CheckerBot

The group was looking to create a robot (CheckerBot) that played checkers using artificial intelligence that was better than last year's robot. Last year's project was studied as well as different types of checkers A.I.s. David Fogel developed the A.I. Engine that is being mimicked\textsuperscript{1}. It was preferred that the robot to be able to play against a human player as well as other robots. Given additional time it would have been possible as well as a drastically improved A.I. Design. This project is fairly novel, commercially speaking. Commercially there are many checkers games that individuals move pieces across a board and the board senses the movements and then it tell notifies the player where it will move or with the use of magnets it makes its move. The CheckerBot is obviously better is the sense that the robot can physically move the pieces better than sliding magnets. Another way CheckerBot surpasses the other commercial options is through the A.I.. The A.I. design allows for great improvement over time.

Details:

Our functional partition can be seen in Figure 1. We would have a display that showed a game time read out as well as information about how long each player has taken for there turns. A push-button would be used to signal an end of turn, much like with

tournament play. A camera would be used to see the board and its configuration this would be sent to the JStamp where it would be processed and a representation of the board would be created. The A.I. on the JStamp would then calculate a move to be made. This would then be sent to the Robot's arm to make the move. It was thought that using a RF interface to a PC if the JStamp was unable to handle advanced A.I.s.

The software partition can be seen in Figure2. The main unit (Brain) does all the calculations for the neural net as well as the calculations for the image processing. The game algorithm handles the game rules. The camera interface is to control the camera and capture images. The Robot control controls the servos. And with an optional RF interface next moves would be calculated on the PC conditioned on whether the A.I. Brain able to fit on the controller.

First decision was what type of arm was to be used and the type of motors to use to drive the arm. Servos were the choice made. And the choice was also made to attempt to use Lego construction to build the arm. The arm was to be a jointed arm with a base servo, 2 lifting servos, and an elbow servo. Servos were chosen due to the ability to drive them to a position using changes in pulse width. After using the servos for while, we noticed that the servos have problems of their own. They allowed us to precision of .5 degrees of precision however they have about 5 degrees (10 times as many). This caused lots of problems when mapping the board. A functional mapping was attempted first but it was an inadequate solution and would not correctly map the way in which was needed. As a result squares were mapped on an individual basis. An additional problem with the servos was they needed to be attached firmly and the Lego’s needed to be reinforced so that the torque of the servos would not separate them. This required the purchase of additional Lego’s so that holes could be drilled and the servos attached. A plastic lid supported the prototype base, with the robot base being a 4” square. This placed too much leverage on the base servo and a lazy-suzan bearing was chosen as an appropriate option to support it. In the original design the CMUcamera was also mounted on the arm in addition to an electromagnet. This additional weight was greater than what the Lego constructed arm would lift using the servos. A search was made for a 4” bearing and one was not found the best option available was a 6” bearing. This caused an additional redesign of the base and the purchase of additional materials in the form of a large base plate and the screws necessary to connect it all together. The relocation of the CMUcamera due to weight and being unable to capture a frame that included the entire board required building a frame to support it. An adjustable frame was constructed using PVC piping and screws. This frame was not used due to the time constraints of presentation. Given additional time the image processing would be implemented and the robot would be able to play against a human opponent. The final component was having the electromagnet operating off of a GPIO pin on the J-STAMP this was implemented using a TP120 power Mosfet. The servos are controlled using GPIO pins also on the J-STAMP. The Robot is powered using a combination of three different voltages. The J-STAMP is powered using a 6 Volt supply. The Magnet is powered using a 12 Volt supply. And the servos are powered using a separate 6 Volt supply due to the current requirements of the motors. They are able to pull up to 2 Amps of current at full stall. This combination of 4 servos, 6” lazy-suzan bearing, electromagnet, and J-STAMP comprised the final Hardware solution for CheckeBot.

The choice of an AI was just as difficult. The Internet was scoured looking for a functional A.I. that would run on the J2ME and one was not found. Therefore a port of
the Joone A.I. (http://www.jooneworld.com) to the J2ME was attempted but was unable to be completed due to dependencies not available in CLDC1.0 library that was required for use of the J-STAMP. This resulted in a complete A.I. engine being developed from scratch. This development also included writing a large number of mathematical functions to enable the AI to function correctly. The A.I. takes approximately 10 hours per evolution and it is in its 18th evolution.

Final Project Results

What did you end up with?

Because of the problems encountered mapping the servos and with the low quality image from the camera, the camera option was not implemented. Instead the checker program played two random players on the board. The robot played a complete game with over 100 moves by human intervention only once (Robot Arm missed 1 checker) this was due to the inconsistencies in the servos that were used. With new changes made to the randomization functionality in the A.I. the CheckerBot now plays completely random competitive games with itself.

Did you meet your goals?

All of the goals were not met, mainly the ability for a person to play to robot. This would have required the Camera option documented earlier. In addition, a pushbutton would have been required so a playing person could tell the CheckerBot when they were finished with their move. The original goal was also to implement a clock allowing a person the option to set a time limit for how long a game would take. This was also not implemented because it would require a person to play with the CheckerBot.

On the other hand, the robot arm is working aside from some small mapping issues, which came up from play in the servos. The A.I. Engine is working exceptionally well giving completely random games. Overall the team is quite happy with the progress made on the CheckerBot considering the headaches caused by bad servos.

Summarize what you learned and achieved.

On the morning of the demo the base servo that was used for rotation across the checkerboard burned up as it was going through warm-up patterns. This was replaced and remapped in the last minute. This taught the group that anything can happen through the development of the project. It is also important to note the lessons learned in getting software to work with hardware that is not an exact process because of natural inconsistencies in the servos. The group also learned to coordinate team members who are working on different parts of the project. Anthony was working on the A.I. engine, Ricky was working on the hardware, and Peter was working on getting the two to work together. A lot was learned through this process.

Looking through the data sheets of the servos the group didn’t notice what the “Dead Band of 5usec” meant. After hours of dealing with inconsistencies, the group learned (very well) what “Dead Band meant. This caused numerous remapping of the servos to the board and also is the reason for going to a rest position between every move. We also learned that servo accuracy depended on the temperature of the servos. This was not in the servo’s datasheet so the only way to learn this lesson was through trial and error.
What would you do differently next time?

With the things learned, TBA-Tech would have done a few things differently. First off, the goal of having a camera working was an over-optimistic goal. This would have required at least another two weeks to implement. The choice to use all servomotors was a bad idea. If started over again, TBA-Tech would have chosen to use a stepper motor for the base servo. Dead-Band wasn’t a problem in the height and reach motors because they were working against gravity. Gravity was constant and made these motors hit the same end of the Dead-Band range every time. The base servo did not have this constant resistance so it couldn’t consistently hit the same part of the Dead Band range every time. A stepper motor would have been a better choice for this motor.

It also wasn’t known how lousy the camera was, so if given the option to start again, the choice for putting the camera on the robotic arm would have been abandoned. TBA-Tech would have opted to build a frame to hold the camera right from the start. This would give us the ability to get pictures from the exact same place every time.

Peter also started mapping the servos using a polar coordinate system. Because of variables in the servos, base and arm, this polar coordinate system didn’t work at all. This was aborted and a mapping function was written which could take angle values for each servo to convert to pulse widths. This probably could have been done with a Mat lab program that would take the extreme squares and give good guesses on expected values in the middle.

The group learned to work together and work through unforeseeable problems. Using Engineering skills TBA-Tech developed a working checker playing robot despite the problems encountered.
Figure 1
Figure 2