Cerberus'05 Team Description Paper

H. Levent Akın[1], Hatice Köse[1], Çetin Meriçli[1], Kemal Kaplan[1], Buluç Çelik[1] and Tekin Meriçli[2]

[1]Boğaziçi University
Department of Computer Engineering
34342 Bebek, Ístanbul, TURKEY
{akin, kose, cetin.mericli, kaplanke, buluc.celik}@boun.edu.tr

[2]Marmara University
Department of Computer Engineering
Göztepe, Ístanbul, TURKEY
tmericli@eng.marmara.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 result of the joint research effort of a group of students and their professors from Boğaziçi University (BU), Istanbul, Turkey and Technical University Sofia, Plovdiv branch (TUSP), Plovdiv, Bulgaria. The team competed in Robocup 2002 and Robocup 2003. Currently Boğaziçi University is maintaining the team and it is the only team participating from East Europe.

Boğaziçi University has a strong research group in AI. The introduction of Robocup as a unifying theme for different areas of study in autonomous robots has attracted many talented students and has accelerated research efforts with many publications. Currently, the department has teams both in Robocup Sony four legged and rescue simulation leagues and in FIRA Mirosot leagues.

2 The Proposed Approach

The entire software system of Cerberus was designed and developed from the scratch for this year. As opposed to the previous years, we began with developing a framework which makes all of our modules platform and hardware independent and allows us to transfer from or to the robot any input, output or intermediate data of the modules. This infrastructure brought us a considerable speedup during development and testing. The first goal of our team is to develop a platform where we can combine research and development techniques with software engineering methodologies. After careful inspection of previous works of our team and other teams, we decided to divide the project into two main parts.

- Cerberus Station
- Cerberus Player

2.1 Cerberus Station

This is the offline development platform where we develop and test our algorithms and ideas. The record and replay facilities allow us to test our implementations without deploying the code on the robot each time. It is developed using Microsoft .NET technologies and contains a set of monitors which enable visualizing several phases of image processing, localization, and locomotion information. It is possible to record live images, classified images, found regions, perceived objects and estimated pose on the field in real time to a log file and replay it in different speeds or frame by frame. Cerberus Station also contains a locomotion test unit in which all parameters of the motion engine and special actions can be specified and tested remotely. For debugging purposes, a telnet client and an exception monitor log parser are also included in station. Since each sub-module of the robot code is hardware independent, all modules can be tested and debugged in station. This hardware and platform independence provides great save on development time when combined with advanced raw data logging and playback system. Cerberus Station communicates with the robot via TCP and uses a common serializable message structure for information exchange.

2.2 Cerberus Player

Cerberus Player is the part of the project that runs on the robots. Most of the classes in Cerberus Player are implemented in a platform independent manner, which means we can cross-compile them in various operating systems like OPEN-R, Windows or Linux. Although, robot dependent parts of the code are planned to run only on the robot, a simulation system for simulating locomotion and sensing is under development. The software architecture of Cerberus Player consists of four objects:

- Core Object
- Locomotion
- Communication
- Dock Object

Core Object The main part of the player code is the Core Object. This object coordinates the communication and synchronization between the other objects. All other objects are connected to this object. Core object takes camera image as its main input and sends corresponding actuator commands to the locomotion engine. Core Object is the container and hardware interface of Vision, Localization and Planner modules. This combination is chosen because of the execution sequence of these modules. All of them are executed for each received camera frame and there is an input-output dependency and execution sequence vision $\rightarrow$ localization $\rightarrow$ planner.

Communication Object Communication object is responsible for receiving game data from the game controller and for managing robot-robot communication. Since the TCPGateway object which was the only communication interface allowed so far will no longer be supported, both the game controller and robot-robot communication infrastructure have been rewritten. They both use UDP as the communication protocol.

Dock Object Dock object is the object which manages the communication between a robot and the Cerberus Station. It redirects the received messages to Core Object and sends the debug messages to the station. Dock object uses TCP to send and receive serialized messages to and from Cerberus Station.

2.3 Modules in Cerberus

Core modules in Cerberus are vision, localization, planner and locomotion and due to our development policy, all of them are platform-independent so they can be adapted to any platform supporting standard C++ easily by writing a wrapper class for interfacing.

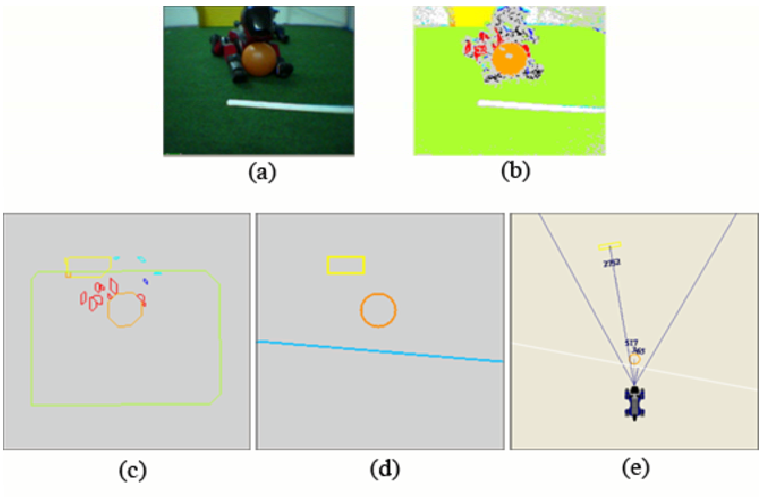

Fig. 1. Phases of image processing. a) Original image, b) Color classified image, c) Found blobs, d) Percepted objects e) Egocentric view

Vision Module Vision module is responsible for information extraction from received camera frame. The vision process starts with receiving a camera frame and ends with an egocentric world model consisting of a collection of visual

percepts as shown in Fig. 1. Vision module is written from the very scratch this year and several novel approaches for image processing have been introduced.

Color Classification: First, we have decided not to use previously implemeted color classification methods like decision trees and nearest neighbor [1]. Instead, we have implemented a Generalized Regression Network (GRNN) [2] for color generalization. After labeling a set of images with proper colors, a GRNN is trained with the labeled data and after the training phase, the network is simulated for the input space to generate a color lookup table for four bits (16 levels) of Y, six bits (64 levels) of U and six bits of V. The resultant color lookup table is very robust to luminance changes and allows our vision system to work without using any kind of extra lights other than the standard ceiling fluorescents.

Object detection: The classified image is processed for obtaining blobs. We use a new approach. Instead of using run length encoding (RLE), we use an optimized region growing algorithm that performs both connected component finding and region building operations at the same time. This algorithm works nearly two times faster than the well known *RLE-Find connected components-build regions* approach. Another novel approach used is the concept of a bounding octagon of a region. Since the robot must turn its head in order to expand its field of view, it is necessary to rotate the obtained image according to the actual position of the head. However, since rotation is a very expensive operation, it is not wise to rotate the entire image. For this reason typically only the identified regions are rotated. Since octagons are more appropriate for rotation than boxes, using octagons instead of boxes to represent regions reduces the information loss due to rotation.

Our vision module employs a very efficient partial circle fit algorithm for detecting partially occluded balls and the balls which are on the borders of the image. Since accuracy in estimation of ball distance and orientation is needed mostly in cases when the ball is very close and it is so often that the ball can only be seen partially in such cases, having a cheap and accurate ball perception algorithm is a must.

Line perception process is an important part of the vision module, since it provides important information for the localization module. The sample images from line perception process are shown in Fig. 2. The proposed approach is the following:

- Hough transform is applied on the white pixels which are close enough to green pixels using Robert's Cross on their Y band as the first operation.
- Two thresholds are used to check each entry in the table prepared in Hough transform. First threshold is minimum acceptable value for the line's entry, whereas the second one is minimum acceptable value for the sum of entries of the line and its neighbors. For a line entry to be accepted, it should be a local maximum as well.
- For a chosen line, to decrease the quantization error, the weighted average of its angle and perpendicular distance are taken, where weights are the values in the Hough transform table.

– Now, we have a more or less fine tuned line, but it is still the border of the field line. Especially in images where field lines are close to the camera, the lines occupy a thick region. The selected line is shifted along its normal vector orientation. The amount and the direction of the shift are calculated by following the normal line at different intervals.
– The lines are rotated according to the pan and the tilt of the camera.
– Then, the lines are mapped to the real 3D field using geometrical transformations.
– Once the lines are mapped to the field, they are with respect to the camera. As they need to be with respect to the chest, they are transformed accordingly.
– Finally, extra copies of the same line, which is a rare but possible situation, are eliminated.

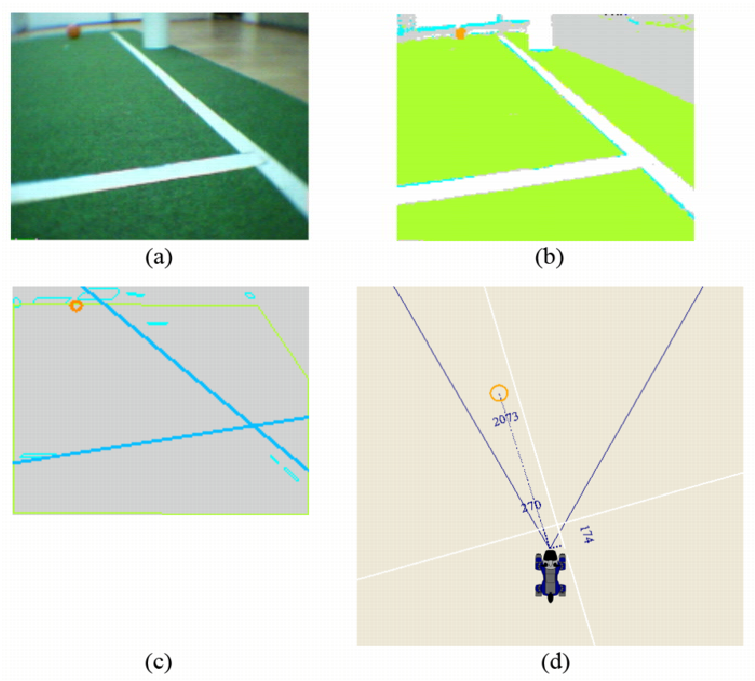

Fig. 2. Phases of Line Detection. a) Original image, b) Color classified image, c) Perceived lines e) Egocentric view

The vision module is one of the fastest vision systems having the features described above developed on AIBOs. On our current robots (ERS-210 with 200 MHz processor) a frame is processed in approximately 50 ms which provides a 20 frames per second speed.

Localization The localization method used by Cerberus team is the Reverse Monte Carlo Localization (R-MCL) method. This is a self-localization method for global localization of autonomous mobile agents in the robotic soccer domain, which proposes to overcome the uncertainty in the sensors, environment and the motion model [3], [4], [5]. This is a hybrid method based on both Markov Localization (ML) and Monte Carlo Localization (MCL) where the ML module finds the region where the robot should be and MCL predicts the geometrical location with high precision by selecting samples in this region (Fig. 3). The method is very robust and requires less computational power and memory compared to similar approaches and is accurate enough for high level decision making which is vital for robot soccer.

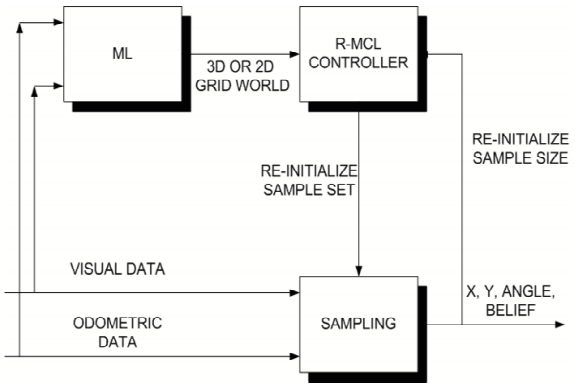

Fig. 3. R-MCL Working Schema

This year, we have also developed a standard vision based Monte Carlo Localization for comparing our localization system. During the development of this localization system, a few extensions have emerged. Briefly, our new approach proposes a set of practical extensions to the vision-based Monte Carlo localization for RoboCup Sony AIBO legged robot soccer domain. The main disadvantage of AIBO robots is that they have a narrow field of view so the number of landmarks seen in one frame is usually not sufficient for geometric calculation. MCL methods have been shown to be accurate and robust in legged robot soccer domain but there are some practical issues that should be handled in order to maintain stability/elasticity ratio in a reasonable level. In other words, the fast convergence ability is required in case of kidnapping. But on the other hand, fast convergence can be vulnerable when an occasional bad sensor reading is received. The first extension to overcome these problems and compensate for the errors in sensor readings is using inter-percept distance as a similarity measure in addition to distances and orientations of individual percepts (Static objects with known world frame coordinates on the field). Another extension is to use number of

perceived objects to adjust confidences of particles. The calculated confidence is reduced when the number of perceived objects is small and increased when the number of percepts is high. Since overall confidence of a particle is calculated as the multiplication of likelihoods of individual perceptions, this adjustment prevents a particle from being assigned with a smaller confidence value calculated from a cascade of highly confident perceptions where a single perception with lower confidence would have a higher confidence value. The third extension is related with the resampling phase. The number of particles in successor sample set is determined proportional to the last calculated confidence of the estimated pose. Finally, the window size in which the particles are spread into is inversely proportional to the confidence of estimated pose [6].

Planner and Behaviors We have used finite state machines (FSM) in our planner module in the past years. However, since the soccer domain is a continuous environment and the sensory input, which determines the state transitions, is quite noisy, discrete planners like FSM suffers from oscillations and quantization errors. Therefore, we adopted a modified version of Potential Field Planner (PFP) previously used for wheeled mobile agents in our lab [7]. Potential Fields is a method of planning robot trajectories based on combining the vector fields induced by mapped objects, such as repulsive fields from obstacles and attractive fields from goals. In the implementations of potential fields for robot control, only the force vector for the current robot position is calculated for each object or schema. The sum of these vectors is then used to control the instantaneous motion of the robot. The process is then repeated from the next robot position to handle any new perceptual information that may have become available.

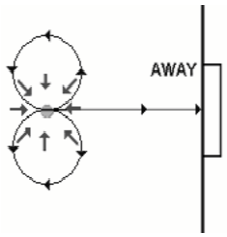

Fig. 4. Ball Field

Most of the objects on the field only require an impulsive force for preventing collision. Nevertheless, we use an attractive force to move a robot to a specific location. On the other hand, the attractive simple fields use a constant force directed to the center of the field. The field generated by the ball is combination of two potential fields. The first one is a simple attractive field. The second one is a special circular field located above and below the line between the ball and the center of opponent goal as shown in Fig.4. The direction of the field

is perpendicular to the line segment between the robot and the center of the nearest circle. On the other hand, the magnitude of the simple attractive field is constant.

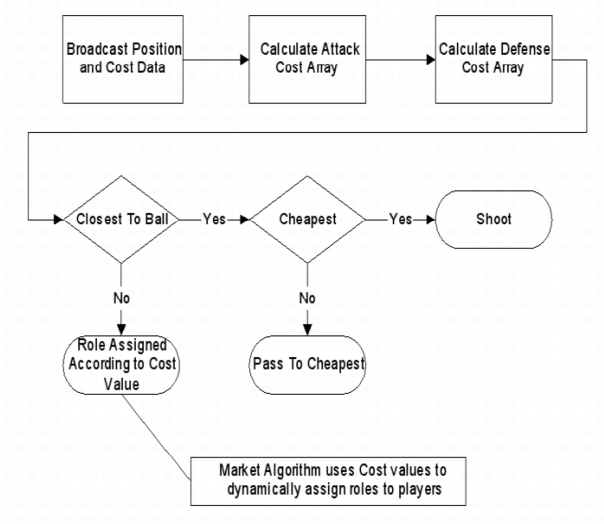

Fig. 5. Flowchart for task assignment

Although PFP approach provides a fast reactive controller for a single agent, soccer is a cooperative game and the robots should cooperate and collaborate with the teammates. For this purpose, we employ a market driven task allocation scheme [10], [8], [9]. In this method, the robots calculate a cost value (their fitness) for each role. The calculated costs are broadcasted through the team and based on a ranking scheme, robots chose most appropriate role for their costs. Here, each team member calculates costs for its assigned tasks, including the cost of moving, aligning itself suitably for the task, and cost of object avoidance, then looks for another team member who can do this task for less cost by opening an auction on that task. If one or more of the robots can do this task with a lower cost, they are assigned to that task, so both the robots and the team increase their profit. Other robots take actions according to their cost functions (each takes the action which is most profitable for itself). Since all robots share their costs, they know which task is appropriate for each one so they do not need to tell others about their decisions and they do not need a leader to assign tasks. If one fails, another would take the task and go on working.

The approach is shown in the flowchart given in Fig. 5. The robot with the smallest score cost C_{ES} will be the primary attacker. Similarly the robot, except the primary attacker, with the smallest $C_{defender}$ cost will be the defender. If $C_{auctioneer}$ is higher than all passing costs ($C_{bidder(i)}$) then the attacker will

shoot, else, it will pass the ball to the robot with the lowest $C_{bidder(i)}$ value. The cost functions used in the implementations are as follows:

$$C_{ES} = \mu_1.t_{dist} + \mu_2.t_{align} + \mu_3.clear_{goal} \quad (1)$$

$$C_{bidder(i)} = \mu_1.t_{dist} + \mu_2.t_{align} + \mu_3.clear_{teammate(i)} + C_{ES(i)}, i \neq robotid \quad (2)$$

$$C_{auctioneer} = C_{ES(robotid)} \quad (3)$$

$$C_{defender} = \mu_5.t_{dist} + \mu_6.t_{align} + \mu_7.clear_{defense} \quad (4)$$

where *robotid* is the id of the robot, t_{dist} is the time required to move for specified distance, t_{align} is the time required to align for specified amount, μ_i are the weights of several parameters to emphasize their relative importance in the total cost function, $clear_{goal}$ is the clearance from the robot to goal area-for object avoidance, $clear_{defense}$ is the clearance from the robot to the middle point on the line between the middle point of own goal and the ball-for object avoidance, and similarly $clear_{teammate(i)}$ is the clearance from the robot to the position of a teammate. Each robot should know its teammates score and defense costs. In our study each agent broadcasts its score and defense costs. Since the auctioneer knows the positions of its teammates, it can calculate the $C_{bidder(id=robotid)}$ value for its teammates.

The game strategy can easily be changed by changing the cost functions in order to define the relative importance of defensive behavior over offensive behavior, and this yields greater flexibility in planning, which is not generally possible.

Locomotion A new motion module similar to PWalk of UNSW [11] has been developed for Cerberus'05. This new motion module has a fully object-oriented structure in which each part of the robot is described as a separate object. First of all, the robot is considered as a single object composed of many other objects. Specifically, an AIBO robot physically consists of four legs, a head, and a tail, each of which carries different number of joints. Each Leg has three joints, which are the rotator, the abductor, and the knee joints. The Head has four joints, which are pan, tilt, roll and mouth joints. Finally, the Tail has two joints, which are pan and tilt joints. All these objects are defined as a separate class. The classes used for the locomotion module is shown in Fig.6.

Besides its object-oriented structure, this new motion module includes different walking styles, such as the combination of low stance and high stance, and locus based parametric kicks. It also provides smoothness in specific motions, such as head-search, since all these motions are declared to be vision-independent.

Some special static actions, such as blocking the ball, contain "don't care" fields in their joint position files. This provides independence of joints from each other; that is, a goalie can continue tracking the ball while trying to block it.

All the necessary parameters are optimized by using evolutionary algorithms in order to obtain smooth walking and kicking motions.

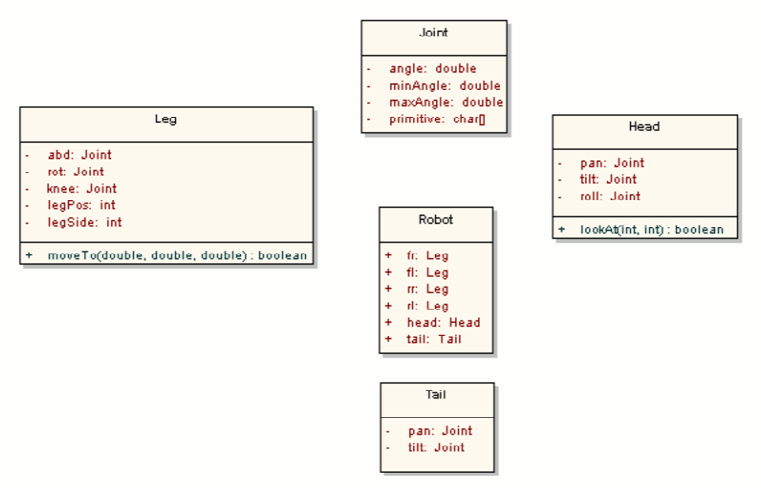

Fig. 6. Classes used in the motion module

References

1. YILDIZ, O. T, L. Akarun and H. L. Akın, "Fast nearest neighbour testing algorithm for small feature sizes", *Electronics Letters*,Vol 40, No 3, pp. 171-172, February 2004.
2. Schioler, H. and Hartmann, U., "Mapping Neural Network Derived from the Parzen Window Estimator", *Neural Networks*, 5, pp. 903-909, 1992.
3. KOSE, H and H. L. Akın,"A fuzzy touch to R-MCL localization algorithm", Robocup 2005 Symposium. (accepted)
4. KOSE, H and H. L. Akın, "Experimental Analysis And Comparison Of Reverse-Monte Carlo Self-Localization Method", CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby, December 2 – 4, 2004, Vienna, Austria.
5. KOSE, H and H. L. Akın, "Robots From Nowhere," *RoboCup 2004: Robot Soccer World Cup VIII*, LNCS 3276, pp.594-601, 2005.
6. KAPLAN, K. 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 Symposium. (accepted)
7. KAPLAN, K. and H. L. Akın, "A Controller Design for Soccer Robot Teams", *IJCI Proceedings of International XII Turkish Symposium on Artificial Intelligence and Neural Networks TAINN 2003*, 1, 1, July 2003.
8. KOSE, H., Ç. Meriçli, K. Kaplan and H. L. Akın, "All Bids for One and One Does for All: Market-Driven Multi-Agent Collaboration in Robot Soccer Domain", *Computer and Information Sciences-ISCIS 2003, 18th International Symposium Proceedings*, LNCS 2869, pp. 529-536, 2003.
9. KOSE, H., K. Kaplan, C. Mericli and H. L. Akın, "Genetic Algorithms Based Market-Driven Multi-Agent Collaboration in the Robot-Soccer Domain", FIRA Robot World Congress 2003, October 1 - 3, 2003, Vienna, Austria.
10. KOSE, H, U. Tatlidede, C. Mericli, K. Kaplan and H. L. Akın, "Q-Learning based Market-Driven Multi-Agent Collaboration in Robot Soccer," Proceedings, TAINN

2004, Turkish Symposium On Artificial Intelligence and Neural Networks, June 10-11, 2004, Izmir, Turkey, pp.219-228.
11. UNSW 2003 team report "http://www.cse.unsw.edu.au/ robocup/report2003.pdf"