CM-Pack'01: Fast Legged Robot Walking, Robust Localization, and Team Behaviors

William Uther, Scott Lenser, James Bruce, Martin Hock, and Manuela Veloso

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

CM-Pack'01 came in second place at RoboCup-2001 in the Sony Legged League. This is our fourth year competiting in this league. We used a simplified architecture this year. Our vision, localization, and behaviors ran synchronously with the camera input while our motions remained an asynchronous process. We focused on reducing latency and increasing timeliness and completeness of environment modelling for this year. We used more structured software engineering this year than in previous years and found this very useful[4].

1 Vision

We used a similar vision setup to our entry in last year's RoboCup. We used an updated version of our low level vision library CMVision [1] to do all of the low level vision processing of color segmentation and region identification. This new version uses a rather more expressive representation for defining colors. The new representation is a 3D table indexed by the high bits of Y,U,V pixels provided by the camera. The table is created using hand-labelled images and an offline learning algorithm. The algorithm counts examples that fall in each table entry and picks the most likely color from those above a confidence threshold. Examples are spread across the table using exponential decay to ensure good generalization ability while limiting the algorithm's runtime to under 2 minutes on a PII 366MHz.

High level vision used the colored regions identified by the low level vision to find objects of interest. Most of the high-level vision was the same as last year. The estimate of marker distance was changed to project rays through the center of the two colored regions of the marker and find the distance at which the rays are 20cm apart in the vertical direction. Robot detection was added for this year. The first stage of robot detection was finding regions that had the right color to be parts of a robot. After collecting regions of robot color, nearby regions of the same color were merged if the resulting region had a sufficiently high density of robot colored pixels in its bounding box. This usually resulted in regions that represented the major patches of color on the robot. Next, the vision system compared regions that had vertical overlap to see if they should be merged. Regions of the same color were merged if they were close enough (as compared to their boudning box) and seperated mostly by colors found on robots' bodies (black and white). The resulting merged regions were treated as robots and had a distance assigned based upon pixel size (assuming the robot was orientated to provide maximum colored area).

2 Localization

We developed a new simple probability-based method for localization in CM-Pack'01 . The theory is similar to previous Bayesian localization techniques used by our CMU legged robot teams [6, 7, 3]. We keep a probability distribution over possible current locations, P_{loc}. When the robot sees a marker, it updates P_{loc} using Bayes' rule: $P_{loc|sensor} \propto P_{loc} \times P_{sensor|loc}$. When the robot moves, it translates P_{loc}, and then it smooths it to account for the noise in the odometry.

This year, P_{loc} was decomposed into two gaussians. There is one gaussian representing X, Y location, and a second representing θ assuming that the robot is at the mean X, Y location.

When the robot sees a marker the vision returns a distance estimate for that marker and an egocentric angle to the marker. We make the approximation assumption that the robot is at its maximum liklihood X, Y location in P_{loc} and that the sensor reading distribution is gaussian. Once this approximation is made, Bayes' rule can be applied algebraically. The actual implementation follows closely the math in [5].

One can view our algorithm as similar to [2], but with the major refinement that the distance we move our position estimate depends upon our current confidence. This year's system was both fast and accurate.

3 The World Model

The world model tracked objects on the field even when they were out of view. Specifically, it tracked the ball, the four goalposts and all the robots. Initially the world model was very similar to the localization system. During the competition, the section tracking the robots was replaced with a monte-carlo system.

The ball was tracked as a single gaussian in ego-centric coordinates. These coordinates were used so that if our location estimate jumped around the ball estimate would still be accurate. The ball standard deviation was increased over time when the ball was not visible.

We tracked both edges of both goals separately. Tracking the edges of the goals allowed us to decide when we were accurately lined up with the goal for shooting. Sanity checking on the distance between the goal edges, and the corespondance with the localization were important.

During the competition we switched to a monte-carlo representation for robot locations. We kept 200 'samples' for each robot color. 100 of these were reseved for samples with IR PSD distance measurements, the other 100 for samples with image size based distance estimates. Each time a robot was seen, its location was recorded as a sample, displacing the oldest sample so far. Any time the robot moved the samples were moved (they were stored in ego-centric coordinates). Any time the robot should have been able to see a sample and couldn't the sample was aged faster than normal. Samples were timed out after 2 seconds. This value was decreased if the original vision confidence was low.

4 Behaviors

CM-Pack'01 used a very simple behavioral system this year. The main behavior system was a simple series of functions. The high level functions called lower level functions directly. There was none of the complex behavior arbitration used in previous years. This set of functions issued commands to the walking system. The behavior system was fast enough that it could be called by the vision system for each frame.

Active Localization: One behavior we found particularly useful this year was our active localization behavior. This behavior had the robot actively fixate on markers. The routine tried to fixate on each of the markers it thought it should be able to see given its current estimate of field location. Markers were fixated in order from right to left and then back from left to right.

This was significantly better than just waving the head about. Firstly, the pause to fixate on markers allows a better distance estimate. Secondly, it was able to know that it should look up for markers and hence was able to localize more effectively near the edges of the field. Finally, it was able to move its head faster between markers because it wasn't expecting to see anything useful at that point.

The one problem with the algorithm to this point is that if the robot is confused about its location then it will be focusing in the wrong places. To overcome this, we made two changes. Firstly, if the robot happens in passing to see a surprising marker (one that localization would say is over 2 standard deviations from where we think it should be) then we fixate on it. Secondly, if we are focussing where we think a marker should be and we cannot see it then we increment a counter. If we fail to see a few markers in a row then we fall back to waving our head side-to-side.

Attack: Our attack strategy was simple. If you are between the ball and the goal then walk to one side of the ball. If you are at 90° to the goal then use the head kick to kick the ball sideways. If you would kick the ball between the goalposts then kick. Otherwise, dribble the ball towards the goal. The dribble will tend to line you up with the goal so that you can then kick. It also tries to dribble around other robots, both allies and opponents.

We also implemented some very simple ally avoidance. Each of the attackers was assigned a side of the field. If you could see both the ball and your ally, and you were over on your allies side of the field, then back off slightly. This was enough to significantly reduce 'leg-lock'[1].

Defense: Our goalie tried to stay on the line between the ball and a point in the middle of the goal area. It tried to position itself at the goal mouth on that line. When the ball was too close it called the attacker functions to kick the ball away from our goal.

[1] The robots when ignoring each other would tend to get their legs entangled. This slowed both robots down and is a major problem.

5 Motion

The motion system is given requests by the behavior system for high level motions to perform, such as walking in a particular direction, looking for the ball with the head, or kicking using a particular type of kick. The system for CM-Pack'01 was based loosely on the design in used in the previous year [3], but features a completely rewritten, more flexible implementation.

The walking system implemented a generic walk engine that can encode crawl, trot, or pace gaits, with optional bouncing or swaying during the walk cycle to increase stability of a dynamic walk. The walk parameters were encoded as a 49 parameter structure. Each leg had 11 parameters; the neutral kinematic position (3D point), lifting and set down velocities (3D vectors), and a lift time and set down time during the walk cycle. The global parameters were the z-height of the body during the walk, the angle of the body (pitch), hop and sway amplitudes, and the walk period in milliseconds. In order to avoid compilation overhead during gait development, the walk parameters could be loaded from a file on bootup. Using this interface, we developed a high performance walk with a maximum walking speed of 200 mm/sec forward or backward, 170 mm/sec sideways, or 2.1 rad/sec turning.

Additional motions could be requested from a library of motion scripts stored in files. This is how we implemented getup routines and kicks, and the file interface allowed easy development since recompilation was not needed for modifications to an existing motion. Adding a motion still requires recompilation, but this is not seen as much of a limitation since code needs to be added to the behaviors to request that that motion be used.

References

1. J. Bruce, T. Balch, and M. Veloso. CMVision (http://www.coral.cs.cmu.edu/cmvision/).
2. Bernhard Hengst, Darren Ibbotson, Son Bao Pham, and Claude Sammut. The UNSW United 2000 Sony legged robot software system. Published on Sony Internal Web Site, 2000.
3. S. Lenser, J. Bruce, and M. Veloso. CMPack: A complete software system for autonomous legged soccer robots. In *Autonomous Agents*, 2001.
4. Steve McConnell. *Rapid Development: taming wild software schedules.* Microsoft Press, 1996.
5. Ashley Stroupe, Martin C. Martin, and Tucker Balch. Merging gaussian distributions for object localization in multi-robot systems. In *Proc. of the Seventh International Symposium on Experimental Robotics (ISER '00)*. Springer-Verlag, December 2000.
6. 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.
7. M. Veloso, E. Winner, S. Lenser, J. Bruce, and T. Balch. Vision-servoed localization and behavior-based planning for an autonomous quadruped legged robot. In *Proceeding of AIPS-00*, 2000.