CMPack '00

Scott Lenser, James Bruce, and Manuela Veloso

Carnegie Mellon University, SCS
Pittsburgh, PA 15213-3891
{slenser,jbruce,mmv}@cs.cmu.edu

1 Introduction

This is a description of Carnegie Mellon University's entry in the Sony legged league of RoboCup 2000. See our web page for more details [4]. The main components of our system are: vision, localization, behaviors(including a basic world model), and motions. The main changes for this year are: basic world model, new behaviors/behavior architecture, walking and kicking motions. We placed third in the competition, losing only to the first place team.

2 Vision

The vision system is responsible for interpreting data from the robots' primary sensor, a camera mounted in the nose. The images are digitized are color segmented in hardware using color thresholds that are learned offline. The low level vision software then carries out the following steps:

- Connect neighboring pixels of the same color class to make regions.
- Connect nearby regions of the same color class into larger regions.
- Calculate statistics for the regions useful for high level vision.

Nearby regions are merged when the density occupied in their bounding area is above a threshold for that color class. The low level vision uses the CMVision librarry which is described in more detail in [3] and is freely available under the GPL [2]. After the low level processing is performed, the high level vision carries out the following steps for each type of object of interest (currently these include the ball, goals, markers, and other robots).

- Scan lists of regions for colors that include the object of interest.
- For each region or regions that may form the object, evaluate domain and geometric constraints to generate a confidence value.
- Perform additional filtering rules to remove false positives.
- Take the hypothesis with the highest remaining confidence value.
- Transform the object from image coordinates to ego-centric body coordinates (not yet implemented for the other robots).

We found our vision system to be generally robust to noise and highly accurate in object detection and determining object locations. However, like many vision systems it remains sensitive to lighting conditions, and requires a fair amount of time and effort to calibrate.

P. Stone, T. Balch, and G. Kraetzschmar (Eds.): RoboCup 2000, LNAI 2019, pp. 623-626, 2001.

3 Localization

Our localization system, Sensor Resetting Localization [6], uses a probabilistic model of the world to estimate the robots location on the field. The robots location is represented as a probability density over the possible positions and orientations of the robot. Since the probability density is in general a very complex function, we approximate the probability density by a set of sample points. The samples are chosen such that if $x\%$ of the samples are expected to be found in a particular area then the probability that the robot is in that area is $x\%$. Each sample point represents a particular location on the field at which the robot might be located. Localization is the process of updating this probability density. To make the computation tractable, we make the Markov assumption that the robots future location depends only on its present location, the motions executed, and the sensor readings. Updates are done in such a way that the expected density of sample points in a region is proportional to the probability of the robot's location being within that region. The localization system automatically resets itself if it notices to high of an error. For more detail including a probabilistic derivation of the algorithm, see the Sensor Resetting Localization paper [6].

4 Behavior System

Our behavior system is a hierarchical behavior-based system. The system is primarily reactive, but some behaviors have internal state to enable behavior sequencing. The input to the system is information about the objects seen (from the vision) and an estimate of the robots location (from the localization). The output of the behaviors is a motion to be executed. The motion can be a type of kick or getup to execute (discrete) or a direction to walk (continuous). The behavior system consists of three interconnected hierarchies for sensors, behaviors, and control. The sensor hierarchy represents all that is known about the world, including a basic world model that tracks objects' locations when they are out of view. The behavior hierarchy makes all of the robot's choices. The control hierarchy encodes everything the robot can do. Our system is similar to the system used by the FU-Fighters small size RoboCup team [1]. We loosened the sensor heirarchy and made behaviors responsbile for their own activation levels. This results in reduced coupling between the behaviors/sensors and different behavior levels.

The sensor hierarchy represents the knowledge that the robot has about the world. We divide the sensors into two categories, real sensors and virtual sensors. Virtual sensors represent a processed version of the real sensors that includes information that the behaviors are directly interested in. The virtual sensors serve to aggregate information and act as a model of the world complete with memory. Virtual sensors also avoid computing the same information twice in separate behaviors. The sensor hierarchy has a data structure for storing the sensor values and code to update the virtual sensors in the data structure from

the real sensors. The data structure output by the sensor hierarchy provides the input to the behavior hierarchy.

The behavior hierarchy and control hierarchies are tightly coupled and cooperate to decide on the action to perform. The behavior hierarchy consists of n behavior sets. Each behavior set represents a set of behaviors. The control hierarchy consists of n control sets. Each control set is a data structure representing all the actions the robot could perform. Behavior/control sets at different levels of the hierarchy operate at different level of detail. A behavior set at level k receives input from the the control set at level $k+1$ and the sensor hierarchy. The behavior set decides what action to perform and writes the decision into the more detailed control set at level k. For example, a behavior set contains a behavior for getting behind the ball. This behavior is activated when the control set above indicates it would help. The behavior uses the sensors to find out where the ball is and sets a goal for the next level down to run along an arc that gets behind the ball. Each behavior calculates its own activation and a combinator resolves conflicts to decide which behaviors are actually run. The combinator allows multiple behaviors to be run in parallel but assures that conflicting behaviors are never run together. Each behavior set takes an abstract description of the task to be performed makes it more concrete. The control hierarchy provides storage for the inputs and outputs of the levels of the behavior hierarchy. The lowest level of the control hierarchy is simply the commands to be sent to the motion module.

5 Motions

The motion componentallows the robot to walk, kick the ball, and get up after falling. The behavior system requests a walking motion in the form of velocity requests for x, y, and θ, a getup routine, or a kick. The motion system determines the required angles and PID values for each of the joints to carry out the command and executes it as soon as stability allows. A successful walking motion must make the robot travel on the desired path direction, while ensuring that the robot does not fall over. Also, the other motions must transition smoothly to and from walking.

The high level motion system consists of walks with separate stepping patterns (walking forward, turning left, turning right), several kicks for manipulating the ball, and four getup routines, one for each possible fall (front, back, left side, right side). The motion system was constructed as a state machine. The non-walking motions were specified as time variant functions to determine raw joint angles and kinematic targets for the legs.

Our walk is a quasi-static crawl gait. This gait is characterized by having 3 feet in contact with the ground at all times. Quasi-static means that ignoring noise and momentum, the robot can stop at any point in the walk without falling over. Our walk uses a fixed attitude of the body to prevent unwanted oscillation of the head. We constrain feet in contact with the ground to not move relative to the ground. We choose a path for the body to follow and the

locations of the feet are calculated using the offset of the contact point to the center of the body and an inverse kinematic model. We represent the path taken by the body using a Hermite spline [5]. These splines take the initial and target points as parameters, as well as derivatives at each, and generate a smooth trajectory to follow conforming to the control points/derivatives. Every time a foot is picked up, the motion system takes the most recent target walk request from the behavior system. It then determines the current position and velocity of the body (simply by consulting the current spline) and plots a new spline path from the current position/velocity to the target position/velocity one cycle later. Thus we have fully continuous body paths and velocities regardless of the sequence of requests by the behaviors and latency of only a single step before beginning to execute a command. Walk speed ranged from 100mm/sec to 0mm/sec, allowing fine speed control. The path of the feet in the air was represented as a Catmull-Rom spline with points chosen to ensure continuity of horizontal velocity upon foot contact and sufficient elevation to clear the ground.

Overall we found the walk to perform significantly better than other implementations of the same gait at the competition, as well as allowing for a simplified software architecture thanks to high level primitives such as splines. We demonstrated the fastest and most stable walk using the crawling gait. Future goals include adapting our approach to other gaits such as trotting (two feet in the air at a time), which allow much higher speeds but which require dynamic stability.

6 Conclusion

We implemented a highly competent robot team. We plan to incorporate more machine learning into our robots to improve performance, robustness, and development time. We also plan to work on better knowledge of the locations of the other robots. We'd like to thank Sony for providing the excellent robots to work with.

References

1. S. Behnke, B. Frtschl, R. Rojas, et al. Using hierarchical dynamical systems to control reactive behavior. In *Proceedings of IJCAI-99*, pages 28–33, 1999.
2. J. Bruce, T. Balch, and M. Veloso. CMVision (http://www.coral.cs.cmu.edu/cmvision/).
3. J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of IROS-2000*, 2000.
4. J. Bruce, S. Lenser, and M. Veloso. CMPack '00 (http://www.cs.cmu.edu/~robosoccer/legged/).
5. J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley, Reading, Massachusetts, second edition, 1990.
6. S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of ICRA-2000*, 2000.