Cerberus'15 RoboCup SPL Team Description

H. Levent Akın, Okan Aşık, Binnur Görer, Bahar İrfan, Metehan Doyran, Berna Erden, Kadriye Yasemin Usta

Boğaziçi University, Department of Computer Engineering
34342 Bebek, Istanbul, TURKEY
{akin, okan.asik, binnur.gorer, bahar.irfan, metehan.doyran, berna.erden6, kadriye.usta}@boun.edu.tr

1 Introduction

The Cerberus team made its debut in RoboCup 2001 competition. This was the first international team participating in the league as a result of the joint research effort of Boğaziçi University (BU), Istanbul, Turkey and Technical University Sofia, Plovdiv branch (TUSP), Plovdiv, Bulgaria. The team competed in Robocup 2001-2014 except the year 2004. Since 2005, Boğaziçi University is maintaining the team. In 2005, despite the fact that it was the only team competing with ERS-210s (not ERS210As), Cerberus won the first place in the technical challenges.

Through the years, the members of the team have done many PhD, MS, BS Thesis studies related to SPL and published more than 40 papers in journals and international conferences, including the RoboCup Symposia [1].

Until 2015, we have been developing all required software modules as a part of general robotics framework in our lab. However, recently team has focused on providing hands-on experience for undergraduate students. Therefore, as a result of this effort, this year we started to use B-Human code architecture and adapted our cognition code base for B-Human infrastructure [1].

The organization of the rest of the paper is as follows. In Section 2, the details of the vision module are provided. Self localization method is described in Section 3. Various approaches we use for the planning module are described in Section 4. Finally, We also present our coach robot in Section 5.

2 Vision

2.1 Image Processing and Perception

The purpose of the perception module is to process a raw image and extract available object features from it. The input to the module is the image in YUV422 format and the output is the range and bearing values of the perceived objects and landmarks.

[1] The full list of Cerberus publications are available here: http://robot.cmpe.boun.edu.tr/ cerberus/wiki/doku.php/publications

Color Quantization We previously utilize a Generalized Regression Neural Network (GRNN) [2] for mapping the real color space to the pseudo-color space composed of a smaller set of pseudo-colors, namely, white, green, blue, orange, red, and "ignore". However, due to high training time, we switched to decision tree trained using labeled images in RoboCup 2014. However, the success of decision tree is generally determined by the chosen labeled images which cannot be guaranteed on competition site. Therefore, we use color table approach developed by B-Human team [1]. This approach creates a pre-calculated color table by color thresholding in HSV color space.

Scanline Based Perception Framework Considering that the cameras of the Nao robots provide higher resolution images and the processors are slower, it becomes infeasible to process each pixel to find the objects of interests in the image due to computational efficiency and real-time constraints. Therefore, scan lines are used to process the image in a sparse manner, hence speeding up the entire process.

The process starts with the calculation of the horizon based on the pose of the robot's camera with respect to the contact point of the robot with the ground, that is the base foot of the robot. After the horizon is calculated, scan lines that are 5 pixels apart from each other and perpendicular to the horizon line are constructed, such that they originate on the horizon line and terminate at the bottom of the image. The first step after that is to scan through these scan lines to find where the green field starts, which is done by checking for a certain number of consecutive green pixels along the line. Of course that results in a green region where all non-green parts that are close to the edges of the field ignored, such as the goal posts and balls that are on the border lines. In order to not lose information about those important objects, a convex-hull is formed for the starting points of the green segments. That way, we define the real green field borders where all objects of interests fall inside; hence, we can basically ignore, say all orange regions, that are outside the field borders. That provides a natural way of pruning false percepts without having to process them beforehand. After the field borders are constructed, the shorter scan lines are extended back to these borders, so that it is possible to use them to detect the goal posts and balls that are close to the borders.

After all these constructions and corrections, each scan line is traced to find colored segments on them. After only one pass over these scan lines, we end up with groups of segments with the colors we are interested in, namely, orange, white, blue. The next step is to build regions from these segments, based on the information on whether two consecutive segments "touch" each other, that is they are on two consecutive scan lines and either of them has a start or end point within the borders of the other one. Two consecutive touching segments are merged into a single region. For white segments though, there are some additional conditions, such as change in direction and change in length ratio. These additional constraints guarantee that all field lines are not merged into a single, very big region, but instead into smaller and more distinctive regions. After the construction of these regions, they are passed to the so called the *region analyzer* module to be further filtered and processed for the detection of the ball, the field lines and intersections of them, and the goal posts.

Robot Perception We percept robots on the field by scanning non green areas in the image. We apply certain sanity checks to validate the distance of robot from field boundary. We also use the size of robots to create an obstacle boundary and discard all possible robots in that boundary. We determine the color of the robot by scanning color of shirt on robot area as seen in Figure 1.

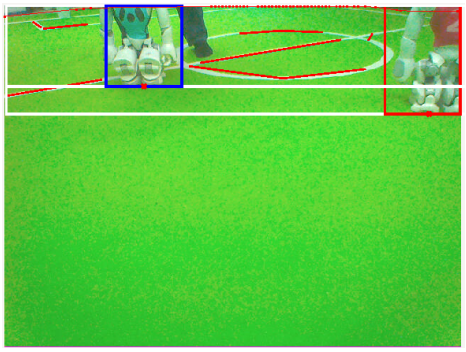

Fig. 1. Robot perception on the image.

White Goal Perception We use the structure of the field to create possible spots for goal bars. The distance of the goal bar from the end of field is fixed. We detect the end of the field and measure the distance of objects in the field. If we get an object closer to the distance between goal bar and the end of the field, we assign first green pixel as the bottom of the goal bar as seen in Figure 2. Although, this method works reasonably well, it susceptible to error for detection of field border. Therefore, we are also working to use edge detection to validate possible goal bar spots.

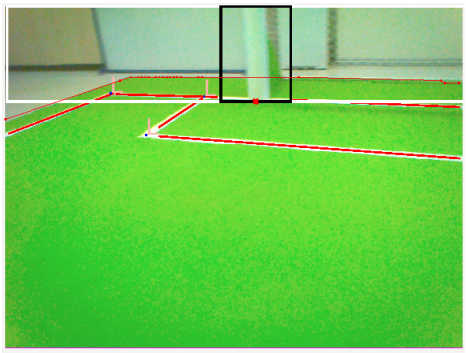

Fig. 2. White goal perception on the image.

2.2 World Modeling and Short Term Observation Memory

The perception module of Cerberus provides instantaneous information. While the reactive behaviors like tracking the ball with the head requires only instantaneous information, other higher level behaviors need more than that.

The planning module requires perceptual information with less noise and in a more complete manner. The world modeling module should reduce sensor noise and complete the missing state information by predicting the state. This is a state prediction problem and we use the most common approach in the literature, the Kalman Filter [3], for solving this problem.

In our setting, the observations are the distance and the bearing of the objects with respect to the robot origin, and the state we want to know consists of the actual distance and bearing information. In addition to that, for dynamic objects like the ball, the state vector also includes distance change and bearing change information to aid prediction.

For any object, the observation is $z = \{d, \theta\}$ where d and θ are distance and bearing, respectively, to the robot origin. For the stationary objects, the state is $m = \{d, \theta\}$ and the state evolution model is $m^1_{k+1} = I \times m_k$ and $z_k = I \times m_k$ where k is time and I is the unit matrix.

For the dynamic objects, the observation is the same but the state is represented as $m = \{d, \theta, d_d, d_\theta\}$ where d_d is the change in distance in one time step and d_θ is the change in bearing likewise. The state evolution model is:

$$\begin{pmatrix} d_{k+1} \\ \theta_{k+1} \\ d_{d,k+1} \\ d_{\theta,k+1} \end{pmatrix} = \begin{pmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{pmatrix} \begin{pmatrix} d_k \\ \theta_k \\ d_{d,k} \\ d_{\theta,k} \end{pmatrix}$$

and the observation model is:

$$\begin{pmatrix} d_{k+1} \\ \theta_{k+1} \end{pmatrix} = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \end{pmatrix} \begin{pmatrix} d_k \\ \theta_k \\ d_{d,k} \\ d_{\theta,k} \end{pmatrix}$$

As can be observed from the model specifications, we omit the correlation between the objects and use filter equations for each object separately. If an object is not observed for more than a pre-specified time step, the belief state is reset and the object is reported as *unknown*. For our case, this time step is 270 frames for stationary objects and 90 frames for dynamic objects.

In the update steps, odometry readings are used. The odometry reading is $u = \{d_x, d_y, d_\theta\}$ where d_x and d_y are displacements in egocentric coordinate frame and d_θ is the change in orientation. When an odometry reading is received, all the state vectors of known objects are geometrically re-calculated and the associated uncertainty is increased.

The most obvious effect of using a Kalman Filter is that the disadvantage of having a limited field of view is reduced. As the robot pans its head, it can be aware of distinct landmarks which are not in the same field of view at the same time.

3 Self Localization

Cerberus employs vision based Monte Carlo Localization (MCL). In the MCL algorithm, the belief state is represented by a particle set and each element represents a possible pose of the robot. We use MCL with a set of practical extensions (X-MCL) which is detailed in [4]. Until last year, we used the output of the world modeling module as input to the localization module. Namely the filtered landmarks are used as observations for the localization module. However, this year we are modifying the algorithm to filter unidentified observations.

The new approach is inspired from FastSLAM [5] algorithm and Multi-Hypothesis tracking [6]. In FastSLAM, each particle has its own world model (i.e. map). In Multi-Hypothesis tracking, there are multiple Gaussians where each relies on a different data association sequence and their numbers are limited by pruning.

In our localization approach, we define landmark groups which a non-unique observation might be observed from. For example a T type field line intersection might be observed from 6 different landmarks. Similarly a goal bar observation might come from left or right goal bar. We define a label for each non-unique observation which indicates the identity in its group. We augment the discrete label variable to the particles. Each particle now represents robot pose and a label variable associating non-unique observations to landmarks in the map. This model is an instance of *Switching Observation Model* [7]. The particles with wrong labels eventually die in the resampling steps.

To overcome the unified goal bar color problem, with the assumption that initial position of the robot is known, the model described above works with minimal change. For the kidnapping situations, we plan to develop a binary hypothesis based approach. The methodology is as follows. After kidnapping (or falling), the robot makes an assumption about the side of the first observed goal bar, and perform localization normally. After that it simultaneously tries to validate this hypothesis based on robot observations, and incoming messages from the teammates.

3.1 Motion

For bipedal locomotion, we use two different walking engines. While the first engine is developed in the lab, we use the B-Human walking engine and motion infrastructure [1].

In the first bipedal walking algorithm, we defined two important features for each leg; leg extension, and leg angle. Leg extension is the distance between the hip joint and the ankle joint. It determines the height of the robot while moving. Leg angle is the angle between the pelvis plate and the line from hip to ankle. It has three components; roll, pitch, and yaw. Using these features helps us have more abstract calculations for the motion.

In order to find the leg angle and foot angle features, motion at each step is divided into five sub-motions; *shifting*, *shortening*, *loading*, and *swinging*.

In the shifting sub-motion, lateral shifting of the center of mass is handled. The second important sub-motion is the shortening signal and it is not always applied. During the shortening phase, both a joint angle for the foot and a part of the leg extension value are calculated. The third sub-motion of the step is loading which is also not always

applied. In swinging part, the leg is unloaded, shortened and moved along the way of motion which reduces the stability of the system considerably. At the end, the corresponding parts of the sub-motions are added, and the values for the motion features are calculated.

After determining a feasible parameter set by hand, we applied an optimization algorithm, *Evolutionary Strategies*, to fine-tune the walking motion. Although both speed and balance is used in the fitness function, our walk engine is an open-loop engine during the game and it is vulnerable on the accumulation of balance disturbance. More details about this walking strategy can be found in [8].

4 Planner

The soccer domain is a continuous environment, but the robots operate in discrete time steps. At each time step, the environment, and the robots' own states change. The planner keeps track of those changes, and decides the new actions. The main aim of the planner is to sufficiently model the environment and update its state. Additionally, the planner should provide control inputs according to this model. Previously, we developed a Dec-POMDP based planner. Currently, we use a finite state machine based planner architecture as explained in Section 4.1. We also extend the behaviors by integrating market-based dynamic role allocation mechanism as explained in Section **??**.

4.1 Finite State Controller based Planner

The Finite State Controller (FSC) based planner makes use of the formal model of the problem. At every planner step, the robot is in a particular state and we want our robot to take the best action in that state. FSC is based on the conventional Hierarchical Finite State Machine model, however, we changed some aspects to use it in high-level robot planning. There are states which correspond to the environment states. Transitions take place according to the current environment observations. There are also actions which will be taken when the robot is at a particular state. The robot can execute many actions in a particular state and these actions may override each other according to their priority. The most powerful part of this planner architecture is that once we code particular transitions or actions, they can be reused in different behaviors. We have developed a GUI tool called *FSC Designer* for this purpose [9]. *FSC Designer* enables easy development of finite state controller based behaviors by using already developed *Transition* and *Action* constructs as seen in Figure 3.

4.2 Dynamic Role Allocation

We define a team formation for four field players. A place in the formation corresponds to a role, such as striker, attacker, defender and winger. Robots are calculating their costs for every role. Then, every robot chooses the one which has the lowest cost for itself. The costs are calculated according to the distance of the robot to the position of the role on the field. This calculation and assignment is done in distributed fashion by using teammate messages shared between players. The exact position of the robots are

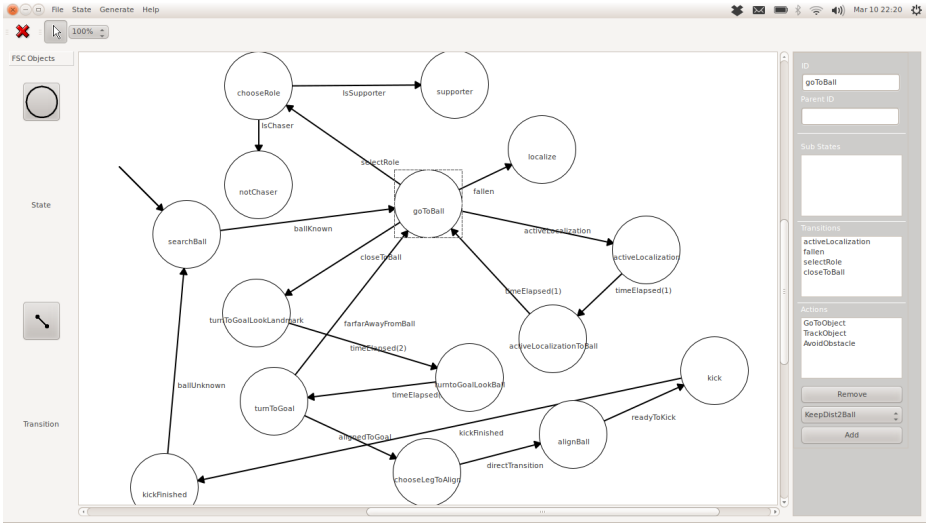

Fig. 3. Snapshot of *FSC Designer*

determined according to the position the ball. Since, it would result in oscillation in role assignment, allocation mechanism is run at once every second.

5 Coach Robot

We developed a coach robot to assist the team with strategic decisions. The coach robot percepts robot positions on the field and create a world model of all robots. It calculates soccer metrics [10] from this world model and chooses one of the predefined team formations. Currently, we have three formations, offensive, defensive and neutral. These different formations determines how likely team will be on other team's half field. The calculated formation is carried out by the dynamic role allocation module of the planner.

6 Conclusion and Future Works

In conclusion, we aim to develop successful autonomous systems which are able to perform well in adversarial environments. We summarize our methods for solving certain problems in RoboCup Standard Platform League. In future, we aim to solve multi-agent planning, and kidnapping problems which are the top challenges of SPL. We perform research on the application of *Dec-POMDP* methods for robot soccer [11]. For solving the kidnapping problem, and in general the localization in symmetric field, we work on methods which effectively merge world models of teammates.

Acknowledgments

This work is supported by Boğaziçi University Research Fund through project 7631.

References

1. Thomas Röfer, Tim Laue, Judith Müller, Dennis Schüthe, Arne Böckmann, Dana Jenett, Sebastian Koralewski, Florian Maaß, Elena Maier, Caren Siemer, Alexis Tsogias, and Jan-Bernd Vosteen. B-human team report and code release 2014, 2014. Only available online: `http://www.b-human.de/downloads/publications/2014/CodeRelease2014.pdf`.
2. Ethem Alpaydın. *Machine Learning*. MIT Press, 2004.
3. Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, Chapel Hill, NC, USA, 1995.
4. K. Kaplan, B. Çelik, T. Meriçli, Ç. Mericli, and H. L. Akın. Practical extensions to vision-based monte carlo localization methods for robot soccer domain. *RoboCup 2005: Robot Soccer World Cup IX, LNCS*, 4020:420–427, 2006.
5. M. Montemerlo. FastSLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association. In *CMU Robotics Institute*, 2003.
6. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.
7. Francois Caron, Manuel Davy, Emmanuel Duflos, and Philippe Vanheeghe. Particle filtering for multisensor data fusion with switching observation models: Application to land vehicle positioning. *IEEE Transactions on Signal Processing*, 55(6-1):2703–2719, 2007.
8. B. Gökçe and H. L. Akın. Parameter optimization for a signal-based omni-directional biped locomotion using evolutionary strategies. In *RoboCup 2010 Symposium*, Singapore, June 25, 2010.
9. O. Aşık and H. L. Akın. FSC Designer: A Visual FSM Design Tool for Robot Control. In *3rd Computer Science Student Workshop (CSW'12)*, 2012.
10. Cetin Mericli and H. Levent Akın. A Layered Metric Definition and Evaluation Framework for Multirobot Systems. In Iocchi, L and Matsubara, H and Weitzenfeld, A and Zhou, C, editor, *ROBOCUP 2008: ROBOT SOCCER WORLD CUP XII*, volume 5399 of *Lecture Notes in Computer Science*, pages 568–579, 2009.
11. O. Aşık and H. L. Akın. Solving Multi-Agent Decision Problems modeled as Dec-POMDP: A Robot Soccer Case Study. In *RoboCup 2012 Symposium, June 24, 2012, Mexico City*, 2012.