Team RoBIU

Team Description Paper for RoboCup 2012 Humanoid KidSize League

Gilad Arnon, Naama Avrahami, Amir Ben Ami,
Guy Cohen, Sarit Dauber, Idan Farin, Amir Kliner, Alon Ludmer,
Yoav Miller, Efrat Taitelbaum, Israel Zilbershmidet
Rafi Amsalem and Dr. Eli Kolberg

Bar-Ilan University, Faculty of Engineering
52900 Ramat-Gan, Israel
E-mail: RoBIU.RoboCup@gmail.com
Web: http://www.eng.biu.ac.il/~millery/robocup

Abstract. This paper presents the hardware and software layers of the kidsize humanoid robots of RoBIU team. Team RoBIU was founded in 2010, consists of only undergraduate students from Bar-Ilan University Faculty of Engineering. The paper describes the robot's specifications and the main aspects of our project, including real-time image processing and object detection, sensors and camera based localization, high-level behaviors implementation and robot agents inter-communication.

1 Introduction

RoboCup 2012 signifies a milestone for team RoBIU. After in the last year the team[1] (formerly known as BI-Forward) couldn't participate in RoboCup 2011 due to security constraints regarding travel to Turkey, team RoBIU hopes to be the first to accomplish that. This year an improved system was assembled, mainly by the use of a more powerful robot, in the aspects of robustness, processor performance and better sensors. RoboCup 2012 will be a great setting to study our enhanced software and team performance.
The RoBIU team is made up entirely by undergraduate senior students, as a part of their final year project, under the academic supervision of Mr. Rafi Amsalem and Dr. Eli Kolberg. The purpose is to let the students experience with a large-scale project, which incorporates challenges, such as strategics orientation, coping with deadlines, mediating between target groups and managing the development of software-intensive systems.

2 Robot Overview

2.1 Mechanical Design

The robot we use is the DARwIn-OP [2]. Fig. 1 shows the dimensions of the robot. The motion mechanism consists of 20 degrees of freedom, divided as 6 DoF for each leg, 3 DoF for each arm and 2 DoF for the head. The robot's weight is 2.8 kg and its height is 45.5 cm. The robot's walking speed is optimized for real-time adjustments maintaining fast and stable locomotion.

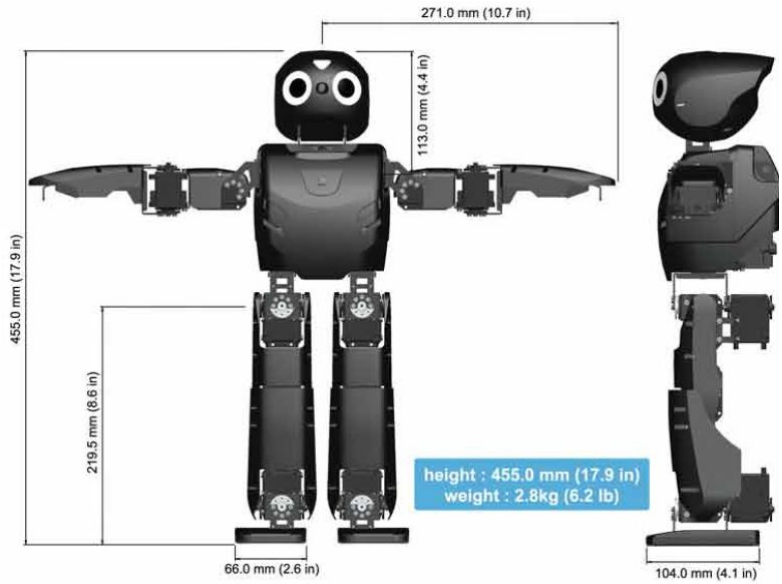


Fig. 1. DARwIn-OP's dimensions and weight.

2.2 Actuators

The robot uses Robotis' RX-28M Dynamixel motors[3]. Each servo motor has its own microcontroller, allowing for a network of servos to be controlled by a single microprocessor via a TTL network. The RX-28M weights 72g, has a resolution of 0.29 degrees and a gear reduction ratio of 193:1. The RX-28M provides the feedback of temperature, position, load and input voltage, indicating an an overheat and emergency shutdown when the temperature exceeds its range.

2.3 Sensors

The robot is equipped with a 3-axis LYPR540AH gyroscope and a 3-axis ADXL335 accelerometer that enable a robust and stable locomotion. The robot has a 2MP HD Logitech C905 Camera, with a 640x480 resolution, providing a visual information that is used for object detection and localization. The robot also uses pressuremeters - 4 FSRs in each foot.

2.4 Controller

The main controller of the robot is a CompuLab FitPC2i board featuring a 1.6 GHz Intel Atom Z530 processor with 1GB of RAM. The FitPC2i has a WiFi enabled for team communication.
As mentioned before, the robot also has a microcontroller board - the CM-730. The CM-730 is a management controller, with an ARM CortexM3 processor. The CM-730 connects between the servos and the FitPC. The CM-730 is connected to the FitPC via a USB port. The camera is connected to the FitPC via a USB port as well.

3 Software

The robot is a complex system which consists of several I/O components (like the camera and the sub-controller). As in a real-time system, all those components must be controlled and accessed simultaneously. The need for parallel processing is obvious, and can be achieved by multi-processing or multi-threading. We chose the multithreaded environment over multiprocessing from following reasons:

- Inter-thread communication is faster.
- All threads within a process share the same memory space.

It should be mentioned that our CPU has only one core which doesn't allow us to actually run different code sections in parallel.
The challenges of working in such environment are synchronizing between the threads and avoiding different threads from being in a critical section at the same time. In order to keep the threads synced we use unix signals. For example, when the robot falls – a thread that constantly checks its status signals the AI thread to handle the falling. Mutual exclusion is obtained using semaphores, which allow only one thread to enter the CS at the same time.

3.1 Vision

We use the HSV image format[4]. Because the R, G, and B components of an object's color in a digital image are all correlated with the amount of light hitting the object, and therefore with each other, image descriptions in terms of those components make object discrimination difficult. Instead, descriptions in terms of hue/saturation/value are far more relevant. The implementation uses some functions from the OpenCV library[5].

We start our image processing with an image segmentation. This is done in a very simple way, using the thresholding method. The second part of the vision section is object detection. Various algorithms have been tested in the ball detection part, ultimately we've chosen the most efficient one in the aspects of fast computations and accuracy. We sort the orange bulbs in the image and choose the best matching circle-shaped bulb. The bulbs are being characterized with several attributes, allowing us to distinguish between

the ball and other orange bulbs. The result is shown in figure 2.

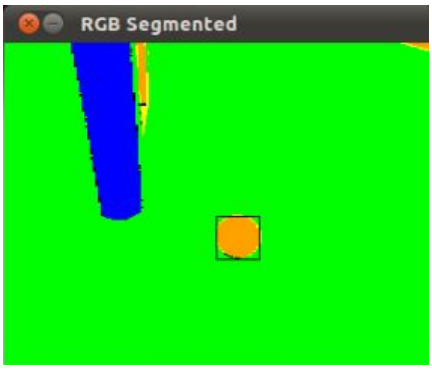


Fig. 2. Ball Detection.

The object detection also includes poles and goal detection. These are done with rather rough algorithms, allowing us to detect the goal and the posts very rapidly at the expense of small inaccuracies. The Goal Detection can be seen in figure 3.

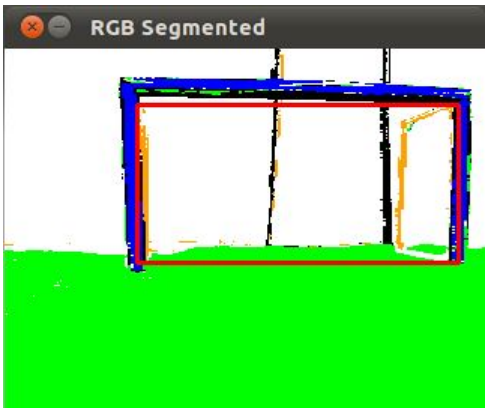


Fig. 3. Goal Detection.

All the information retrieved from the object detection is sent to the localization, in order to the robot to localize itself in the field. Once an object is detected, the distance from it is estimated according to the amount of pixels in it. The angle from each object is also calculated, by the formula:

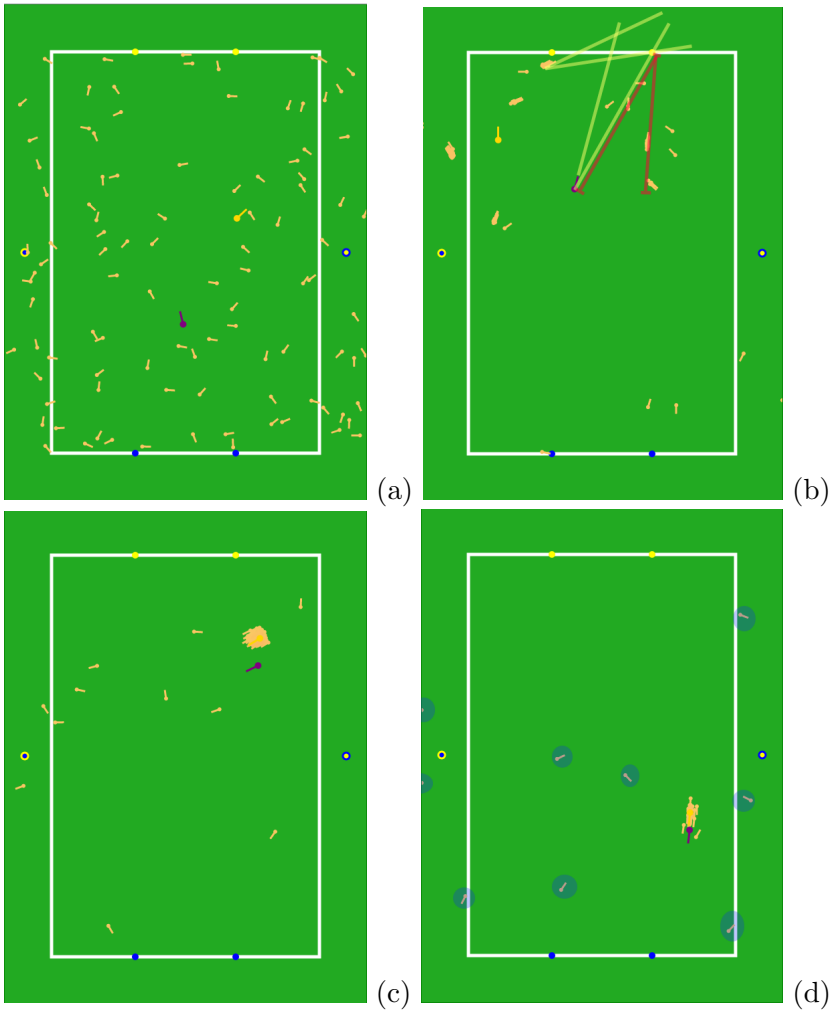
$$Angle_{deg} = (Object\,Offset\,from\,Center)_{pixels} * \frac{Horizontal\ Field\ of\ View_{deg}}{Horizontal\ Pixels_{pixels}} + Head\,Pan\,Angle_{deg}$$

where the fraction in the above formula is a constant, which depends on the camera properties.

3.2 Localization

The localization is done using Particle Filtering, a.k.a. Sequential Monte Carlo method[6]. The inputs we use are objects detected by the camera, step tracking and one gyro. By using the filter, we can make an educated guess of the robot's location and direction, relative to the fixed objects in the field (goals, side poles, etc.).

The algorithm goes as follows: M particles represent the robot's position estimation. The initial global uncertainty is achieved by randomly generating such M particles (fig. 4-a). Given an image, particles are graded by the angle the pole is seen, and by the distance. Noise is added to each input (fig. 4-b). After a few iterations, the particles are converging to the robot's real location (fig. 4-c). After some more iterations, the particles are very close to the robot's real location. Note that some particles (10%) randomly scattered are added each iteration to the particle sets (fig. 4-d) in order to be able to handle failures such as robot kidnapping or global localization failures.



(a) (b) (c) (d)

Fig.4. Monte Carlo localization with random particles. Each picture shows a particle set representing the robot's position estimate (small lines indicate the orientation of the particles). The yellow particle depicts the mean of the particles, and the true robot position is indicated by the purple particle. The pictures illustrate the robot's global localization in the robocup field.

3.3 Locomotion

In the locomotion section, our biggest concern was the robot stability while moving. We've managed to enhance the robot's mobility in both aspects: speed and stability. The way to achieve good results requires many tests which follow every change in the basic movement configuration. By making the robot to lean forward and bend a little, we increased its maximal speed without falling. This way we got to a maximal speed of approximately 40cm per second. Additionally, in the actions field (kicking, coming up and passing) major improvements were reached. The robustness of the actions is now higher, while the time to complete each action has been decreased.

3.4 Communication

According RoboCup Soccer Humanoid League rules, the robots may communicate only via the wireless network provided by the organizers which must support the referee box. Though sending any transmission from an external computer to the robots is prohibited (except from the game controller), the robots may communicate with each other at any time during a game.

Sharing the information about the robots' location and ball location can be significant to achieving efficient robotics team. This can be useful while in-game, for example when a certain robot doesn't see the ball, he can use other teammate's information as a guide. Disseminating information among the robots includes saving a database of knowledge of the robot itself, and maintaining databases of other robots' knowledge. Decision making system takes in account (along many other elements) the information from the other robots, relying on it according to how accurate and up-to-date the information is.

References

[1] Team BI-Forward, Team Description Paper, RoboCup 2011, Humanoid League.

[2] DARwIn-OP Brochure, http://darwin-op.springnote.com .

[3] Robotis Product Information, http://www.robotis.com .

[4] R. Gonzalez and R. Woods, *Digital Image Processing*, Third Edition, Pearson Education, 2008.

[5] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, October, 2008.

[6] S. Thrun, W. Bugard and D. Fox, *Probabilistic Robotics.* MIT Press, 2005.