Team RoBIU
Team Description Paper for Humanoid KidSize League of RoboCup 2017

Brauner Ireman
Mantzur Barak
Munk Elyasaf
Pinsky Gil

Academic Supervisors:
Dr. Kolberg Eli   Dr. Abramov Benjamin   Mr. Amsalem Rafi

Advisors:
Halak Liran   Haluba Ori

December 1, 2016

Contact Person: Gil Pinsky
E-mail: gilness1@gmail.com
https://github.com/RobocupBarIlan/RobocupBiu

Abstract. Team RoBIU was founded in 2010. The team consists of under- graduate students from Bar-Ilan University, Faculty of Engineering. This paper presents an overview description of the hardware and soft- ware layer of the kidsize humanoid robots of RoBIU team. The fol- lowing documents describe the robots 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 behaviours 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 6th year that the team is applying for participation in the KSL league. Each year the team is assembled with new undergraduate senior year computer engineering students, as a part of their final year project under the supervision of Dr. Eli Kolberg and the mentors Dr. Beni Abramov and Mr. Rafi Amsalem. One purpose of the Robocup KSL project is to let the students experi- ence 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-intensives systems. RoboCup 2017 would be a great setting to study our enhanced software and team performance.

1.1 Prior Performance In RoboCup Competitions

2014 - 2nd Round Robin.
2013 - 1st Round Robin.
2012 - Quarter finals.

1.2 Enhancements of the Robots Compared to the Previous Year

- In order to have access to more stable advanced functionality we will use the Robotic Operating System(ROS).
- We will introduce enhancements in our localization algorithm. In comparison to last year's alogrithm which was based only on geometry, we will use Monte Carlo localization.
- We will have a brand new line, T and cross detection algorithm.
- AI strategy will be improved. We will have coordination between the players.
- We will provide improvements in running time of the ball goal detection algorithms.

The software was developed by the current team members, based on last year program. The program includes new algorithms with better perfor- mance in terms of runtime, accuracy and robustness. In addition, the old algorithms have been adapted to the new robot.

2 Hardware

The robot's hardware - Motors, Sensors and Specifications - is presented in the Robot Specifications document.

3 Software

Our robots software is assembled from several components that combine a real time soccer player robot. In order to do so, we designed a software that unites all the necessary functionalities into one multi-threaded program.
Our software, from design to implementation, was developed by team RoBIU without any use of software from other teams.

Multi-Threading Multi-threaded environment let us delegate the robots authorities (such as brain, vision, localization, etc.) in order to be more efficient. We preferred to use multi-threading over multiprocessing so we could use the same memory space for all our software components, and in order to save the context-switch time. Our software consists of 4 main threads - Brain, Vision, Localization and Communication.

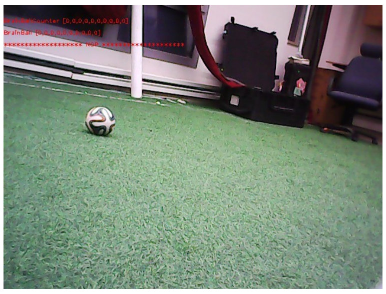
3.1 Artificial Intelligence

Artificial intelligence - AKA Brain - is the main component in our design. It is in charge of taking all important inputs from other modules, understanding and deciding the next move. The brain implementation is based on a FSM (finite state machine), which computes the next state according to the current state and the different updated inputs (vision, localization, etc.). We started by writing a simple FSM 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 coordination between robots for the robot to be more intelligent.

3.2 Vision

The Vision module is responsible for image processing. The main goal is to detect meaningful objects - ball, goal and white lines. The implementation uses some functions from the OpenCV image processing library[2].

Calibration Tool Our design contains a separate tool that adjust our image-processing to the current environment colors. The tool teaches the robot how to define the green color spectrum and the white spectrum. The tool shows the user 2 images - the original image and an only green/only white image. The user clicks on the green pixels in the original image and the tool colors only these pixels in white in the 'only green'/'only white' image, as can be seen in figure 1.


(a) Original Image

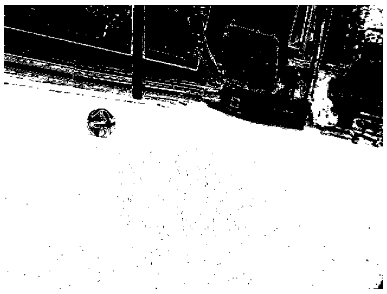

(b) 'Only Green' Image

Fig. 1: Calibration-Tool Example

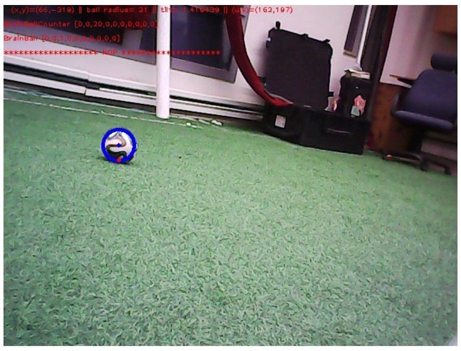
Common Image Processing Techniques
HSV - We prefer to use the HSV(hue,saturation value) image format[3] over the common RGB format. RGB components of an object's color in a digital image are all correlated with the amount of light hitting the object,and therefore with each other. Thus, image descriptions in terms of RGB components make object discrimination difficult. Instead, descriptions in terms of HSV are far more relevant.

Dynamic Threshold - Almost every image processing in our code starts with an 'image segmentation'. This is done in a very simple way, using the threshold method. The threshold uses the values that were calibrated with our calibration tool. By doing this, the threshold varies with every calibra- tion, making it adaptive to a variety of lighting conditions, grass color, etc.

Erosion Dilation After thresholding we use Erosion Dilation to 'fill holes' in the given BW image. Erosion is done by apllying a filter on the image that changes pixel color to white iff all the pixels that surrounds it are white. Dilation does the same action for black pixels. Combining these filters gives us the ability to 'fill holes' and more.
Various algorithms have been tested regarding ball, goal and white-lines detection. ultimately we've chosen the most efficient one in terms of fast computations and accuracy as described below.

3.3 Ball Detection

The ball detection algorithm is based on finding circle shapes in a BW image. Assuming light conditions are very noisy and therefore unexpected, as well as the colors of the ball are unknown, we have tried to develop an algorithm that is based on the geometry of the object and not on the objects color. We use OpenCV's Hough transform functions, in particular Hough- Circles, to detect circular objects. In addition, we ensure that the founded circular object is located on a green background, in order to eliminate circu- lar objects positioned outside the field. We also take into consideration the fact that the ball is 50% white. We color in white the other 50% non-white pixels (which are inside the ball boundaries) by applying ErosionDilation to the BW image. By doing it we get a perfect circle, which is easier to detect. A ball detection example is presented in figure 2



```
(x,y)=(66,-319) || ball radius= 21 || tilt= 1.416439 || (u,v)=(162,197)
BrainBallCounter [0,0,20,0,0,0,0,0,0,0]
BrainBall [0,0,1,0,0,0,0,0,0,0]
************************ NOP ************************
```

(a) Figure 2: The robot detects the ball

Fig. 2

3.4 Gate Detection

The goal detection is based on finding objects, which are suspected as the goals posts, in the input image. Then, the algorithm selects the most relevant ones to be posts. If only one object was found, the algorithm will determine which one of the two posts the robot sees. Object is suspected as a post if it satisfies the following terms:

1. **White:** First, we use a simple threshold function on the HSV transform of the given image (White is easy to recognize in the HSV transform image). We get a BW image, in which only white objects are white.

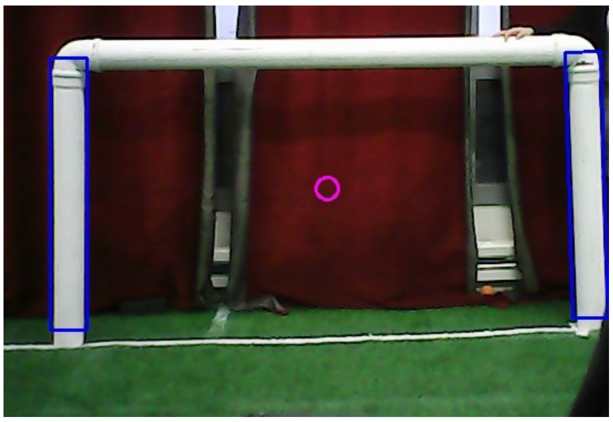
2. **Vertical:** We perform a vertical erosion algorithm on the given image to remove any horizontal white objects from the image. Only vertical white objects are left.

3. **Rectangle-shaped:** We use OpenCV's minAreaRect to surround all these objects with minimum area rectangles. We check the ratio be- tween the output rectangle and the white-object area, and we eliminate any rectangle that does not satisfy the threshold ratio.

4. **Straight-angled:** From the robots eyes, the posts are orthogonal to the fields plane. We check that the rectangles angle is close to zero.

5. **Inter-edge ratio:** The post's shape is characterized by long vertical edge and a short horizontal edge. We eliminate any rectangle that does not meet this characteristic.

Full goal detection: We take the 2 largest candidates to be the posts:



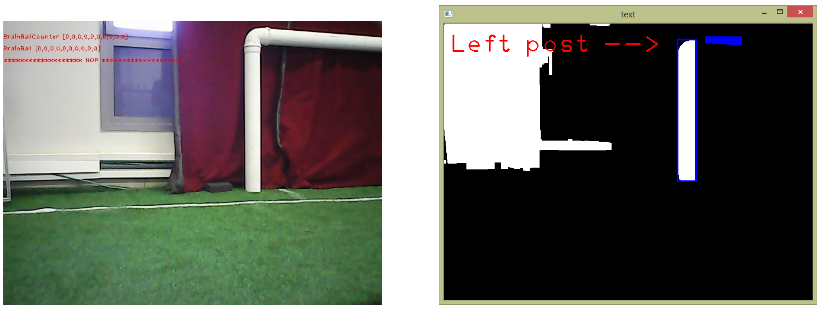
(a) The robot detects the goal and its center.

Fig. 3

Single post detection: If only 1 object was found we need to determine which one of the post the robot sees (left or right). We do it by detecting the crossbar. The algorithm detects the crossbar by performing horizontal erosion on the BW image, in order to remove any vertical white object. With that being done, the algorithm counts white pixels from the left and the right of the post's top. The direction with the larger number of white pixels determine the location of the post, as presented in figure 4.

4 Localization

Localization of the robot is one of the main features for its successful functionality. It means that the robot understands and decides where it is located and it consequences. The problems related to the localization algorithms include improper image data, symmetric playing field - e.g. in case the robot sees only a white line, he wont be able to decide which line in which side of the court it is, if local- ization is not used. In addition, localization is required for planning the robots next move. Ac- cording to its location, the robot can determine what



(a) The Original Image

(b) The crossbar is detected (colored in blue), and the algorithm detects that the post in the image is the left post.

Fig. 4: Single post detection

should be its next move; whether to go to ball, come back to help the defense or move to the center of the goal (in case it is a goalkeeper).

There are various and different ways to implement localization. Most of them are saving and using former data and current sampling in order to conclude the current localization. We decided to use ' *Particle Filter* ' in our project, which can help us to resolve the main following issues:

- **Position tracking** - In this scenario, we want the robot to find its location, as the initial location of the robot is known, as well as its control data since it started. The Particle Filter can solve this problem simply, by only changing the initial distribution.
- **Initial localization** - We want to be able to find the robot position in the field when there is no an initial position. We can use a uniform distribution as a starting point and rely on the Particle Filter to converge given enough parameters.
- **Kidnapped robot problem** - This problem is the hardest. In this scenario, the robot can be ' teleported 'at any time (e.g. the robot being moved by the referee) and the robot still need to be able to find its position after few iterations of the filter. To solve this scenario we use the Monte-Carlo Particle Filter. As an input for the localization, we mainly use vision along with the ob- stacles the robot identified so far. In addition, we take an advantage of the *Gyro*, which enables us to configure in which directions the robot turned and walked.

Parenthetically, we should take into consideration that the data on which the localization algorithm relies, namely Gyro and image processing, is not clean as it contains sample noise and other erroneous data. Consequently, the resolution of the robot's location will need to be a 'wise decision', relying on a high probability.

Another parameter we took into consideration is the type of environment. In a static environment, the one of which the Particle filter was developed for, the only changing variable is the robot's position. However, in our case, there are other moving objects (e.g. other robots). Despite of this fact, we can still consider our environment static since we can detect the field (a static variable) even though there are objects on it using image processing. Thus, we can still use the Particle Filter.

5 Conclusion

In our documents weve introduced the mechanical structure and software design of our robots. Although it will be the 5th year that Bar-Ilan University will participate in the competition, all the team members have changed (except of the mentors), so this will be our first chance to take part in RoboCup. We look forward to participate in the RoboCup competition this year, and are determined to play as worthy competitors.

References

1. obotis Product Information, http://www.robotis.com.

2. . Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, OReilly Media, October, 2008.

3. . Gonzalez and R. Woods, Digital Image Processing, Third Edition, Pearson Education, 2008.

4. . Thrun, W. Bugard and D. Fox, Probabilistic Robotics. MIT Press, 2005.