaviors are quite effective, as evidenced by our cumulative score of 22–1 in 5 games (and 3rd-place finish) at the 2005 RoboCup US Open tournament [9] , and quarterfinalist status at RoboCup 2005.

7 Related Work

In the early sections of this article, we have referred to some of the relevant research related to each of the model-building subtopics covered. In this section, we focus on the most related work pertaining to planning under uncertainty.

One family of approaches to planning under uncertainty comes from the classical planning community. Using a STRIPS-like representation of states and actions (operators), systems such as Weaver and Buridan take decision-theoretic approaches to searching for plans with maximum expected utility [27]. However the symbolic representations of the world state assumed by these approaches are often hard to come by in robotic applications.

In robotics, different techniques have been implemented for dealing with uncertainty in the robots' inputs and actions, depending on the application. Sim et al. [28] present an approach to SLAM and robot exploration that generates an optimized online control policy such that the robot can explore new places quickly while obtaining data that leads to the most accurate map of the world. A similar idea is presented by Whaite and Ferrie [29]. Roy et. al [30] look at the problem of uncertainty for health care robots where the robot has to find and assist residents of a health-care facility. Work has also been done on planning robot actions in partially observable environments using POMDPs [31]. POMDPs have been used to control robot medical assistants, which keep track of patients and detect missing people [32]. In multiagent scenarios, coordination graphs have been used to achieve cooperative behavior among agents,

[9] The top three teams were quite evenly matched as evidenced by the fact that we beat the eventual champion in an exhibition match and lost to the 2nd-place team by only 1 goal.

even in the absence of communication between them [33].

Though several such approaches exist for control, coordination and action selection, especially for multiagent teams, very few approaches address uncertainty simultaneously at different levels, from the low-level sensors to the high-level decision-making and back to the low-level actuators. In our domain, we have presented techniques that enable a group of four robots to do so while effectively sharing information and functioning efficiently as a team.

8 Conclusion

Mobile robots encounter uncertainty from many different sources. Compared to wheeled robots with distance-based sensors, legged robots with vision-based sensing must cope with extreme uncertainty. This article identifies sources of, and proposes methods for mitigating, uncertainty coming from pixel segmentation, sensor modeling, action modeling, and localization. It then focuses on the mechanisms for planning and decision-making in the face of the resulting uncertainty. We report methods for enabling the robots to i) interleave planning, action, and information-gathering; ii) execute consistent actions over time; iii) behave reactively when appropriate and iv) share and merge local perceptual information among teammates as a way of accurately tracking the world state.

All the experiments were conducted on a commercial, off-the-shelf robot and evaluated using the robotic soccer test-bed environment. Our ongoing research agenda includes generalizing to multiple platforms and testing these algorithms in more uncontrolled (e.g. outdoor) environments.

Acknowledgments

This research is supported in part by NSF CAREER award IIS-0237699, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035.

References

[1] The Sony Aibo robots, `http://www.sonystyle.com` (2004).

[2] H. Kitano, M. Asada, I. Noda, H. Matsubara, Robocup: Robot world cup, IEEE Robotics and Automation Magazine 5 (3) (1998) 30–36.

[3] W. Uther, S. Lenser, J. Bruce, M. Hock, M. Veloso, Cm-pack'01: Fast legged robot walking, robust localization, and team behaviors, in: The Fifth International RoboCup Symposium, Seattle, USA, 2001.

[4] R. C. Gonzalez, R. E. Woods, Digital Image Processing, Prentice Hall, 2002.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms (Second Edition), MIT Press, September, 2001.

[6] M. Sridharan, P. Stone, Real-time vision on a mobile robot platform, in: The IEEE International Conference on Intelligent Robots and Systems (IROS), 2005.

[7] D. Cohen, Y. H. Ooi, P. Vernaza, D. D. Lee, UPennalizers, RoboCup-2003: The Seventh RoboCup Competitions and Conferences, Springer Verlag, Berlin, 2004.

[8] M. Sridharan, P. Stone, Autonomous color learning on a mobile robot, in: The Twentieth National Conference on Artificial Intelligence (AAAI), 2005.

[9] B. D. Zarit, B. J. Super, F. K. H. Quek, Comparison of five color models in skin pixel classification, in: Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999, pp. 58–63.

[10] D. H. Johnson, C. M. Gruner, K. Baggerly, C. Seshagiri, Information-theoretic analysis of neural coding, In Journal of Computational Neuroscience 10 (2001) 47–69.

[11] M. Sridharan, P. Stone, Towards illumination invariance in the legged league, in: The Eighth International RoboCup Symposium, Lisbon, Portugal, 2004.

[12] N. Kohl, P. Stone, Policy gradient reinforcement learning for fast quadrupedal locomotion, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2004.

[13] D. Stronger, P. Stone, Towards autonomous sensor and actuator model induction on a mobile robot, Connection Science 18 (2), special Issue on Developmental Robotics. To appear.

[14] S. Thrun, Particle filters in robotics, in: The 17th Annual Conference on Uncertainty in AI (UAI), 2002.

[15] D. Fox, W. Burgard, H. Kruppa, S. Thrun, Markov localization for mobile robots in dynamic environments, Journal of Artificial Intelligence 11.

[16] M. Sridharan, G. Kuhlmann, P. Stone, Practical Vision-based Monte Carlo Localization on a Legged Robot, in: The International Conference on Robotics and Automation, 2005.

[17] S. Lenser, M. Veloso, Sensor resetting localization for poorly modelled mobile robots, in: The International Conference on Robotics and Automation, 2000.

[18] T. Rofer, M. Jungel, Vision-based Fast and Reactive Monte-Carlo Localization, in: The IEEE International Conference on Robotics and Automation, Taipei, Taiwan, 2003, pp. 856–861.

[19] C. Castelpietra, L. Iocchi, D. Nardi, R. Rosati, Coordination in multi-agent autonomous cognitive robotic systems, in: The Second International Cognitive Robotics Workshop (CogRob), 2000.

[20] I. Noda, H. Matsubara, K. Hiraki, I. Frank, Soccer server: A tool for research on multiagent systems, Applied Artificial Intelligence 12 (1998) 233–250.

[21] P. Stone, K. Dresner, P. Fidelman, N. K. Jong, N. Kohl, G. Kuhlmann, E. Lin, M. Sridharan, D. Stronger, UT Austin Villa 2004: Coming of Age, AI Technical Report 04-313, Tech. rep., Department of Computer Sciences, University of Texas at Austin (October 2004).

[22] M. K. Sahota, Reactive deliberation: An architecture for real-time intelligent control in dynamic environments, in: Proceedings of the Twelfth National Conference on Artificial Intelligence, 1994, pp. 1303–1308.

[23] E. Gat, Three-layer architectures, in: D. Kortenkamp, R. P. Bonasso, R. Murphy (Eds.), Artificial Intelligence and Mobile Robots, AAAI Press, Menlo Park, CA, 1998, pp. 195–210.

[24] Y. Bar-shalom, X. R. Li, T. Kirubarajan, Estimation with Applications to Tracking and Navigation, Wiley and Sons, 2001.

[25] M. Roth, D. Vail, M. Veloso, A Real-time World Model for Multi-Robot Teams with High-Latency Communication., in: The IEEE International Conference on Intelligent Robots and Systems (IROS), 2005.

[26] C. Kwok, D. Fox, Reinforcement learning for sensing strategies, in: The IEEE International Conference on Intelligent Robots and Systems, 2004.

[27] J. Blythe, Decision-theoretic planning, AI Magazine 20 (2) (1999) 37–54.

[28] R. Sim, G. Dudek, N. Roy, Online Control Policy Optimization for Minimizing Map Uncertainty during Exploration, in: The IEEE International Conference on Robotics and Automation (ICRA), 2004, pp. 1758–1763.

[29] P. Whaite, F. P. Ferrie, Autonomous Exploration: Driven by Uncertainty, in: The International Conference on Computer Vision and Pattern Recognition (CVPR), 1994.

[30] N. Roy, G. Gordon, S. Thrun, Planning under Uncertainty for Reliable Health Care Robotics, in: The Fourth International Conference on Field and Service Robots (FSR), 2003.

[31] J. M. Porta, M. T. J. Spaan, N. Vlassis, Robot planning in partially observable continuous domains, in: Robotics: Science and Systems, 2005.

[32] J. Pineau, G. Gordon, POMDP planning for robust robot control, in: The Twelveth International Symposium on Robotics Research, 2005.

[33] J. R. Kok, M. T. J. Spaan, N. Vlassis, Non-communicative multi-robot coordination in dynamic environments, Robotics and Autonomous Systems 50 (2-3) (2005) 99–114.