CM-Trio '99

Manuela Veloso, James Bruce, Scott Lenser, and Elly Winner

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
mmv@cs.cmu.edu

1 Overview

The robots used in this competition were generously provided by Sony [3]. The robots are the same as the commercial AIBO robots except for slight hardware changes and programming capabilities. These autonomous robots are about 30cm long and have 18 degrees of freedom. The neck pans $\pm 90°$ allowing the robot to scan the field with its on board camera. Six uniquely colored landmarks are placed around the field (at the corners and center-line) to help the robots localize. Each team consists of three robots. Like our team last year, CMTrio-98 [5], we divided our team between two identical attackers and one goalie.

We divided our system into three main components: vision processing, localization, and behaviors. The vision system is responsible for calculating distance, angle, and confidence measures for all objects visible by the robot. The localization is responsible for calculating the position of the robot on the field given the movements executed and the landmarks seen (we did not implement goals for localization this year). The behaviors are responsible for taking this information and winning the game.

Results from our matches in RoboCup-99 at 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 itself.

2 Vision

The vision system processes images captured by the robot's camera to report the locations of the ball, the 6 unique location markers, the two goals, and the robots. The main steps in vision processing are: 1) capture an image and classify each pixel's color in hardware using predetermined color thresholds, 2) find connected regions of the same color, 3) merge close regions of the same color, 4) use geometric filters to remove false positives, 5) calculate distance and angle of objects in ego-centric coordinates.

The on-board camera provides 88x60 images in the YUV space at about 15Hz. Hardware Color classification is then performed on these images. The thresholds for color segmentation are created by a supervised learning method based upon hand labelled images captured from the dog's camera. This results in a new image indicating color class membership rather than raw camera colors. This image is run length encoded (RLE) for further processing.

The region finding [1] method employs a tree-based *union find* with path compression. The algorithm outputs a forest of disjoint stubby trees corresponding to a connected region in the image. We next extract region information. The bounding box, centroid, and size of each region is calculated incrementally in a single pass over the forest data structure. The regions are separated by color and sorted by size (putting larger, more important blobs first).

A problem with connected components is that a single row of pixels incorrectly classified can seperate an object into two components. We used a density based merging scheme to try to overcome this problem. We merge close regions of the same color if the resulting new region has a sufficiently high density.

The next step is to calculate the location of the various objects given the colored regions. Various top down and geometric object filters are applied to limit the occurrence of false positives and serve as the basis for confidence values. The largest orange blob below the horizon is labelled as the ball. The confidence value and distance is calculated based on image area and a circular object model. The field markers are detected as pink regions with green, cyan, or yellow regions nearby. The confidence is the ratio between the squared distance between the centers of the regions and the area of each region. The distance is calculated from the distance between the centers of the two regions. The goals are the largest yellow or cyan regions below the horizon. The very course distance approximation is based on the angular height of the goal in the camera image. The confidence is based on the aspect ratio of the goal in the image. The final objects detected are opponents and teammates. Due to the multiple complicated markers present on each robot, no distance or confidence was estimated and the marker regions are returned in raw form.

The system performed well in practice; it had a good detection rate and was robust to the unmodeled noise experienced in a competition due to competitors and crowds. The distance metrics and confidence values were also useful in this noisy environment.

3 Localization

Our localization algorithm is based upon a classical Bayesian approach which updates the location of the robot in two stages, one for incorporating robot movements and one for incorporating sensor readings. This approach represents the location of the robot as a probability density over possible positions of the robot. Our localization algorithm, called Sensor Resetting Localization (SRL) [4], is based upon a popular approach called Monte Carlo Localization (MCL) which represents the probability density using a sampling approach.

MCL [2] represents the probability density for the location of the robot as a set of discrete samples. The density of samples within an area is proportional to the probability that the robot is in that area. We calculated the robot's position from these samples by taking their mean. We estimated the uncertainty by calculating the standard deviation of the samples. We encountered some problems implementing MCL for the robot dogs. MCL took more samples to do global localization than we could actually run on the hardware. MCL also has problems dealing with our large modelling errors.

SRL is motivated by the desire to use fewer samples, handle larger errors in modelling, and handle unmodeled movements. SRL adds a new step to the sensor update phase of the MCL algorithm. If the sensor reading and locale belief state disagree, we replace some of our samples with samples consistent with the current sensor readings. In this way, we effectively through out our history and reset. Note that when tracking is working well no resetting is done and SRL behaves exactly the same as MCL.

This resetting step allows SRL to adapt to large systematic errors in movement by occasionally resetting itself. SRL is also able to recover from large unmodeled movements easily by using this same resetting methodology. Unexpected movements happen frequently in the robotic soccer domain we are working in due to collisions with the walls and other robots. Collisions are difficult to detect on our robots and thus cannot be modelled. We also incur teleportation due to application of the rules by the referee.

Robot movement was modelled as three Gaussians with hand measured parameters; one Gaussian for distance travelled, one for direction travelled, and one for heading change. Sensor readings were modelled as two Gaussians, one for distance and one for angle. Standard deviations were estimated by testing.

We used 400 samples in actual competition. We weren't quite capable of keeping up with real time this way if we saw a lot of markers, so we through out some sensor readings from time to time to catch up. The localization is accurate to about 10cm and 15° while the robot is looking around for markers and moving. Performance drops somewhat when the robot goes long periods of time without looking around for markers as often happens during play. We observed that the localization algorithm quickly resets itself when unmodeled errors such as being picked up occur.

4 Behaviors

Choosing behaviors for the robot is a difficult challenge. The robot must act under uncertainty and varying amounts of localization information. The robot can affect the amount of information available to it by actively localizing. Actively localizing involves stopping the robot and scanning for markers, a process taking 15–20 seconds. Stopping the robot reduces the computational load on the localization system allowing it to operate in real time. The behavior system has to balance the time spent localizing with time spent acting. Every moment spent looking around provides an opportunity to the opponent robots.

Our solution is structured as a finite state machine. Each state corresponds to a set of behaviors that all accomplish the same goal. Each behavior expects a different amount of information to be available. The best behavior that has all its required information available is chosen to be executed. In this way, the robot takes advantage of all the information that is available to it. We call this approach multi-fidelity behaviors [6].

Switching between action and localization is controlled by timeouts that switch the robot between states of the finite state machine. These timeouts make sure that the robot localizes occasionally and that the robot spends enough time acting. The robot needs to localize ocassionally to prevent the localization output

from drifting from reality without being detected. The timeout for action ensures that the robot doesn't spend all of its time localizing. The localization timeout is turned off during behaviors that do not require localization information.

Our behavior state machine has 5 main states: score, recover ball, search for ball, approach ball, and localize. Search for ball and approach ball do not require localization. Search for ball employs a random search method that alternates between walking forward/backward random distances and rotating in place a random number of degrees. Approaching the ball uses the visual input to approach the ball. If localization information is available, the robot attempts to approach a point behind the ball to save time. The scoring behavior circles the ball (while facing it) to get behind it and then pushes the ball towards the goal. If sufficient localization information is available, the robot: circles in the quickest direction, avoids circling into the wall, and does not bother visually acquiring the goal. The recover ball behavior backs up when the robot looses track of the ball. This optimizes for the common case of the robot loosing the ball by walking past it. The localization mode stops the robot and scans for markers.

We have specialized behaviors for kickoff and goal protection. At kickoff, we charge the ball to ensure the best ball position possible. The outcome of the kickoff often decided who would score next. Our goalie runs a specialized set of behaviors. The goalie scans for the ball from its home position in front of its goal. When the goalie sees that the ball is close enough, the goalie clears the ball and then returns to the home position using the goals as landmarks for navigation. When the goalie is clearing the ball, the goalie uses localization information to hit the ball off-center such that the ball heads towards the opponents half of the field. This tactic was sufficient to allow our goalie to clear a ball all the way into the opponents goal.

Acknowledgments: We thank Sony for providing the robots for our research. This research was sponsored by Grants Nos. DABT63-99-1-0013, F30602-98-2-0135 and F30602-97-2-0250, and by an NSF Fellowship. The content of this publication does not necessarily reflect the position of the funding agencies and no official endorsement should be inferred.

References

1. J. Bruce, T. Balch, and M. Veloso. Fast and cheap color image segmentation for interactive robots. 2000. *Proceedings of WIRE-2000.*
2. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of AAAI-99*, 1999.
3. M. Fujita, M. Veloso, W. Uther, M. Asada, H. Kitano, V. Hugel, P. Bonnin, J.-C. Bouramoue, and P. Blazevic. Vision, strategy, and localization using the Sony legged robots at RoboCup-98. *AI Magazine*, 1999.
4. S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of ICRA-2000*, 2000.
5. M. Veloso and W. Uther. The CMTrio-98 Sony legged robot team. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 491–497. Springer Verlag, Berlin, 1999.
6. E. Winner and M. Veloso. Multi-fidelity robotic behaviors: Acting with variable state information. 2000. *AAAI-2000.*