Team RoBIU

Team Description for Humanoid KidSize League of RoboCup 2014

Bartal Moshe, Chaimovich Yogev, Dar Nati, Druker Itai,
Farbstein Yair, Levi Roi, Kabariti Shani, Kalily Elran,
Mayaan Tal, Negrin Lior, Poss Menachem, Zuckerman Michael
Rafi Amsalem, Benjamin Abramov, and
Dr. Eli Kolberg

Bar-Ilan University, Faculty of Engineering
52900 Ramat-Gan, Israel
E-mail: robocup-biu-2014@googlegroups.com
Web: http://www.eng.biu.ac.il/ kalilye/robiu2014.html

Abstract. Team RoBIU was founded in 2010, the team consists of undergraduate students from Bar-Ilan University Faculty of Engineering.This paper presents an overview description of the hardware and software layer of the kidsize humanoid robots of RoBIU team. The paper describes the robot's hardware specifications and a high level description of the various software algorithms, including real-time image processing, stabilization, sensors and camera based localization, debug features, robot agents inter-communication and high-level behaviors implementation.

1 Introduction

This paper describes the Robocup Kid Size League team RoBIU from Bar Ilan university. The team was founded in 2010 and this is the 3rd year in a row the team is participating the KSL league. Each year the team is assembled with new undergraduate senior computer engineering students, as a part of their final year project under the supervision of Mr. Rafi Amsalem, Mr. Beni Abramov and Dr. Eli Kolberg. Part of the purpose of the Robocup KSL project is to let the students experience with a large-scale projects, which incorporates many challenges, such as strategics orientation, coping with deadlines, mediating between target groups and managing the development of software-intensive systems.

RoboCup 2014 would be a great setting to study our enhanced software and team performance.

2 Commitment

The RoBIU team is committing to participate in the Robocup 2014 Kid Size League Humanoid competition taking place in Joao Pessoa, Brazil. The RoBIU team is committing to provide a person with sufficient knowledge of the rules available as a referee during the competition.

3 Robot Overview

3.1 Mechanical Design

We chose to use Dynamic Anthropomorphic Robot with Intelligence-Open Platform (DARwIn-OP) [2].

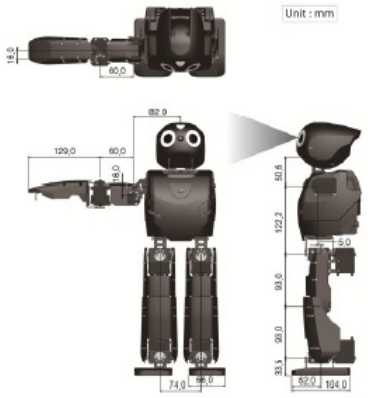


Fig. 1. DARwIn-OP's dimensions.

Figure 1 illustrates the mechanical design of DARwIn-OP. This robot has a total of 20 degrees of freedom: 6 DoF for each leg (X2 for both legs), 3 DoF for each arm (X2 for both hands) and 2 DoF for the head. The center of mass is located in the middle of its upper body. This location is ideal for proper balancing. The robot's height is 45.5 cm and it weights 2.8Kg. The hollowness of its frames results in a low weight. The robot's walking speed is determined according to locomotion considerations.

3.2 Actuators

DARwIn-OP uses Robotis RX-28M Dynamixel motors. It is possible to control all the servos by a single microprocessor via a TTL network since each motor has its own microcontroller. The RX-28M features a conventional potentiometer for position control. The RX-28 motor weights 72g and has a resolution of 0.29°and a gear reduction ratio of 193:1. Dimensions: 35.6mm x 50.6mm x 35.5mm

3.3 Sensors

DARwIn-OP has several kind of sensors as illustrated in figure 2. The basic sensors are 3-axis gyroscope and a 3- axis accelerometer for posture estimation and balancing. A camera (2MP HD Logitech C905, resolution 640x480) and 3 microphones are located in the head. 4 FSRs (Force Sensing Registers) are located in each foot.

Fig. 2. DARwIn-OP's components.

3.4 Controller

DARwIn-OP uses a CompuLab FitPC2i board featuring a 1.6GHz Intel Atom Z530 processor with 1GB of RAM. The FitPC2i has a WiFi enabled for team communication. The robot has CM-730 microcontroller. 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.

4 Software

Our robots are assembled of several components that combine a real time soccer player robot. In order to do so, we need to design a software that unites all the necessary functionalities into one multithreaded program. Multithreaded environment let us delegate the robot's authorities (such as brain, vision, localization etc.) in order to be more efficient. By using multithreaded environment we enjoy using same memory space for all threads and inter-thread communication is faster.

4.1 Artificial Intelligence

Artificial intelligence AKA Brain - is the main process of the robot. It's in charge of taking all important inputs from other modules, understanding and creating the next move. We started by writing a simple Brain which knows how to find the ball and kick to the opponent's goal. Afterwards we slowly considered more

and more factors such as Localization properties, communication between robots for the robot to be more sophisticated.

4.2 Vision

We use the HSV image format[4]. Because the RGB 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 are scanning for orange objects over green field, then we choose the biggest circle shape.we are cacluating the size of the object in the center of mass in order to kick the ball in the ideal way.

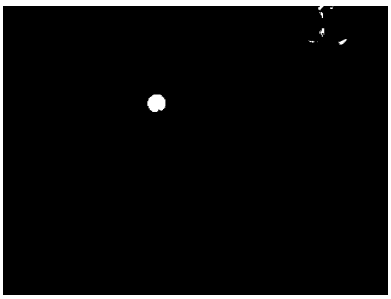


Fig. 3. Ball Detection.

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 rapidly at the expense of small inaccuracies. In addition we have white lines detection, in which we use to map the field in the localization section. the white lines are only detected if thet are over green field as can be seen in figure 4.

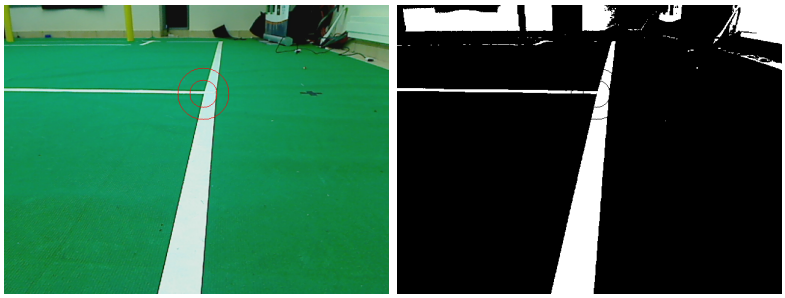


Fig. 4. Line Detection (Before and After).

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:

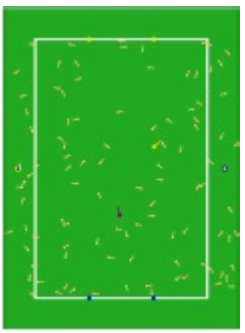
$$\alpha_{deg} = (Object\ Offset)_{pxl} * \frac{Horizontal\ Angle\ of\ View_{deg}}{Frame\ Width_{pxl}} + Head\ Pan_{deg} \quad (1)$$

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

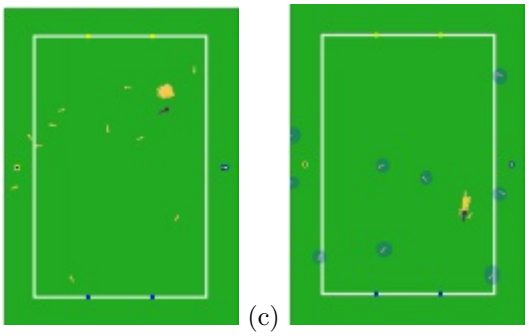
We use Bresenham's circle algorithm in order to classify white corners such as T corner, + corner and L corner, using circle mask around the suspicious corner located by fast algorithm.

4.3 Localization

Localization is the logical approximation of the robots location and angle in space based on previous knowledge of the field (eg. the location of goal posts, white lines and exact field dimensions) and knowledge of the dynamic entities on the field (eg. goalie, other robot players and robots own previous location) and inputs from the robots built in sensors (eg. camera, foot pressure sensors, gyroscopes and accelerometers.). For this task we implemented a particle filter using the Sequential Monte Carlo Method[6] which works iteratively in the following manner: Start by randomly distributing M logical particles over the field (fig. 5-a). We then give each particle a weight based on the probability that it is the robots location by using the inputs and previous knowledge as described above. The particles are then redistributed over the field with positive bias given to areas that had high probability based on the last iteration, 10 percent of particles are randomly distributed to compensate for cases such as kidnaped robot and other localization failures (fig. 5-d) and noise is added to each input (fig. 5-b). Given reliable inputs, after a number of iterations the particles converge to the actual robot location(fig. 5-c).


(a)

(b)



(c) (d)

Fig.5. 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.

4.4 Motion

Motion contains walking behaviors and actions. Dynamixel MX-28 is the robot's servo. It lets us use high resolution (4096) engine state. We wrapped the built-in action and walking ability with smart software which controls the engines. We made sure only one entity uses the engines at a certain time. We built functions which increase the robot's walking ability such as speed control (slow takeoff, speed increasing while walking and slow landing) and crescent walking.

4.5 Stability

In this section, our biggest concern was maintaining the robots stability while walking. We've managed to enhance the robot's stability using inverse and forward kinematics calculation. The forward kinematics calculation helped in tracking the legs positions and add control algorithms to handle variations in results. The forward kinematics also aided us to get better tracking of the robot's movment in the field and supply better localization inputs. The inverse kinematics calculations helped us determaining various servo's angles according to the final leg position desired. During our work we've also experimented with various walking gaits in order to provide the best solution possible maintaining stabilization while increasing the walking speed.

4.6 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.

Communication between the robot's is crucial in order to achive an efficient robotics team. Using communication methods the robot's can share information regarding their location and the ball's location. There are many aspects in which communication can help in developing better in-game algorithms and help robots to co-work with each other in order to reach their goal. Based on past experiance where the WI-FI network was only partially availble during the competition we've decided to use UDP protocols to transfer messages between the robots, and defined a new message format in which the robots should use in order to communicate.

4.7 Debug and Game Control

Based-on the UDP communication between the robots, we've created a real-time debugging console which listens to the robots reports and displays them on a special UI. The debugging process will enable us to follow and understand the status and localization valuation of each robot, and to examine the decision-making process of each robot.

Fig. 6. Debug Software GUI

5 Conclusion

In this paper we've introduced the mechanical structure and software design of our robot. Although it's the 3rd year Bar Ilan University is competing, for many of us this is the first time participating in such a large scale event. We look forward to participate in the RoboCup competition this year, and we hope to play as a worthy competitors.

References

[1] Team RoBiu, Team Description Paper, RoboCup 2012, 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.