The CMTrio-98 Sony Legged Robot Team

Manuela Veloso and William Uther

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{veloso,will}@cs.cmu.edu

Abstract. Sony has provided a remarkable platform for research and development in robotic agents, namely fully autonomous legged robots. In this paper, we describe our work using Sony's legged robots to participate in the RoboCup'98 legged robot demonstration and competition. The robots are fully autonomous with on-board vision, control, and navigation. The challenges we addressed in this framework include the color calibration of the vision hardware, the landmark-based robot localization on the playing field, and the development of robot behaviors for the actual play of the game. The paper presents our approach and contributions to these issues. We apply machine learning techniques for automated color calibration and we develop effective vision-servoed navigation. We present our Bayesian localization algorithm. Team strategy is achieved through pre-defined behaviors. Our team of the Sony legged robots, CMTrio-98, won all of its games in the RoboCup-98 competition, and was awarded the first place in the championship.

1 Introduction

RoboCup-98 included a new demonstration league: the Sony legged robot league. The three participating teams, namely Osaka University, University of Paris 6, and Carnegie Mellon University, all used identical robots supplied by Sony Corp. These small autonomous legged robots provide a very challenging platform for robotic soccer.

The Sony legged robot as a robotic soccer player is a fully autonomous robotic system without global vision or wireless remote operation. The RoboCup-98 legged robot exhibition match was therefore a software competition between robots with the same hardware platforms.

The Sony legged robots use on-board color-based vision as their only sensing. The vision processor is past of the robots' hardware. It provides a robust eight-color discrimination when calibrated correctly. Robots need to act solely in response to the visual input perceived. The work to build our Carnegie Mellon team, namely CMTrio-98, was decomposed along the following aspects:

- Reliable detection of all of the relevant colors in the game: orange (ball), light blue (goal and marker), yellow (goal and marker), pink (marker), light green (marker), dark blue (teammate/opponent), and dark red (opponent/teammate).
- Active ball chasing: the robot actively interleaves searching for the ball and localization on the field to evaluate both an appropriate path to the ball and final positioning next to the ball.
- Game-playing behaviors: robots play attacking and goal keeping positions.

This chapter is organized as follows. Section 2 presents our machine learning algorithm to automatically learn the YUV thresholds for color discrimination. Section 3 describes our Bayesian probabilistic localization. Section 4 outlines the basic robots playing behaviors. Section 5 summarizes the CMTrio-98 RoboCup-98 demonstrations and games and concludes the paper.

2 Supervised Learning of YUV Colors

The Sony legged robot has specialized hardware for the detection of colors. This hardware quickly maps the high-resolution colors of YUV color space into one of eight *symbolic colors*. However, this hardware still requires pre-setting of the appropriate thresholds in YUV color space for the desired mapping. It is well known that color adjustments are highly sensitive to a variety of factors, such as lighting and shading. Given that the legged robots inevitably act under many different conditions, we developed a method to automatically acquire the necessary YUV color thresholds.

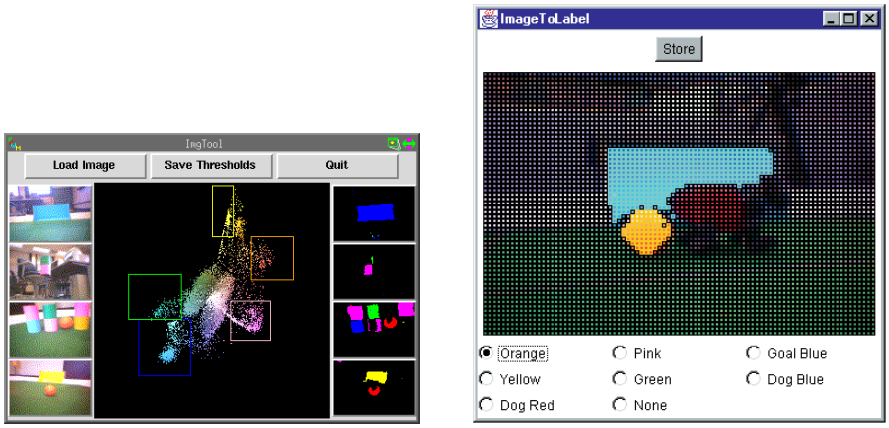
The YUV color space describes a color in terms of three numbers. The Y value describes the brightness of the color. The U and V values together describe the color (without brightness information). Initially we developed a tool to manually experiment with different boundaries in the UV plane for each symbolic color, and ignored the Y axis.[1] Figure 1 (a) shows some images taken through the robot's camera, the projection of the colors in those images onto the UV plane along with rectangles specifying which areas of the UV plane map to which symbolic colors and then the images with just the symbolic colors marked.

Based upon the experience using the UV plane tool, we understood that it would be necessary to adjust the color thresholds separately for different brightness levels. Instead of modifying the previous tool to handle the different Y values separately, we developed a classification algorithm to automatically adjust the thresholds to maximize the accuracy of the desired color detections.

Our algorithm relies on supervised classification using a set of training and testing images. By moving the robot to different positions on the field, we accumulate a series of images. For each image, we manually classify the regions of the different symbolic colors using an interface that overlays the original image and the supervised classification (See Figure 1 (b)). The result is a list of labelled pixels. Each has a position in YUV color space representing its actual color, and a label specifying which of the symbolic colors we think this is. The position in the image it came from is ignored.

Once the data has been labelled, the YUV color thresholds are learned separately for each symbolic color using a conjugate gradient descent based algorithm. For each symbolic color and Y value, the edges of a rectangle in UV space are adjusted to minimize sum-squared classification error in the labelled data. We can view each boundary of the rectangle as a function between either the U or the V value and the probability that this pixel is in the correct range. These

[1] This tool was developed by Kwun Han, for which we thank him.



(a) (b)

Fig. 1. The tools for color separation: (a) manual specification; (b) labelling data for automatic separation.

boundaries are each of the following form:

$$C = \begin{cases} 1, \text{if } x > a_i \\ 0, \text{otherwise}, \end{cases} \tag{1}$$

where C is the probability this pixel lies above the threshold, x is the current U or V value, and a_i is the value of this threshold.

To represent a rectangle in UV space, the probability that we are on the correct side of the individual thresholds was multiplied. In order to use gradient descent, we need to calculate derivatives for the model parameters, i.e., the rectangle boundaries, a_i. This is not possible for the step function shown. To solve this, each threshold was replaced by a sigmoid function:

$$C = \frac{1}{1 + e^{t(a-x)}}, \tag{2}$$

where C, x and a are as above, and t is a measure of the 'smoothness' of a threshold. We know which side of the rectangle each pixel should fall given its label, i.e. if the probability of being in the rectangle should be 0 or 1. Gradient descent is performed on the sum-squared error in probability. Initially t is small leading to very smooth thresholds. t is gradually increased over time, hardening the thresholds and making them more like the step functions used by the hardware.

In our experiments, we generally use about twenty images for training. The learning algorithm converges in about three hours achieving a high classification accuracy.

3 Bayesian Probabilistic Localization

In order to kick a ball towards a goal, it is necessary to know the direction to that goal. However, the Sony robots used during the competition did not have any built-in localization mechanism. Relying on dead-reckoning in the legged robot for localization is completely unrealistic because of variability in the gait. Additionally, because of the limited field of view of the robot's camera, it was usually not possible to simultaneously keep the ball and the goal in view at the same time. Our solution to this problem was to have the robot track its position on the field and the direction to the goal using a Bayesian localization procedure (e.g., [2, 1]).

To compensate for the highly unreliable dead reckoning, the field environment for RoboCup-98 includes several fixed colored landmarks. These landmarks, along with the goal itself, were used to localize the robot on the field, calculate its orientation on the field and to estimate the direction to the center of the goal.

3.1 Position Localization

For position/angle localization, the field was divided into a $20 \times 30 \times 8$ (X, Y, θ) state grid. We recorded for each state the probability that the robot was in that state. This probability distribution was updated as observations were made about the world, and as the robot moved in the world.

A table of values was chosen because of some of the distributions we wish to represent do not have a nice parametric form. For instance, given a uniform prior distribution, the observation of the angle between two markers gives a high probability circle through the state space that is not representable by a Gaussian distribution.

Incorporation of observations is based upon Bayes' Rule:

$$P(S_i|O) = \frac{P(S_i)P(O|S_i)}{\sum_j P(S_j)P(O|S_j)}, \tag{3}$$

where $P(S_i)$ is the apriori probability that the robot is in state S_i, $P(S_i|O)$ is the posterior probability that the robot is in state S_i given that it has just seen observation O and $P(O|S_i)$ is the probability of observing O in state S_i.

Incorporation of movement is based upon a transition probability matrix. Given a previous movement M, for each state the algorithm computes the probability that the robot will end up in that state:

$$P(S_i|M) = \sum_j P(S_j)P(S_j \Rightarrow S_i|M), \tag{4}$$

where $P(S_j)$ is the apriori probability of state S_j and $P(S_j \Rightarrow S_i|M)$ is the probability of moving from state S_j to state S_i given the movement M. It is assumed that the transition probabilities, $P(S_j \Rightarrow S_i|M)$, take into account any noise in M.

For example, imagine the robot sees an angle of 90° between two markers, turns to the left and then sees an angle of 90° between two more markers. Initially it does not know where it is - our prior distribution is flat. After its first observation, the projection of the state probability matrix onto the X, Y plane would be as shown in Figure 2(a). During the turn, the projection spreads over the X, Y plane - representing the increased uncertainty introduced by the dead reckoning as the robot turns (see Figure 2(b)). The second observation finally localizes the robot (see Figure 2(c)).

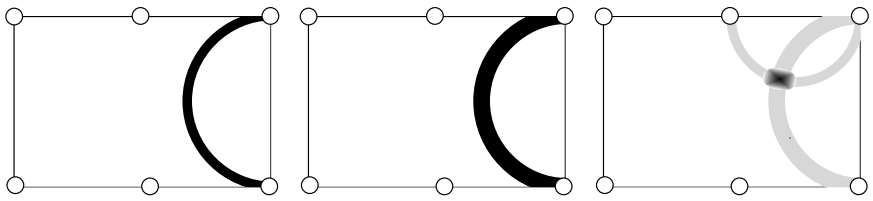


Fig. 2. The positioning probability: (a) after the 1st observation; (b) after a 90° turn; (c) after the 2nd observation.

In fact, there were two different types of observation used in our algorithm. The first was the angle between landmarks. These angles were pre-computed for each state. The second was the size of the landmarks in the robot's camera image. This size is linearly related to the inverse square of the distance to the marker. Once calibrated this was the most informative form of observation.

In order for this algorithm to be computationally tractable, one important approximation is needed. Probabilities that are close to zero are set to zero. Bayes' Rule never changes a zero probability, so these be checked for during updating and ignored. Once the robot is reasonably sure of its location most states have zero probability leading to a dramatic increase in speed.

3.2 Multiple Localization Levels

Having only 8 directions is not enough to accurately aim a ball. However due to both memory and time constraints, it is not possible to increase the angular resolution of the previous discretization. To increase the angular resolution, we introduced two more localization systems that ran in parallel with the one described above. All three used the same basic algorithm, but the number of states, the types of observations and the movement transition matrices are changed.

The second localization system tracked absolute angle on the field using a higher resolution (100 states) under the assumption that the robot was at the maximum likelihood X, Y location given by the first system. The third localization system tracked the angle of the goal. Again this was a high resolution angle and the system used the first two systems as well as vision and movement information.

3.3 When to Localize?

Our localization algorithm is passive in that it updates based on any information seen, without needing to control the robot to gather data. Unfortunately, sometimes this is not enough. When the maximum state probability is lower than a pre-defined threshold the localization becomes active. The robot searches for landmarks to improve its knowledge of its location. In our experiments, the robots localize themselves with high accuracy.

4 Role-Based Behaviors

Following up on our experience with our other RoboCup leagues, namely the small-size wheeled robots [5, 4], and the simulator league [3], we developed different behaviors based on positioning of the robots on the field. As of now, robots play two different roles, namely attacking and goal keeping.

The procedure used by an attacking robot consists of the following steps: (i) find the ball; (ii) localize attacking goal; (iii) position behind the ball, aligned with the goal; (iv) shoot. CMTrio-98 includes the simplest model for collaboration between the two attacking robots, namely the spatial division of the field. The field is divided in two slightly overlapping left and right longitudinal regions and the robots move in their own regions, as detected by their localization.

The procedure used by the goal keeping robot is a simplified version of a combination of the goal keeper and defender of the CMUnited-98 small-size team [4]. Its behavior consists of the following steps: (i) find the ball; (ii) remain close to the goal; (iii) move sideways aligned with ball; (iv) clear the ball when it gets close to it.

5 Discussion about Real World Conditions

All the algorithms described above were developed in our lab. While we had a complete mock up of the competition field, conditions at the RoboCup-98 site, namely La Cité des Sciences in Paris, were still significantly different from our laboratory conditions in a number of ways.

As we had anticipated the lighting conditions were significantly different between the lab and the competition field. With our automated color calibration we were able to effectively adjust for this. However, the program being automated caused one unforeseen difficulty. With some early versions of the vision hardware there are initialization problems that occasionally cause pictures taken by the pet to be shifted in color space until the vision is restarted. This shift, if undetected during the manual classification phase, causes incorrect parameters to be learned.

The second large change between the lab and competition conditions, which we anticipated but underestimated, was the effect of the audience. The robots, when placed on the field, need to look up at the localization markers which are on poles about the edge of the field. The background in these pictures is the audience - wearing whatever colored clothing they chose. This problem was

exacerbated because we could not take calibration pictures with the audience - only with the empty stadium.

Finally, there was an unforeseen interaction between the penalty system and our localization routines. When a team member was given a penalty it was picked up by the referee and placed elsewhere on the field. Our Bayesian localization system did not expect to be teleported at any stage and so the robot lost track of its location when this happened. Luckily we had an auto-reset mechanism if our internal model of location was significantly different from our sensor information, but this was not meant to be relied upon as much as it was.

6 Conclusion

In this paper, we reported on our work using the Sony quadruped legged robots to play robotic soccer. We briefly described the components of Sony's legged robots. We then presented our vision-based navigation, Bayesian localization, role-based behaviors.

Our color calibration, vision-based navigation, and localization algorithms showed to be effective. Our robots were the only ones to complete the RoboCup-98 physical challenge, where a single robot had to push the ball into the goal starting from a configuration where the the ball was not lined up with the goal from the robot's view point. Our CMTrio-98 team was also victorious in the competition games, winning 2-1 against both the French and Japanese teams.

References

1. W. Burgard, A.B. Cremers, D. Fox, D. Haehnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of AAAI-98*, Madison, WI, July 1998.
2. Alberto Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 1989.
3. Peter Stone and Manuela Veloso. Task decomposition and dynamic role assignment for real-time strategic teamwork. In *Proceedings of the ATAL workshop (Agent Theories, Architectures and Languages)*, Paris, July 1998.
4. Manuela Veloso, Michael Bowling, Sorin Achim, Kwun Han, and Peter Stone. The CMUnited-98 champion small robot team. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II.* Springer Verlag, Berlin, 1999.
5. Manuela Veloso, Peter Stone, Kwun Han, and Sorin Achim. CMUnited: A team of robotic soccer agents collaborating in an adversarial environment. In Hiroaki Kitano, editor, *RoboCup-97: The First Robot World Cup Soccer Games and Conferences.* Springer Verlag, Berlin, 1998.

This article was processed using the LaTeX macro package with LLNCS style