Multi-Fidelity Robotic Behaviors: Acting Under Variable State Information

Elly Winner and Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{elly,veloso}@cs.cmu.edu
http://www.cs.cmu.edu/{~elly,~mmv}

Abstract

Our work is driven by one of the core purposes of artificial intelligence, namely to develop and contribute real robotic agents that act and achieve complex high-level goals in real-time environments. Behaviors for robots select actions as a function of the state of the robot and of the world. Designing robust and appropriate robotic behaviors is a well-recognized and difficult problem, in particular due to the noise, uncertainty, and cost of acquiring the necessary state information. We addressed this challenge within the concrete domain of robotic soccer with the fully autonomous Sony dog robots. In this paper, we present one of the outcomes of this research work, namely the introduction of *multi-fidelity behaviors*, as an approach for robotic behaviors to explicitly and efficiently handle different levels of accuracy of the state information. The paper motivates and introduces the general approach and then reports closely on the concrete work with the Sony robots. Our developed multi-fidelity behaviors allow for the robots to successfully achieve their goals in a highly dynamic and adversarial environment. A robot acts according to a set of behaviors that aggressively balance the cost of acquiring the state information versus their value towards the achievement of the robot's high-level goals. The paper includes empirical experiments needed to support the choice of the parameters that control the investment of effort to gather incrementally-accurate state information.

Introduction

Intelligent agents operating in dynamic domains rely heavily on real-time information about the world around them to direct and control their behaviors. This information may be raw sensor data, processed sensor data, or sensor data that the agent has had to spend effort to acquire.

In complex domains, raw sensor data often is not refined enough to allow for high-level deliberation or control. In cases like this, the agent must spend effort to process the data it receives from its sensors. In addition, many real-world agents do not have complete perception. Sometimes the sensor data needed to make decisions about the agent's behavior is not immediately available. The agent must spend time to acquire the raw data, perhaps in addition to the time spent processing it.

Clearly, in many complex and dynamic domains, the quality of the agent's information about the world around it is dependent on the amount of time and resources the agent is able to devote to acquiring that information. In order for an agent to act sensibly in such a domain, it must be able to balance its need for information with the benefit this information provides. It must be able to take advantage of all available information and act sensibly when less information is available.

Our novel multi-fidelity behaviors approach provides a behavior framework which allows an agent's performance to upgrade quickly in the presence of better information and to downgrade gracefully when information quality is worse.

The remainder of this paper is organized as follows. THIS IS NO LONGER VALID AT ALL. Section presents our scheme for balancing the benefit of accurate localization information with the time needed to achieve it. Section presents our algorithm for scoring goals without any localization information. Section ?? presents the methods we use to allow the robot to upgrade and degrade its performace gracefully as the availability of information changes. It also presents the localization-dependent performance enhancements we implemented. Section ?? describes some of the special cases for which the balance between localization and action described in Section no longer apply, and the algorithms we use to deal with this. Section ?? describes the results of tests we ran to determine the effects of different localization strategies. Section concludes the paper.

Multi-Fidelity Behaviors

Multi-fidelity behaviors[1]

Mode-Based Architecture

We built a control architecture based on basic modes of behavior we identified as important. To switch between modes, the robot uses knowledge about basic features

[1] Interestingly, complex and dynamic artificial intelligence systems share a problem with wireless networking: the need to adapt performance as the cost and availability of resources dynamically varies.

of its state, such as: i) whether it is paused, ii) into which goal it is trying to push the ball, iii) whether it is in possession of the ball, whether it is close to the ball, and iv) whether it knows where the goal is. Performance enhancements include switching to a different mode (for example, if the robot has good enough localization information, it knows where the goal is without seeing it, so it can appropriately change from circling the ball to pushing it towards the goal), and improving the performance of the robot within a mode.

We are able to optimize the performance of the low-level implementation of the modes by using localization information. By separating the high-level behavior from the low-level implementation, we ensure that the robot's high-level behaviors do not change frequently as the available information changes. Instead, the way the robot executes these behaviors changes in response to the lack or availability of localization information.

Implementation in RoboCup-99 Legged League

We applied our multi-fidelity behavior approach to the robotic soccer domain in the Sony legged robot league of RoboCup-99. All teams in the RoboCup-99 legged robot league used the same hardware platform: the Sony quadruped legged robots (?), shown in figure ref-fig:dog. The robots are autonomous, and have onboard cameras. The onboard processor provides image processing, localization and control. The robots are not remotely controlled in any way, and as of now, no communication is possible in this multi-robot system. The only information available for decision making comes from the robot's onboard camera and from sensors which report on the state of the robot's body.

The Sony AIBO robots were commercially sold in March of 1999. But the robots used for the RoboCup-99 competition are not the same as those commercially available. The robots we used are equipped with slightly different hardware, and, unlike the commercial product, are programmable.[2]

Figure 1: The Sony quadruped robot dog with a ball.

[2]Our extensive videos of this event provide invaluable illustrative support to the work presented in this paper.

The soccer game consists of two ten-minute halves, each begun with a kickoff. In the kickoff, the ball begins in the center of the field, and each team may position its robots on its own side of the field. After each goal, play resumes with another kickoff.

Each team consists of three robots. Like our team last year, CMTrio-98, and most of the other 1999 teams, we subdivided our team into two attackers and one goaltender. In this paper, we will focus on the attackers.

The field is 280 cm in length and 180 cm in width. The goals are centered on either end of the field, and are each 60 cm wide and 30 cm tall. Six unique landmarks are placed around the edges of the field (one at each corner, and one on each side of the halfway line) to help the robots localize themselves on the field. The landmarks are 20 cm off the ground and 20 cm high. Figure 2 shows a sketch of the playing field.

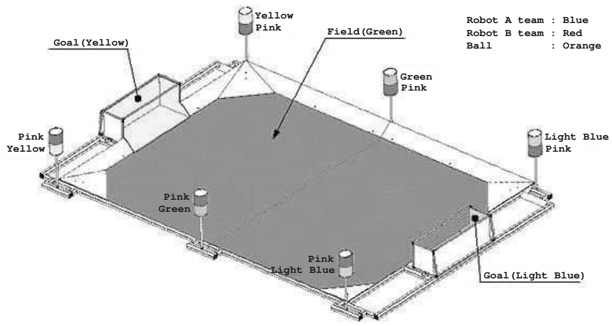

Figure 2: The playing field for the RoboCup-99 Sony legged robot league.

The robotic soccer domain is a very appropriate one in which to study the utility of our approach. Raw data from the robot's camera must be processed to extract the information used to control the robot's behavior. First, the data are sent to the vision module, which reports which objects are seen, with what confidence, and at what direction and distance (?). This information is then sent to the localization module, which reports an estimate of the robot's angle, θ, and of its x and y location on the field, along with the standard deviations (σ) of these values (?). The two-pass processing is computationally expensive, and is not always feasible, due to the computational limits of the robot's onboard processor. In these cases, the localization module does not recalculate its estimate of the robot's position and angle.

To estimate the robot's position, the localization system applies Monte Carlo sampling and sensor-based resetting to data about the position of field landmarks relative to the robot. The data provided by the localization system is not accurate when the robot has not recently seen landmarks. Therefore, to get accurate localization information, the robot must look for landmarks.

As the robot walks, the camera experiences pitch and roll, which causes the images it collects to change significantly from one frame to the next. Because of this, the vision system's identification of objects and the estimate of their distances and angles degrades. The localization system depends heavily on accurate information about the landmarks, so our system requires that the robot stop moving while looking for landmarks. The process of stopping and scanning for landmarks usually takes the robot between 15 and 20 seconds.

Because of these factors, it is often very costly for the robot to acquire information about its location on the field. Although it is obvious that this information is very useful to a soccer-playing robot, soccer, like other dynamic domains, is time-critical, so every moment spent looking around is lost time. Opponents can also use the robot's inattention to their advantage.

Modes of Behavior

To implement our multi-fidelity behaviors approach to the legged robots, we first had to identify the basic modes of behavior and define their interaction. We defined five modes of behavior for the attackers:

1. `Head Searching`: the robot is searching for the ball by turning its head;
2. `Searching`: the robot is searching for the ball by moving its head and its body;
3. `Approaching`: the robot is approaching the ball;
4. `Circling`: the robot is circling the ball;
5. `Scoring`: the robot is pushing the ball towards the goal;

A simple algorithm describes the main algorithm the robot uses to switch between the modes of behavior. Localization-dependent enhancements slightly optimize this basic algorithm, but retain its spirit:

```
If robot sees ball and is not close to it,
  mode = Approaching;
If robot does not see ball and did recently,
  mode = Head Searching;
If robot does not see ball and has not recently,
  mode = Searching;
If robot does not know where goal is and
    is close to ball
  mode = Circling;
If robot knows where goal is and is close to ball,
  mode = Scoring.
```

Low-Level Behavior

Frequently during games, the standard deviations of the robot's localization information are so high that the information should not be used. As explained in Section , the robot should not stop and look every time its localization information is inaccurate. Therefore, we must make sure that it can act reasonably even when its localization information is not good enough to use.

We have divided our description of the robot's behavior into two categories. In one, the robot does not have possession of the ball; in the other, it does.

Behavior without the Ball The robot has three main behaviors when it does not have possession of the ball: recovering a recently lost ball, searching for the ball and approaching the ball. We will describe the algorithms the robot uses to perform these behaviors with no localization information.

- Recovering a Recently Lost Ball: The robot often loses sight of the ball while it is trying to manipulate it. This is usually because it has walked past the ball. It must search for the ball, but because it has just lost sight of the ball, more information is available for it to use.

  With no localization information, the robot cannot use a model of the world to try to return to the point at which the ball was last seen. However, because, in most cases, the robot has lost the ball because it has walked past it or pushed it to the side, a very efficient way to recover the ball is to walk backwards. We allowed the robot to do this for nearly half the width of the field. If it still does not see the ball, then the robot tries a strategy used by our team last year, CMTrio-98: it turns in the direction in which the ball was last seen (?). After this, the robot considers the ball lost and begins a random search for it.

- Searching for the Ball: When the robot does not know where the ball is, it must wander the field to search for it. Without any localization information, the robot cannot do a systematic "oriented" search of the field. So instead, we wrote a random search algorithm that does not rely on localization information at all. Until the robot sees the ball, it alternates between walking forward a random amount and turning a random amount.

- Approaching the Ball: Although it would be most efficient for the robot to chart an approach to the ball that would allow it to finish "behind" the ball, facing the goal, this is not possible without localization information. For that reason, when the robot has no localization information, it approaches the ball by running straight towards it.

Behavior with the Ball Once the robot has possession of the ball, its only goal is to push the ball into the goal. This is difficult withough localization information, because the robot does not know which direction to push the ball, or how far away the goal is. But we have written a simple algorithm that allows the robot to score goals without any localization information:

```
Until see goal,
  Circling: walk sideways around ball;
Scoring: walk forward into ball.
```

This allows the robot to score goals consistently with no information from the landmark-based localization module at all.

Localization-Dependent Performance Enhancements

We built alternate implementations of two of the robot's behavior modes that take advantage of good localization information. If the localization information is good, we also allow the robot to optimize its transition between modes.

Many performance enhancements are possible with perfect localization, but the three we developed, although they rely on some amount of localization information, are robust and reliable even with noisy information.

- Circling Mode: This optimization streamlines the circling mode. Based on knowledge of the robot's position, it decides which direction to circle around the ball, or whether to circle at all. If the robot has $\sigma\theta < 30°$ and $\sigma x, \sigma y < 60$ cm, then it can choose the shortest direction to circle around the ball. With $\sigma x, \sigma y < 30$ cm, the robot is able to realize whether this would mean circling into a wall and choose to circle in the other direction, or can realize that it is too close to a wall and will not be able to get directly behind the ball, so it should push the ball down the side of the field and wait until it reaches the corner to try to push it away from the wall.
- Approach Mode: This optimization makes the approach mode much more useful. If good localization information is available, the robot is able to use its approach to the ball to get into position behind it. If $\sigma\theta < 30°$, then the robot is able to "skew" its approach to the ball, so that when it reaches the ball, it is closer to its goal position of behind it. This allows it to complete part of the work of circling of the ball without circling and without localizing during the approach.
- Scoring without Vision: This enhancement allows the robot to transition between the circling and scoring modes more efficiently. We specified in Section that, in the absence of useful localization information, the robot should circle the ball until it sees the goal. However, if $\sigma\theta < 30°$ and $\sigma x, \sigma y < 600mm$, then if the robot is facing the right direction, *even if it does not see the goal*, we allow it to leave the circling mode and begin the scoring mode, pushing the ball towards the goal. This enables the robot to score goals consistently even when it cannot see the goal at all.

Unenhanced Behaviors

Although we built optimizations for two of the robot's modes and for one transition, we did not enhance everything. There are two reasons for this. Some behaviors do not benefit from localization information. Other behaviors are so urgent that even if they would benefit from localization information, stopping to acquire it would be too expensive, and so they eschew it entirely.

Behaviors with No Need for Localization One of the best ways of finding a balance between localization information and time spent to acquire it is simply to avoid localizing when it is not necessary. We have identified two times when it is unnecessary: when the robot has recently lost sight of the ball, and when has no information at all about the location of the ball. Our scheme allows the robot to act intelligently but does not rely on localization information in either of these situations.

- Recovering a Recently Lost Ball: The robot often loses sight of the ball while it is trying to manipulate it. This is usually because it has walked past the ball. It must search for the ball, but because it has just lost sight of the ball, more information is available for it to use.

  One way of solving this problem is to incorporate a full-scale world view into the robot architecture. Such a model would remember at what point the ball was last seen and attempt to steer the robot back toward that point. However, this would require a significant amount of overhead and would rely heavily on localization information in order to accurately move the robot.

  Instead, we implemented a very simple but extremely effective algorithm. Our team last year, CMTrio-98, searched for a lost ball by turning in the direction in which the ball was last seen (?). We incorporate this into our new algorithm. But first, the robot walks backwards for nearly half the width of the field. If it still does not see the ball, it then turns in the direction in which the ball was last seen. After this, the robot considers the ball lost and begins a random search for it.

  This two-step process almost always succeeds in finding a ball which the robot has briefly lost because almost every time the robot loses sight of the ball, it has walked past it, or nudged it to the side. When the robot backs up, the ball quickly comes back into view. If it does not, it is likely because another robot has taken the ball away, or is obstructing the ball from view.
- Random Search: When the robot does not know where the ball is, it must wander the field to search for it. One way of searching for the ball is to build an "oriented" search, in which the robot uses localization information to systematically search each area of the field. This relies on very accurate localization information and on a complete search of the field, both of which take a lot of time. Instead, we use a very simple algorithm that is much faster. We wrote a random search algorithm that does not rely on localization information at all. Until the robot sees the ball, it alternates between walking forward a random amount and turning a random amount.

  Our algorithm does not allow the robot to recognize that it is running into a wall or to walking into a goal. Because the vision module does not recognize walls, the former would be difficult to add. We did not add the latter because of time constraints, but it could easily be added.

Even without these improvements, we found that the random search algorithm allows the robot to cover the field more quickly, on average, than does a comparison scheme we built in which the robot attempts to walk a circuit of the field. This is because "oriented" searches, like the circuit, require the robot to have accurate localization information. To have such accurate localization information, the robot must frequently stop and look around, which takes time.

Localization information thus clearly does not help the robot to devise a more efficient path around the field. And, since the ball is no more likely to be in one segment of the field than in another, the localization information also does not help the robot decide the best way to move initially. Therefore, we do not allow the robot to attempt to localize itself while searching for the ball, nor do we use any information the robot already has about its localization to influence its search behavior.

Special Cases—Urgent Action In some cases, the balance between localization and action does not apply. Although each of these cases could perform much better with localization information, immediate action is necessary, so the cost of localizing thoroughly is prohibitively high. We have found shortcuts and compromises that allow the robot to perform as well as possible in these special cases, even without spending time looking around for landmarks.

- Approaching the Ball Possession of the ball is a critical part of a soccer game. The team that is able more frequently to possess the ball has an incredible advantage over its opponents. Therefore, in our strategy, when the robot sees the ball, it rushes towards it, not waiting to localize.

  This strategy has negatives, clearly. If the robot does not know where it is on the field, it will not know what to do with the ball when it gets to it. Nevertheless, it is better for a robot to look around when it is in possession of the ball than when it is farther from the ball. When the robot is standing near the ball, it is blocking one side of the ball from visibility and attacks, and is more able to respond quickly to an attack because it is already close by.

  Also, as discussed in Section ??, even if the robot had only limited localization information, it could still skew its approach to the ball, and choose the most efficient direction when circling the ball.

- Kickoff We were surprised to discover, in the initial games of RoboCup-99, how much an advantage is gained by winning the kickoff. When the ball moves to one side of the field, it is very difficult for the robots to move it to the other side of the field. In the RoboCup-99 games, the team that won the kickoff usually scored a goal, simply because the ball never emerged from the side of the field to which it was initially pushed.

  Because of this, we felt it was crucial to have an extremely aggressive kickoff. We took advantage of the fact that, at each kickoff, the robots were facing the opposite goal by having them run straight towards the ball, which is placed in the center of the field, between them and the goal.

  Although we have two attackers, and kickoff (and the resulting play) is therefore a multi-agent problem, we found it difficult to develop multi-agent strategies. There is currently no capability for communication between the robots. Our vision system cannot identify other players. However, we took advantage of the fact that we had two robots during the kickoff. Since we had the opportunity to place the robots manually before kickoff, we placed one closer to the ball. The second is able to take over the ball if the first loses sight of it briefly.

  We found it was best to allow them to run with the ball without localizing at all for almost half the length of the field—nearly the whole distance between their starting positions and the goal. Even when the error-prone motion of the robots causes them to stray far from their projected path towards the goal, they are usually able to drive the ball onto the other robots' side of the field before stopping to localize.

  This was a very important advantage in almost all of our games, since we were able to win most of the kickoffs. In the one game we lost, we had accidentally turned this feature off during the first half of the game, and our robots were no longer able to win the majority of the kickoffs. When we turned it back on in the second half, our robots again dominated the kickoffs, and were therefore able to score a goal and prevent the other team's robots from scoring any.

Balancing the Costs of Information

In every domain, a different balance must be struck between the costs of acquiring resources and the benefits of using them. Even in the Sony robotic soccer domain, different teams came to different conclusions about how much time should be spent acquiring localization information. This year's team from LRP University in France (?) chose to localize the robot very infrequently, if at all. However, the benefits of accurate localization are significant.

Our system uses two constraints to balance the cost and benefits of good localization information because we found that one constraint is insufficient. Because the goal is to achieve good localization information, at least one constraint must deal with that. However, minimum allowable quality thresholds on localization information are not sufficient to balance the cost and benefits of good localization information. If we set the thresholds to levels that allow the robots to use the localization information most of the time, then they spend most of their time looking for landmarks to maintain the high quality of information. However, if we try to lower the thresholds so that the robots will spend more time acting before stopping to look for landmarks, we find that in order for the robots consistently to act for a reasonable amount of time, the thresholds must be below usable levels. Although sometimes the quality of the information climbs to useful levels in this scheme, it is not maintained for very long. Even if the robot has not looked for landmarks in a long time, it will not stop to look until the quality of its information has fallen below the set threshold, even if its current information is unusable.

To avoid this problem, our two-constraint system constraints time and information quality. One constraint insures that the robot spend a sufficient amount of time acting. The other insures that if the robot's localization information is no longer usable, it will stop to look as soon as the time constraint allows it to.

We ran two experiments to determine the best values for these two constraints. We tested how long it would take one robot on an open field to score a goal. If the robot took longer than fifteen minutes to score a goal, we stopped the trial and recorded its time as fifteen minutes.

We positioned the robot at a fixed point, (700, 450) in our coordinate system. This corresponds to 3/4 of the way down the field from the yellow goal, and 1/4 of the way from the right side wall (if facing the yellow goal). We oriented the robot at $-135°$ in our coordinate system, or $45°$ to the left of facing straight towards the yellow goal. We placed the ball directly in front of it, at the midpoint of the width of the field, since we did not wish to time the robot's search algorithm. We timed how many minutes it took the robot to push the ball into the yellow goal.

Figures 3 and 4 show the results of the two experiments we ran. Each trial is represented by a small tick. The mean of each set of trials is indicated as a bold tick. The grey bar around the mean is one standard deviation.

The experiments have, as we expected, a very high variance, which captures the challenge of the control task inherent in the robots. The results from the robot's sensors have non-neglible variance and the motion of the robot is unreliable. Also, the statistical localization inevitably gives different absolute results even with exactly the same inputs. We believe that most of the variance is due to the high rate of error in the robot's motion. This unreliability is magnified when the ball is involved. Interestingly, when the robot tries to push the ball forward, it often ends up pushing it at an angle, or out to the side, or even walking past the ball. When it walks around the ball, it often accidentally taps it, sending the ball off in an unpredictable direction, as described in Section .

Constraint 1—Enforcing Action

Our first constraint requires the robot to spend a certain amount of time acting before it stops to look for landmarks. This is crucial for two reasons. The first we mentioned before: it is bad for the robot to spend most of its time localizing, as it would have to do if we required that the localization information always be usable. But we must worry about more than the percentage of time the robot spends localizing versus acting. We must also ensure that the robot does not interrupt its action too frequently. Each time the robot stops to look for landmarks, there is some chance that it will have trouble finding the ball when it finishes localizing and looks for it again. This happens because the robot accidentally nudges the ball away, or because it fails to stop moving before looking away from the ball. Therefore, stopping more frequently greatly increases the chance that this will happen and the robot will have to begin searching for the ball, a time-consuming procedure.

Our scheme uses a counter to require the robot to act for a specified amount of time before looking for landmarks. The amount of time the robot must act before looking could depend on the confidence the robot has in its current localization information and on its current goals. In our scheme, however, it is invariant.

We require that the robot act for the time it takes the image module to proccess 350 frames of data, or about 40 seconds. Recall that stopping to look for landmarks takes the robot between 15 and 20 seconds, not counting the time it takes it to recover the ball afterwards. So we demand that it spend about 2/3 of its time acting.

We chose to count the time in image frames processed by the vision module because a full system call is more time-consuming than using this information. The number of frames processed is relatively constant per unit time, and is immediately available to the control module since each processed frame invokes an update in the control module.

We conducted several tests to discover how long we should force the robot to act before looking for landmarks. Figure 3 shows the results of ten trials each of three different time intervals. We also tested the interval 500 frames (or about 60 seconds), but with this interval, the robot occasionally scores own-goals because its localization information is very poor by the end of the interval. Because of this, we did not consider that interval a possibility, and did not include it in the graph.

The standard deviations of the trials are, as shown, very high. Nevertheless, there is a clear penalty for localizing too frequently, as shown in the results for the 100-frame (or about 13 second interval). This is because so much stopping to look disrupts the robot's activity. It loses sight of the ball more frequently, and must stop to look for it. Also, scanning the landmarks so often simply takes a lot of time.

There is also a clear penalty for not localizing enough, as we found when we tested the 500-frame interval. If the robot does not look around enough, it takes it much longer to score goals. And, as mentioned in Section , the longer the robot spends moving around near the ball, the more likely it is to tap the ball in the wrong direc-

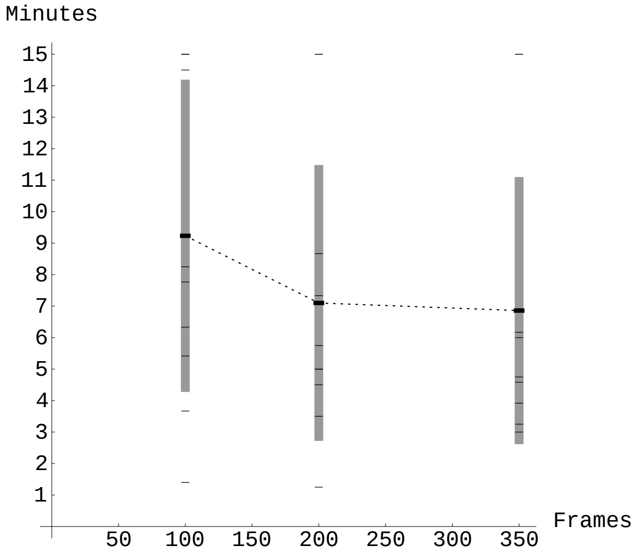

Figure 3: Time taken to score a goal versus how long we require the robot to act before looking for landmarks.

tion, which, as our results show us, sometimes leads it to score own-goals.

Although our results show that 350- and 200-frame intervals (about 40 seconds and 25 seconds, respectively) are roughly equivalent, we chose to use a 350-frame interval. In an actual game, the penalty for stopping more often is much higher than when there is only one robot on the field. When a robot stops to look for landmarks, other robots have a chance to take the ball away. Recall that stopping to look for landmarks takes the robot between 15 and 20 seconds, not counting the time it takes it to recover the ball afterwards. So we demand that it spend about 2/3 of its time acting.

Constraint 2—Sufficient Localization

The second constraint is how accurate we demand the localization information to be. We measure accuracy with the standard deviations returned by the localization module. If the information is accurate enough, the robot should not stop to look for landmarks when the timing constraint allows it to. But if the localization information is not accurate enough, the robot should not use it.

It is not immediately obvious how good our localization information must be before it is usable. Clearly, if our demands are too high, the robot will rarely be able to use the information it has gathered. And if they are too low, it will use information that is so inaccurate as to be useless at best, and damaging at worst.

Less Accurate Localization

We also conducted tests to determine how low we should require the standard deviations of the localization values to be before using them. Figure 4 shows the results of ten trials each of two different sets of

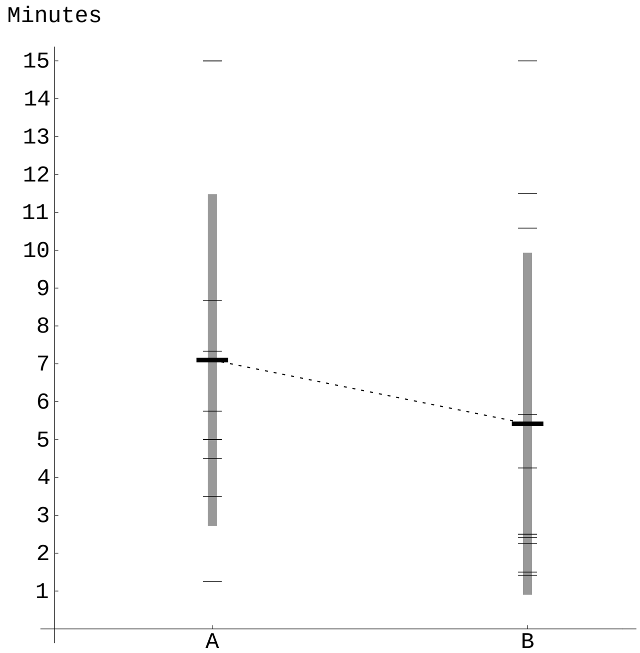

Figure 4: Time taken to score a goal versus how low we require the standard deviations of the localization values for θ (the robot's orientation) and x and y (its location on the field) to be before using them.

standard deviation maxima, labelled A and B. Setup A specifies $\sigma\theta < 15°$ and $\sigma x, \sigma y < 30$ cm. Setup B requires $\sigma\theta < 30°$ and $\sigma x, \sigma y < 60$ cm. We also tried two other setups, one wither lower standard deviation maxima (θ standard deviation of 5°, x and y standard deviation of 15 cm) and one with higher maxima (θ standard deviation of 60°, x and y standard deviation of 120 cm). However, we found that both of these settings caused the robots to frequently score own-goals. We concluded that both alternate settings were unsuitable and excluded them from the graph.

If we demand the localization values be too accurate before using them, we lose the benefit of using them frequently, and must revert to our original strategy, detailed in Section . If we are still able to use them sometimes, this simply slows us down, as seen in Figure 4, setup A. But as mentioned in Section , if we must rely too heavily on this strategy, the robot is much more likely to tip the ball in the wrong direction and accidentally make an own-goal. This is borne out by our experience with setting the maximum standard deviation values too low.

But if we demand too little accuracy from the localization values, the robot will rely on them even when they are faulty. This is clearly borne out by our experience with setting the standard deviation values too high. The robot mistakenly believed that it knew where it was and pushed the ball right into its own goal.

We chose to use the standard deviation values used in setup B from Figure 4 ($\sigma\theta < 30°$, $\sigma x, \sigma y < 60$ cm), because, in our experiments, there was a clear advantage to these settings.

Because we found $\sigma x, \sigma y < 60$ cm is a good value, this means that the 95% confidence interval for the x and y location of the robot is ± 120 cm, or almost the entire field. These experiments show that, although it is crucial to have a rough estimate of angle, it is not important for the robot to know where it is on the field.

Conclusion

In this paper, we reported on our work on the control system of soccer-playing Sony quadruped legged robots. We briefly described the components of the robots and the setup of the game. We then contributed an algorithm to control and balance the amount of localization information the robot has; to score goals without that information; to employ as much of that information as is available and to upgrade and degrade performance gracefully as availability changes; and to deal with special cases in which more urgent action is necessary. Finally, we presented the results of tests we ran to determine the best way to control the amount of localization information the robot has.

Results from our matches in RoboCup-99 at IJCAI-99, Stockholm show our algorithms to be effective. Our team won all but one of its games, and the one it lost was lost by only one goal. Our team was the only one in this year's league to score goals against opposing teams and never to score a goal against itself. Our goaltender was the only one in this year's league to score a goal.

Acknowledgements

We would like to thank Sony for providing us with wonderful robots and walking movements to work with. We would like to thank Jim Bruce for developing the vision system and Scott Lenser for developing the localization algorithm for our robots. Thanks to Tucker Balch for helpful discussions and suggestions on this work.

This research is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement numbers F30602-97-2-0250 and F30602-98-2-0135. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the Air Force or the United States Government.