RoboTurk 2011 Team Description

Kadir Firat Uyanik[1], Mumin Yildirim[1], Salih Can Camdere[2], Meric Sariisik[2]

[1]Department of Electical Engineering
[2]Department of Mechanical Engineering
Middle East Technical University (METU)
06531 Ankara, Turkey
kadir@ceng.metu.edu.tr
{mumin.yildirim,salihcan.camdere,meric.sariisik}@ieee.metu.edu.tr

Abstract. This paper briefly explains the current development status of the recently reforming RoboCup Small-Size League(SSL) robot team, RoboTurk. RoboTurk SSL robot system is being designed under the RoboCup 2011 SSL rules so as to participate in RoboCup competition being held for the first time in Turkey.This year we have made crucial changes in the 2009's design, and in fact, all the hardware and software modules are being redesigned. Most of the effort have been spent on the mechanical structure of the robot, motor driver board, and ROS [1] based software architecture with the improved simulated robot in Webots[2].

1 Introduction

RoboTurk RoboCup SSL robot system is a project that's been studied by the members of IEEE METU Student Branch Robotics and Automation Society (IEEE METU RAS) since 2008. However, being undergraduate students as well as dynamically changing society members has slowed down the course of development considerably. After more than one year of development gap, this year we gathered a new team for the sake of hosting the RoboCup Competition in our home country, and re-designed most of the modules less than a few months.

We can divide the current system into two main parts, one is the sensorimotor unit of the system namely *ssl-robots* and the other is central decision making unit, called *ssl-game-planner*, with the the assumptions that the 2-D pose of the proponent and opponent robots and the position of the ball are provided with respect to the field reference frame as well as game state is already available specifying the allowed formations of the robots under the rules of *robocup-ssl*.

The major improvements we have done so far can shortly be listed as follows:

– Mechanical design; we have decreased the weight of the chasis from 1.9kg to 1.4kg by the factor of almost 25% with the new design based on the Skuba[5], and BSmart[6] teams' designs. Many improvements are done on the wheels themselves and the wheel assemlies.

– Motor control board design; we have replaced the earlier control circuit which happened to be unreliable from time to time, especially during sudden changes in the acceleration.
– Software design; we have started developing a *robocup-ssl* stack on ROS. In this design, we also started adding ROS support to the *ssl-vision* [3].

In the following section we state the details about the current development stage of the two main units of the current system, ssl-robots and ssl-game-planner, including new developments and the planned extensions until the RoboCup'11 competition.

2 SSL-Robots

As it's name implies SSL-robots unit consists of the ssl-robots which are then can be decomposed into mechanical and electrical sub-units. First, we describe the mechanical elements, then we describe the electrical parts needed to drive the mechanics.

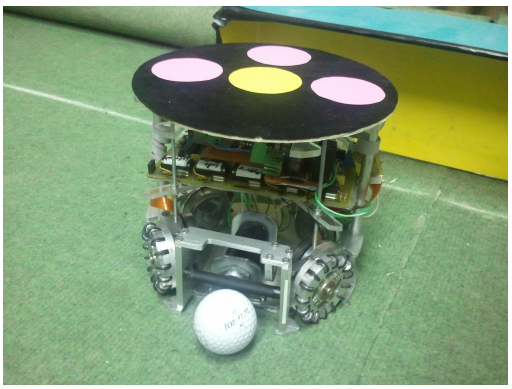

Fig. 1. RoboTurk 2011 robot

2.1 Mechanical Design

The mechanical system will consist of the case of the robot, motors, kicking mechanisms, dribbler, solenoid and wheels. The case will be limited to 18cm in diameter and 15cm in height. In mechanical point of view motors, solenoids, batteries, sensors and digital cards should occupy the smallest volume possible. Our earlier chassis design was more than 1900 grams; however, as a result of our latest improvements, we decreased our chassis to 1419 grams excluding circuit boards. In the following sections we will describe the omniwheels, motors, kicker, and dribbler parts.

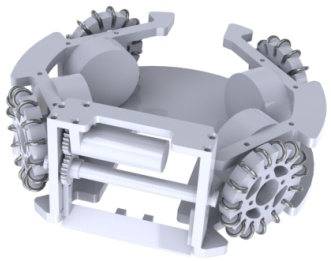

Fig. 2. RoboTurk 2011 robot mechanical design

Omni-wheels Omni-wheels allow the robot to move in any direction without having to rotate, or move on a line and rotate at the same time so that the robot can reach the destination at the correct angle. This makes path planning and following quite easy, and therefore it is widely used in RoboCup robots. In previous years we used Kornylak Omni-wheels. However, these omni-wheels were resulted in unstable motion. Therefore, this year we are using new omni-wheels that were designed and built by our team. The new wheels have a diameter of 1.968 inches and have 15 rollers with rubber gaskets in order to increase the road holding (Fig.1). Hence, the robots' motion abilities have enhanced. Omni-wheels are placed at 53 degrees with respect to the y-axis at the front and 45 degrees with respect to the y-axis at the back. This configuration is the result of the requirement that motors are uniformly loaded as well as there must be enough space to accommodate the dribbler on the front.

Motors Each omni-wheel is driven by a 30 Watt-Maxon EC45 Flat Brushless DC motor with hall sensors. These motors are small and compact; therefore they save space in the robot.

Dribbler In this year's design, we have also enhanced dribbler system. We are going to use Maxon EC-16 brushless DC Motor in order to actuate dribbler part of our robots since the dimensions of this motor fits the space constraints very well. Also, we are planning to use a system which consists of gears and cogwheels with the ratio of 2:1 instead of pulleys that are used in 2009's design.

2.2 Electrical Design

This section includes detailed explanations about our motherboard and microcontrollers, kicking and motor driver circuits and communication. Electronics hardware consist of three main parts; the main controller board, motor controller board and kicking control circuitry.

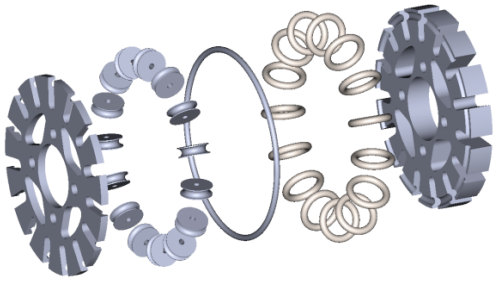

Fig. 3. Our latest omniwheel design

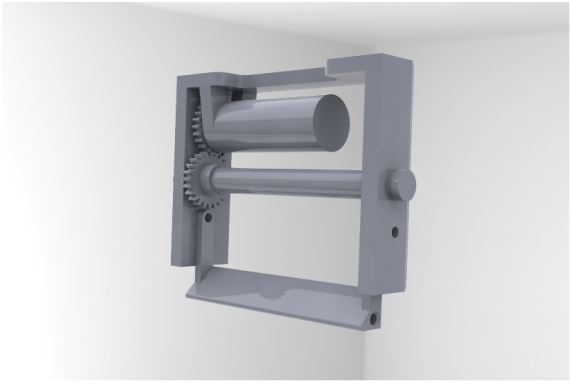

Fig. 4. Dribbler design

Main Controller Board In the first prototype of our 2011 SSL Robots main controller board consists of a PIC16F88 and Xbee Series 2 RF transmitter. The data packet sent from the central-control computer is received by Xbee by using UART protocole. An RDA interrupt runs in PIC and it distributes the package information to the motor controller board on I2C bus. The data packet sent on the I2C bus consists of 5 bytes. First byte defines direction and on/off options and rest four bytes includes speed data of each motor. Xbee is programmed to work on 57600 baud, with no parity and one stop bits. Furthermore, a personal area network is constructed for communication of each robot with the coordinator Xbee which is directly coordinated to the central-control computer. Low baud rate value is chosen due to the robustness of the system and PICs failure at higher baud rates. PIC 16F88 acts as a coordinator. It receives the data package on UART at 57600 baud rate. The data package is actually a string which contains intended speed values for motors and directions for dribbler and kicker. The coordinator PIC, also known as master PIC, uses a built-in SSP module (namely I2C) in order to communicate with each PIC in motor controller board on I2C bus. The infrared sensors positioned at dribbler part are directly connected to this PIC. Therefore, this coordinator PIC can also transmit the information about the ball possession.

Motor Controller Board Motor controller board consists of five independent motor driver modules. Each module contains a PIC 16F88 and a L6235N brushless motor driver IC in order to drive four Maxon EC-45 flat to drive omni-wheels and one Maxon EC-16 to drive the dribbler mechanism. L6235 has an internal logic decoder to drive the brushless motor from the hall sensor states. It also has internal MOSFEDS and MOSFED drivers. Thus, it is a clean and convenient solution to drive brushless motors.

Fig. 5. Motor control board

Kicker Controller Board Kicking control circuit contains a DC-DC booster for charging a capacitor, and a capacitor discharging circuit for main kicking solenoid. It supplies a feedback for voltage stablization at 200 Volts. The controller of this booster is another PIC16F88 whose duty is to supply pwm for booster toroid and maintain a stable capacitor voltage. The microcontroller is controlled by main controller board only for kicking purpose. A 200 V , 2000uF cappacitor is charged by DC-DC booster up tp 200 V in less than 4 seconds. A simple voltage division method and ADC conversion is used to determine the capacitor voltage. The capacitor is discharged on a tubular push type solenoid which has 3280 turns @ 31 awg. A power transistor is responsible for switching of the discharge. As a result the ball can be shot with the speed of 6.5 m/sec.

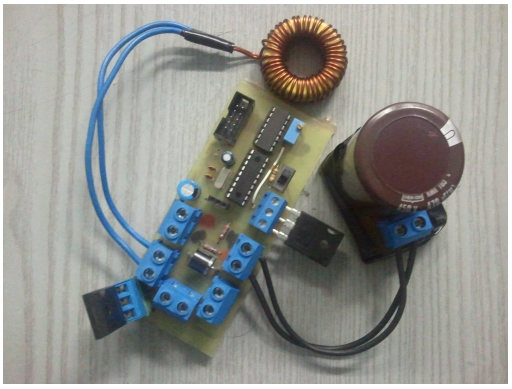

Fig. 6. Kicker control board

3 SSL-Game-Planner

SSL-game-planner unit can be investigated from two perspectives, software design and algorithmic content.

3.1 Software Design

This year, we have started to design our system by heavily using ROS graph concepts to establish communication between the processes which are the nodes in ROS network. Thanks to the message-based communication architecture, we can easily replace the real robots with the simulated robots and ssl-vision with the sim-vision (simulated vision) or the ssl-referee-box with the sim-refree (simulated refree) by making no change in the source code whatsoever.

Current architecture is shown in the figure 9. Most of the nodes are currently under development and some of them are stub at the moment (viz. ssl_refree,

ssl_sim_refree and most importantly ssl_game_planner).

This graph prabably is going to be more simplified during a real game as it is shown in the figure 7

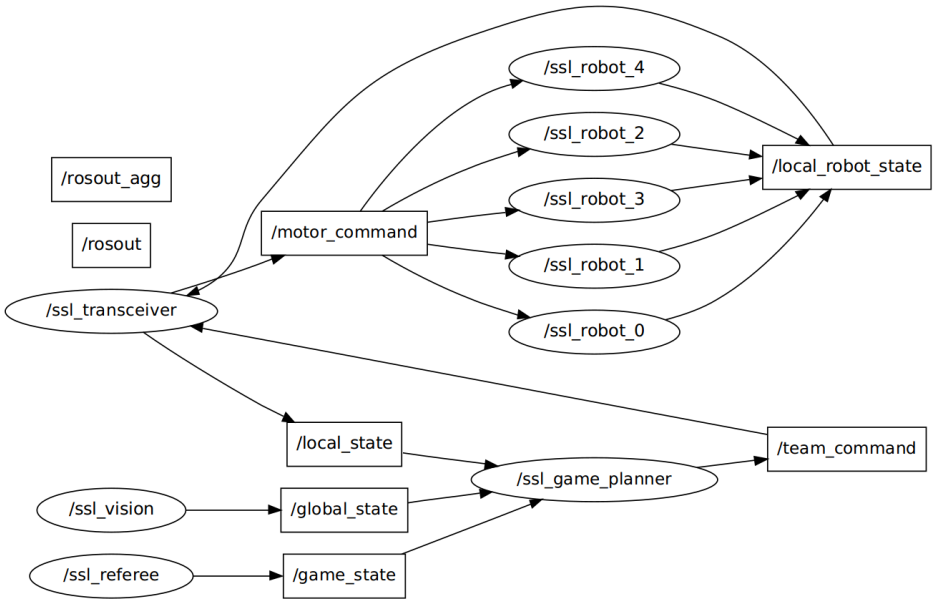

Fig. 7. This figure obtained via the ROS *rxgraph* command which shows currently running nodes in elliptical shapes and the topic messages in squared shapes. If an arrow is going out from a node, it means that particular node publishes data to the corresponding topic, and it is otherwise -subscribed to a topic- if an arrow in pointing to the node.

We are also working on adding ROS support for the ssl-vision mostly developed by Zickler et.al[3] and became the standard vision system for the RoboCUP SSL.

We are using Webots simulation environment for developing path planning algorithms and state estimation filters as they are explained in the next section. Since Webots is a commercial simulator and we are using trial version -soon going to be expired-, we are considering to move our simulated robot design to the Gazebo by using URDF and Xacro language, depending on our financial status. This alternative seems to be reasonable since ROS fully supports Gazebo simulator, and it is already included in several ROS distributions.

Fig. 8. ROS graph of the current system obtained by the ROS *rxgraph* command. Most of the nodes are currently stub-like. The nodes corresponding to the simulation environment are -most of the time- expected to be mutually excusive with the non-sim nodes (the nodes corresponding to the real environment). This figure best viewed in the soft-copy of the document which enables reader to zoom in the figure and clearly see the names of the nodes.

Fig. 9. Simulated robot in Webots simulation environment. It has exactly the same number of rollers in the omniwheels, and it can kick, chip-kick the ball as well as dribble although these behaviors are not developed in the current system.

3.2 Algorithmic Content

We reached to the stage where ssl-robots can realize the commands given by ssl-game-planner without colliding to the statical obstacles, following the path close to the optimal/shortest path. In order to accomplish this ability, ssl-robots should be able to project the velocity commands to each of the wheels that is where Motion Control module plays the role.

Obstacle avoidance is satisfied with the well-known artificial potential fields method. We have improved the method based on the object-grouping method proposed in [4], which is explained in detail in the following sections.

3.3 Motion Control

Current robot design has an assymmetrically distributed four-omni-wheel base. Wheel angles are, from the vertical line where robot points upward are 53° in the front and 45° at the back. Motor velocities can be obtained by using the following equation, where :

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} sin(\alpha) & -cos(\alpha) & -r \\ sin(\beta) & cos(\beta) & -r \\ -sin(\beta) & cos(\beta) & -r \\ -sin(\alpha) & -cos(\alpha) & -r \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

In simulation, motors are currently controlled with PID controller, yet it is not added to the actual robots since there are no encoders mounted to the motors, yet.

3.4 Path planning

In the traditional artificial potential fields, every obstacle is handled seperately, which prevents observing the overall picture of the environment. In this approach, visibility and proximity of the obstacles are also taken into consideration. For instance, the obtacles which are close enough to each other are considered as one obstacle(virtual) by grouping. Besides the obstacles which are not visible to the agent -because of the other obstacles between them- are not considered as obstacles which decreases the oscillations.

First, a linkability metric, say LINKDIST, should be defined which specifies when to link obstacles. According to LINKDIST obstacles are linked to each other starting from the nearest obstacle in the path. While linking, obstacles are checked if they are visible to the agent. If an obstacle is not visible, it is simply discarded. An obstacle is considered only once during linking process. That is, linked obstacles form a linked list data-structure.

After linking, linked obstacles are merged to form a single obstacle which has a proper radius so as to include all of the obstacles. The result of this process is virtual obstacles which are used to apply repulsive forces on the robot.

With this method, we get rid of the local minima and oscillations problems that are commonly encountered potential field based methods.

Algorithm 1 Obstacle Programming

```
while preObstacleList.size() > 0 do
  closestObstacle ⇐ getClosestObstacle()
  if isVisible(closestObstacle) = TRUE then
    tempObstacle ⇐ closestObstacle
    repeat
      if isLinkable(tempObstacle, preObstacleList[i]) = TRUE then
        linkObstacles(tempObstacle, preObstacleList[i])
        tempObstacle ⇐ tempObstacls− > Neighbor
      end if
    until preObstacleList.size() = 0
  end if
  virtualObstacle ⇐ closestObstacle
  groupObstacles(virtualObstacle)
  postList.push_back(virtualObstacle)
end while
```

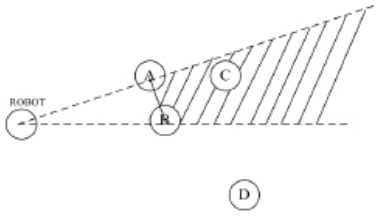

Fig. 10. Closest obstacle and its neighbor obstacle are linked to each other and nwe virtual obstacle makes obstacle C invisible to the robot (adapted from[4])

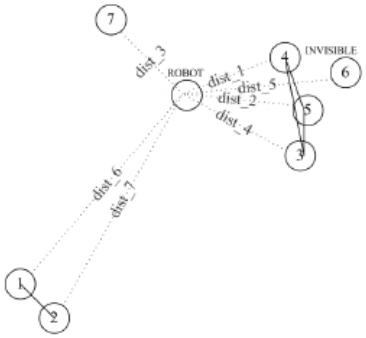

Fig. 11. Obstacle 3, 4 and 5 are linked to each other.They form a virtual obstacle (adapted from [4])

3.5 Game Planning

Our system is not able to perform higher level behaviors for the time being. We have successfully implemented ball tracking behavior for the real robots, but without encoders mounted on the wheels, robots are not able to respond to the precise rotational actions. Due to this situation, ball-related behaviors are not developed yet.

However, we are planning to stick to the commonly used hybrid hybrid deliberative/reactive control architecture and divide the planning problem into different stages such as *strategy*, *play* and *role*.

4 Conclusion and Feature Work

In this document, we have shown the current development stage of the RoboTurk SSL robot soccer system. We have emphasized the major changes in the mechanical design, motor control board, and software architecture. New robot design will enable the system to react quickly to the changes in the game state, as well as perform more efficient than the robot design we have developed in 2009.

We are planning to sutdy on the main focus of the RoboCup SSL domain which is game-planning throughout the whole semester. Robots, on the other hand, are going to be equipped with the Gumstix Overo-Fire single board computers which then runs GumROS, a gumstix software development library running ROS communication methods.

With the very young and motivated team members (Mumin, Salih and Meric second and first year undergraduate students, and Kadir being first year MS student) we are planning to attend RoboCup'11 Turkey for the first time if the chance is given.

Acknowledgments. Many thanks to the METU EEE department for providing us with a laboratory to work on a RoboCUP project, and to the IEEE METU Student Branch for their endless moral support. We also very grateful for their support in the constuction of the prototype robots, to Sariisik Makina.

References

1. Quigley M., Conley K., Gerkey B., Faust J., Foote T. B., Leibs J., Wheeler R., and Ng A. Y. (2009). Ros: an open-source robot operating system. n International Conference on Robotics and Automation, ser. Open-Source Software workshop.
2. Michel, O. Webots: Professional Mobile Robot Simulation, International Journal of Advanced Robotic Systems, Vol. 1, Num. 1, pages 39-42, 2004
3. S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso: SSL-Vision: The Shared Vision System for the RoboCup Small Size League. RoboCup 2009: Robot Soccer World Cup XIII. Pg:425–436

4. B. Zhang,W.Chen, M. Fei, An optimized method for path planning based on artificial potential field, Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06) Volume 3 (2006)
5. Wasuntapichaikul P, Srisabye J, Sukvichai K. Skuba 2010 Team Description. 2010.
6. Laue T, Fritsch S, Huhn K, et al. B-Smart Team Description for RoboCup 2010.