RoboJackets 2007: Small Size League

Andy Bardajagy, Stuart Donnan, Jason Kulpe, Phillip Marks, John McConville, Roman Shtylman, Ryan Stewart, and Scott Travis
alphabetical
andyb@gatech.edu, stuart@gatech.edu, gth722p@mail.gatech.edu, phillip.marks@gatech.edu, gtg346y@mail.gatech.edu shtylman@gatech.edu, gtg097y@mail.gatech.edu, scorpio@gatech.edu
http://www.robojackets.org

RoboJackets, Georgia Institute of Technology,
Atlanta Ga 30332, USA

Abstract. We are an undergraduate student organization at the Georgia Institute of Technology. Our goal is to develop a starting platform of knowledge and experience for further development in the comming years of RoboCup competition. The direction of the project is to lay the groundwork and have a basic system ready to compete for the 2007 Small Size League....

1 Introduction

The RoboJackets RoboCup team is composed of undergraduate members of the Georgia Tech RoboJackets robotics club. While this will be our first year of competition in the small-szie league this certainly isn't the first year the Georgia Tech has participated in a RoboCup event. In previous years there was another team in the four-legged league that was sponsored through the College of Computing. We hope to build on thier efforts for this years competition.

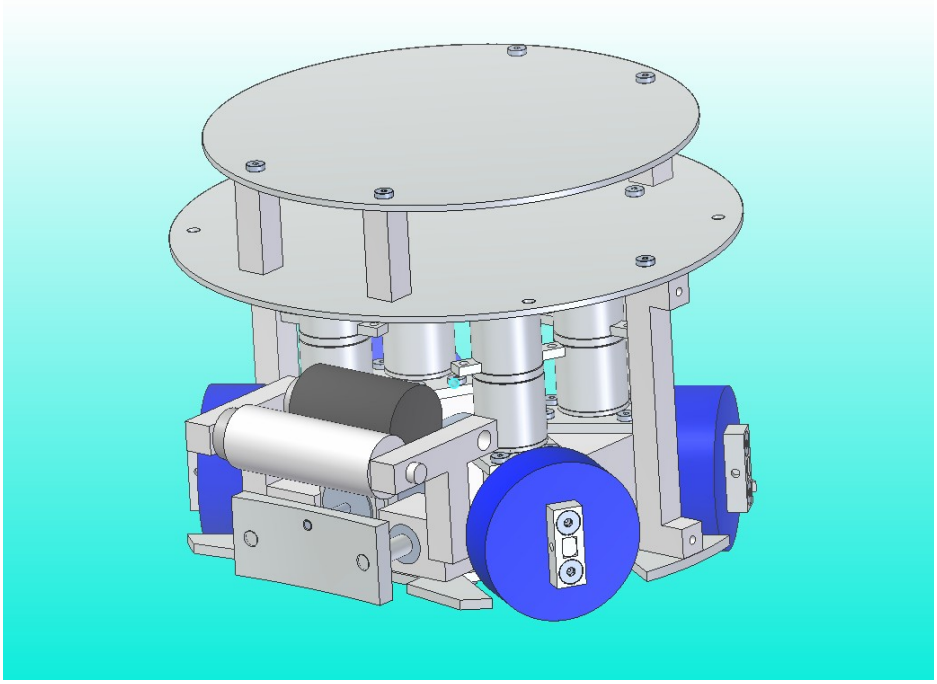
The team is comprised of three parts, the mechanical design team, the electronical and vision design team, and the software design team. The mechanical team designned the robot's base and shooting and dribbling mechanism; while the electrical and vision team desingned the on robot hardware and vision algorithms. The software design team interfaced the vision and electrical systems using a potential fields naviagation algorithm and simple set of AI code.

2 Mecanical

2.1 Introduction

The Mechanical team consists of four Mechanical Engineering undergraduate students. Our primary goal was to design and build a robust platform. The idea was to design and construct the robots quickly enough to leave the electrical and programming groups a sufficient amount of time to test and implement the software and more complicated algorithms.

2.2 Design Overview



Drive Train Our team will utilize 5 identical robots, each with four omni-directional wheels (2 in diameter) controlled by Maxxon DC motors for navigation, a solenoid to operate the kicking mechanism, and a smaller DC motor to control the dribbling subsystem. We designed the robots to have enough acceleration to traverse a third of the length of the field in one second starting from rest equating to a 3.2 m/s final velocity. We decided that a rapid acceleration was necessary in order for the robots to be able to sufficiently react to real time changes in the game. The DC motors were chosen to the specification listed above. Also, DC motors were chosen, as opposed to brush-less motors, because of the simple programming needed to control them. The selected motors contained ample power and a sufficient 14:1 ratio provided by the complementary gear box. As shown in Figure 2, the motors are installed vertically giving plenty of interior room for the kicker assembly and batteries. The group decided to place the motors in a skewed configuration resulting in a 60 degree angle between two motors on each side, relative to each other. This solution was chosen over an orthogonal wheel configuration to give more power for driving in a direction perpendicular to the kicking surface. This drive wheel angle sacrifices power for moving parallel to the kicking surface, but this sacrifice is acceptable; putting more focus on rotating then driving in the desired direction. The angled wheels, in addition to the vertical motors give ample room for ball control.

Ball Control The robot will control the motion of the ball through the use of a dribbler mechanism and a kicker plate. The horizontal dribbler is controlled by a DC motor mounted directly behind the dribbler. The kicker plate consists of an aluminum plate actuated by a solenoid. The most interesting part of this system is the geometry of the kicking plate. The surface of the plate that will make contact with the ball contains a parabolic curve to help ensure the direction of travel of the ball is consistent with numerous kicks.

3 Electrical

We've designed our electrical system this year to give us a strong system on which to base future designs. It consists of six subsystems: the wireless modules, the motor controllers, as set of sensors, a micro-controller, a CPLD, and a kicker solenoid drive circuit. A diagram of the system is shown in figure x.x.

The wireless system consists of an Linx HP3 series transmitter and receiver, both in the 900MHz band. A serial asynchronous protocol operating at 56kbps is implemented on these devices with each packet containing a 16-bit header followed by a variable length data payload and 16-bit checksum. The bulk of communication is from the host controller to the robots in the form of 154-bit muli-cast messages sent at 60 hertz. These messages contain the x and y velocity being commanded by the host along with an angular rate. Each robot sends back to the host information such as whether or not it has the ball at a much slower rate of 10 hertz.

The four main motors are driven by an H-bridge motor driver capable of delivering 5A at 46V while the dribbler is turned on or off by a MOSFET switch. A shaft encoder giving quadrature output pulses is used along the commanded velocity gyro are used to implement PID feedback control on the motors to adjust their output.

A set of infrared LED's operating at 32kHz with a 50% duty cycle are used in conjunction with a receiver to detect if the ball is near the robot. The LED's are placed so that their signal reflects off the ball and towards the receiver. An ADXrs-300 gyro is used to measure the robot's rate of angular rotation. This rate is feed into the PID loop to control angular rate.

A CPLD in conjunction with a Phillips LPC2138 ARM based micro-controller is used to coordinate the on-robot electronics. The LPC2138 implements the serial protocol, handles sensor interface, and runs the PID control loop. It is helped by the CPLD which interfaces with the encoders converting the quadrature signal and re-transmits it to the LPC2138 as a velocity over SPI.

For the qualification video a simple prototype of the electrical system was built using a second ARM instead of the CPLD and without the sensors and encoders. Using the knowledge gained from this prototype we hope to improve

on our design. With the base system we've built this year we hope to add such capabilities as on robot sensing.

4 Vision

4.1 Introduction

Often, the most important midlevel image processing task is to group pixels or groups of pixels into regions based upon some shared trait such as texture, color or luminance. Color is generally regarded as the most resilient and significant trait of an object and therefore the most often used segmentation metric. There are many approaches to this task including linear color space thresholding, nearest neighbor classification, probabilistic methods, and constant thresholding.

Linear thresholding works by partitioning the color space into strata with linear boundaries. The pixels are then classified according to which partition it lies in. This is convenient for learning algorithms such as neural networks or multivariate decision trees [2].

Another approach is to use nearest neighbor classification. First, several hundred tokens are precomputed which have a unique location in the color space and an associated classification with a particular segment. Then, the nearest K tokens are found and the pixel is characterized by finding the largest proportion of classifications of the neighbors [2].

Probabilistic methods such as expectation maximization use similar techniques as nearest neighbor classification where instead of tokens, probability distributions, which lie within the colorspace, are precomputed for each segment. Then, the distributions are used to place pixels into the segment which has the highest probability of containing it. Unfortunately, linear thresholding and nearest neighbor color segmentation require the examination of every pixel in the image for every color segment. This implies O(n*k) computation are required to segment an image with n pixels into k segments. With excess of 100 frames per second, and cameras in the megapixel range, color segmentation can easily overwhelm the most advanced modern processors [3]. Clearly another approach is necessitated.

4.2 Description of the Segmentation Algorithm

While segmentation is generally considered a monolithic vision task, several components are necessitated to increase the accuracy, robustness and speed of the segmentation. First, the image must be preprocessed to convert the colorspace to one which is less affected by illumination changes. Next, linear thresholding segmentation is used to characterize the pixels as one of several predefined groups.

4.3 Colorspace Transformation

The proposed approach involves thresholding in a three dimensional colorspace. There are several different models to represent color including HSI (Hue Saturation Intensity), YUV (luminance, blue chrominance, red chrominance) and RGB (Red Green Blue). Most cameras and imaging software use the RGB colorspace making it a natural choice for colorspace classification. However, as the lighting in the scene changes, the apparent color will shift. That is, a lighting change will cause a change in the point in the three dimensional colorspace. However, either HSI or YUV colorspaces can be chosen to restrict or eliminate the color shifts due to lighting changes [4]. That is, in both the HSI and YUV colorspaces, the chrominance and luminance components are separate dimensions. Therefore, the luminance dimension can be discarded to yield a two dimension colorspace that is luminance (and therefore lighting) independent. The YUV colorspace will be employed as it is a more common standard then HSI.

Since most cameras (and the simulation software more on that later) output data in RGB format, it must be transformed into YUV format where only the chrominance components will be kept. Mathematically, it is a one to one linear mapping where every point in three dimensional RGB space is mapped to two dimensional chrominance space (luminance Y is discarded). Where the transformations are given as

4.4 Thresholding

As previously mentioned, a linear thresholding approach will be taken as it should be the lightest, fastest and most efficient thresholding algorithm. In this approach, linear thresholds within our two dimensional color model will be precomputed. These thresholds are represented as rectangular regions within the UV colorspace. To ensure each pixel is classified uniquely, the threshold regions will not overlap. The pixel is classified according to which region in the colorspace it lays.

In traditional linear thresholding, the pixel is compared to a minimum and maximum value for each dimension of the colorspace. Even limiting the colorspace to two dimensions, a minimum of four comparisons per pixel per segment color would be required to segment the image by thresholding.

Rather then thresholding integer color values, a bitwise comparison approach will be employed as set forth by J. Bruce et. al.[5]. That is, the thresholding colors are boolean value decomposed and stored as an array of discretized levels. This implies a simple bitwise AND of the elements of each array indicated by the pixel color component values.

Take the following example. The threshold "orange" could be represented by the following discretized arrays.

U = [0 0 1];
V = [0 1 1];

To check if a pixel with color values (3,2) is a member of the "orange" segment, U(3) is ANDed with V(2). The result is true. That means the pixel is a member of the segment "orange". The true speed increase comes when multiple comparisons are made concurrently. If the threshold "blue" is represented by the following arrays

U = [0 1 0];
V = [0 0 1];

Then the "blue" and "orange" threshold arrays can be combined to form

U = [00 01 10];
V = [00 10 11];

Now to compute a pixel's membership in either the "blue" or "orange" segments, only a single AND operation is necessitated. If the pixel has color values (3,2), U(3) AND V(2) yields 10 meaning the pixel is a member of the "orange" segment and not the "blue" segment. As many segments can be computed as the length of the constituents of the arrays. Using this scheme, bitwise parallelism is exploited to reduce the number of comparisons to linearly threshold dramatically.

4.5 Clustering

As in the implemented paper [1], simple four connectedness grouping is used to group pixels into clusters. That is, pixels which are touching on either their north, south, east or west edges are considered to be in the same group. Clustering was not considered an integral part of the paper, nor my implementation as quicker methods have been developed.

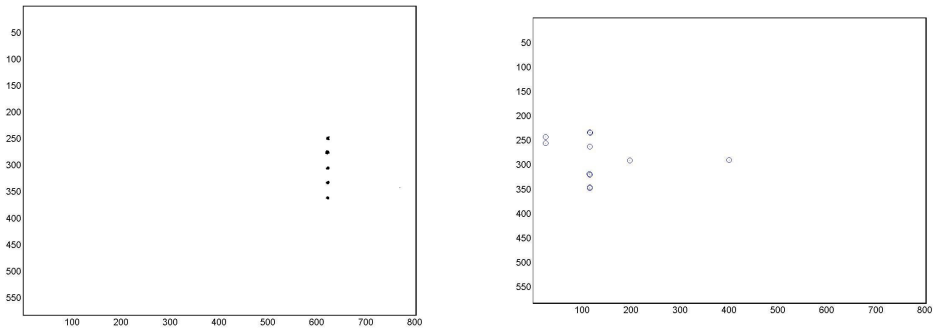
Once the clusters were computed, their centroids were found. Other statistics could be computed about each cluster, such as color variance, or average color. These features, while trivial, are unnecessary for our planning and navigation algorithms and therefore were not implemented.

4.6 Implementation

The algorithms were first implemented in MatLab (fairly straightforward). The MatLab prototypes were nice, and showed the algorithm worked, however their performance was nowhere near real time. Next, the thresholding algorithm was implemented on an Altera Cyclone EP1C6Q240 FPGA by adding single instruction, multiple data (SIMD) instructions to the NIOS II core. The development

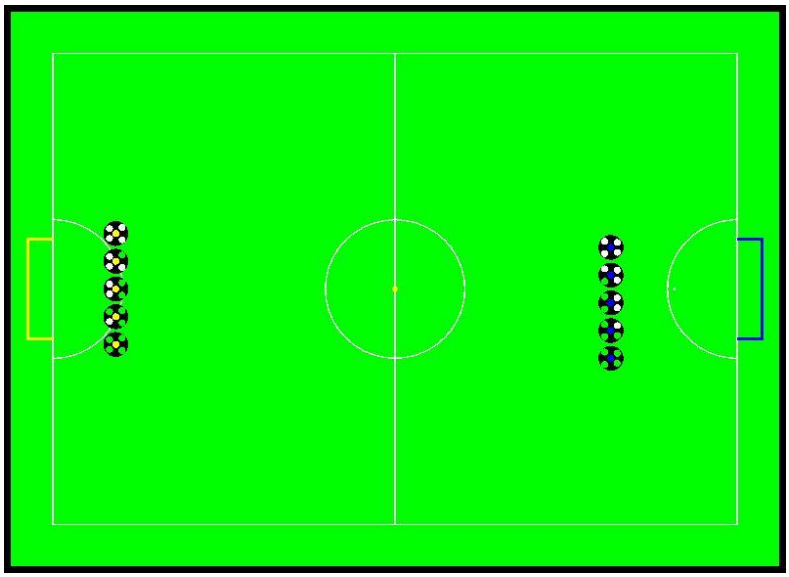
was done on a UP3 dev kit which has one megabyte of flash and two megabytes of ram. This is unfortunately not enough for more then one image. Additionally, there are no high-speed interfaces on the board. This implies a custom FPGA board will have to be manufactured. Currently, a custom PCI card with a Xilinx Vertex 4 FPGA, two Firewire interfaces, and one gigabyte of ram is being designed. Plans include writing preliminary software in C, and there is potential to port it to the GPU.

4.7 Results



5 Software

The focus of the software team has been to develop a simulation environment for testing and implementation of control ideas and code. The rational for the focus is the lack of mechanical components in the early stages. It has since grown to become a test base for control ideas and a stepping stone for high-level AI strategy development and testing. Software can be thought of as a combination of two components: simulation and control.



5.1 Simulator

The Simulator is a client/server framework for the creation and testing of our code. The server aspect contains a physics simulation and a game-play manager, while the client contains the control code. This allows the server to maintain control of all the referee controls, physics, and client communication. This further ensures consistency and allows the client to worry about the higher level code.

To communicate with the server, the client must implement the server's protocol. It is a three socket protocol for communication. The primary connection is TCP from the client to the server over a given port the server is listening on. The two secondary connections are UDP. One is for the radio data sent from the client to the server, and the other is for camera data sent from the server to the client. This allows the client and server to communicate these signals simultaneously and have them handled at the appropriate times.

The server's role is to accept client connections and communicate with the client about game-play and field data. It is the responsibility of the server to control the physics and facilitate the control of robots for the client. The server starts games once clients are connected and allows the client to run though scenarios quickly and see the result on the client side.

Once a connection to the server has been established, the client must actively receive camera data and send out radio data to the robots. It can also use the provided open-gl rendering library to draw the camera data for easy debugging

and idea testing. The client also contains the control code as well as the higher level decision code.

For our system a hierarchical software architecture will be implemented. The modules will include the low-level interface, the potential fields navigation, a simulated robot controller, and the AI system. The camera receives data on the state of the field and makes predictions on the robots and the ball's locations and velocities between frames using a physic model based on the simulator. The AI system functions as a coach calling plays based on the state of the field from this model. Each play is defined as an initial formation and a set of schema for each robot. The schema given by the coach will include tasks such as acquire ball and the set of behaviors necessary for accomplishing these tasks. Each task can be interpreted by the robots as goal states based on their own positions and/or internal states. The behaviors are derived from the tasks and premises such as move towards ball. The actual motion towards the physical positions are defined by either the tasks or the behaviors given to each robot and is implemented using a potential fields algorithm. In this algorithm, the goal has an positive, attractive potential, while the other robots on the field have a negative, repulsive potential. The path is determined by summing weighted vectors representing these potentials, and then the sum is applied to the robots current heading. The interface to the robots is done using the protocol developed specifically for the wireless system. A possible means of training the AI coach to assess which plays to execute for any given state will be the utilization of reinforcement learning algorithm based on Q-value optimizations. The AI coach will be evaluated and refined depending on matches against an opponent AI, which will be designed based on data provided by human players.

6 Future Development

6.1 Mechanical

Our first priority is to install the motors and kicking mechanism which allows us to begin testing our drive train and some of our navigation algorithms. We will experiment to determine the optimal kicking configuration, kicking speed, and solenoid power. Further experimentation will be done with the dribbler mechanism. These experiments will test for the optimal speed and geometry of the dribbler. In addition, we will test several materials to find a sufficient exterior covering for the dribbler to provide smooth and consistent control of the ball. A prototype robot has already been assembled, and now we are beginning the testing phase for the parameters previously mentioned.

6.2 Software

In the future, as a means of enhancing each robot's capabilities, tasks and behaviors will be implemented on each robot with respective positions being fed

continuously to them through the wireless system. In this system the robots themselves would generate their own paths themselves depending on plays selected by the AI coach. These robots would act with greater independence.

7 Conclusion

The RoboJackets are entering thier first year of competition in RoboCup and look forward to not only competeing but also to being the host school for 2007. We have already designned and built one protoype testbed and we hope to take the lessons learned in this prototype and implement them on a possible second revision. Our software team has already gained some experience working with the simulator and are working towards building the final software. The electrical, vision, and mechanical teams have also had some experience in development of the prototype.

8 References

J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmetation for interactive robots. Proc. Int. Conf. Intelligent Robots and Systems (IROS), Takamatsu, Japan, 2000, pp. 20612066.

C. E. Brodley and P. E. Utgoff. Multivariate decision trees. Machine Learning, 1995.

T. A. Brown and J. Koplowitz. The weighted nearest neighbor rule for class dependent sample sizes. IEEE Transactions on Information Theory, pages 617619, 1979.

W. Skarbek and A. Koschan. Colour image segmentation a survey. tech. rep., Institute for Technical Informatics, Technical University of Berlin, October 1994.

P. Jonker, J. Caarls, and W. Bokhove. Fast and accurate robot vision for vision based motion. Lecture Notes in Computer Science, vol. 2019, pp. 149-153, 2001.

M. Asada and H. Kitano. RoboCup-98: Robot Soccer World Cup II. Berlin, Germany: Springer-Verlag, 1999, Lecture Notes in Artificial Intelligence.