EEL 4665 – Intelligent Machine Design Laboratory

Formal Report: Deep Azure

Amanda Reilly

Instructors:

Dr. A. Antonio Arroyo

Dr. Eric M. Schwartz

Teaching Assistants:

Tim Martin, Ryan Stevens,

Devin Hughes, Josh Weaver

Date – April 19th, 2010

Contents

Contents

1. Abstract

Game-playing robots and computers have existed for quite a while. One of the first such computers

to gain media attention was Deep Blue, a computer from IBM that could beat chess grandmasters. The challenge when designing a robot that will play a game is balancing resources such as processing power and time, with the robot's skill. Although it is possible to create an index of every move possible in a game of checkers and choose the best possible move at every step, such a feat would require far more memory than is available on a robot, or even most computers of today's age. I chose to create a robot that will play checkers because game theory is always something that has been interesting to me, and that I have little experience in.

2. Executive Summary

Robots are particularly engaging to people when they act out human tasks. We look to see how accurately a robot can duplicate the actions of a human. When it came to creating a robot that would play checkers, I chose to build a robot with “stage presence”, that would move about on top of the game board and physically pick up the pieces. What follows in this report is a detailed summary of the process involved in creating the robot, including any pitfalls encountered or changes made to the design.

3. Introduction

Robots that can interact with people tend to be natural crowd pleasers. My original inspiration was to have a robot with "attitude". I was woken up one morning with the idea that I should build a robot that plays a board game, but gets enraged and quits whenever it loses. However, when thinking over what that project would involve, I decided that it would be more rewarding to just focus on having a robot play a board game. I chose checkers because I wanted to use magnets for game pieces, and thus the disk-shaped games pieces were appealing. The goal of the project is to build a robot that can sense the location of the pieces on the checkers board and then manipulate those pieces to play a game against a human opponent. To do this I will need to implement sensors, actuation, and learn

and implement game theory.

4. Integrated System

The envisioned robot would be located on an arm attached to tracks running along the side of a checkers board. The robot could move forward and backwards along the track, as well as back and forth along the arm, to move across the entire board. The game pieces would be magnets, with north allocated to one color and south to the other. After the human indicates that their turn is over using a bump switch, the robot would complete a quick pass over the board to determine the current state of the board, and then decide its move. The robot would sense the pieces using sensors sensitive to magnet field. Once its move is decided, it would move to the piece and pick it up by lowering and powering on an electromagnet. The magnet may be covered by a waveguide to prevent the magnetic field from picking up neighboring pieces. After its turn is completed, the robot would signal to the human that it is their turn. If the human obstructed the path of the robot during its turn, Azure would stop and tell the user to remove the obstruction.

5. Actuation

To actuate the robot I have used servos, and two stepper motors. I chose servos for their ease of use and accuracy. I can use the servos to position the electromagnet. The stepper motor would be useful to move along the arm, where there is no frame of reference, and I was able to borrow a powerful stepper motor for moving the gantry. An electromagnet was chosen to pick up the pieces because of its reliability. Several other suggestions were made, including hooks, but none so far have struck me as reliable. Using magnets as the game pieces is an intuitive solution that lends itself well to various aspects of the problem. The robot itself shall move about the board in a fashion similar to a CNC machine.

6. Sensors

In order to enable the robot to interact with the world around it, I intended to use several sensors. These include bump switches, hall effect sensors, and an IR sensor. Additionally, a motor encoder will be used for the stepper motor.

- Bump switches – a simple sensor composed of a tactile switch connected to a pull-down resistor. These can be used to calibrate the home position of the stepper motor, and potentially for another level of collision avoidance as well.

- Hall-Effect sensors – these sensors can detect magnetic field. These are generally only sensitive to fields of one polarity, so I would need two for each of the 8 columns the robot will sweep. Interestingly, the sensors were only readily available in omnipolar and unipolar, so I must use one of each and some rudimentary logic to determine if the robot is sensing a north or south pole (or none!). One additional Hall-Effect will be used to detect a line of magnets alongside the board, to determine position.

- IR sensor – IR sensors detect heat and thus can be used to tell if something hot (a person or a flame, for example) is nearby. I have found a short range model that will do well for avoiding hands that are near the playing field.

- Motor encoder – these encoders act as self-contained feedback systems. They can detect the position of the motor and control the new position. I have chosen a stepper motor driver that is compatible with 3.3V logic and has self-contained current limiting.

7. Behaviors

Azure will be designed to play checkers with a human opponent. Thus its behavior can be divided into several stages. The first stage is for sensing. After the human makes a move, he will hit a switch or

otherwise indicate that he is finished with his turn. The robot will then make one pass across the board. The sensing for a row will be triggered when it detects the presence of a new row, using the embedded column of magnets. Ideally it will take a snapshot of every row without stopping, making the travel time relatively short. The sensing will be complete when it has reached the other side of the board from its home position. The second stage involves the game algorithm. Knowing where the pieces are, Azure will try to calculate the best move. I will most likely implement some sort of tree-based algorithm and have a timeout after which the robot will simply make the best move it has encountered so far. Another student alerted me to the existence of computer algorithms that cannot lose at checkers, but I decided at the time to create my own. First of all, my robot will not have the processing power that advanced computers have, nor will it have a lot of time to make its move. Second, if I use another person's algorithm, I don't learn anything. However, as time has passed it has seemed more likely that I will need to use someone else's checkers algorithm, as I will have barely enough time to code the other behaviors. The third stage is the execution stage. Once it knows its move, the robot will position itself to the piece it wants to move, pick it up, and move it to its proper place. If it is jumping the opponents' pieces it will set the piece down in every appropriate location and then remove the jumped pieces. Time will determine how gracefully it removes the pieces; it may just dump them over the edge of the board. If it has kinged one of its pieces it will place a second piece on top of the first, though I will be implementing that functionality only after everything else works. Once its turn is completed it will return to the home position and indicate to the human that it is their turn.

Some changes took place from this ideal goal in the actual implementation of the robot. Because I did not anticipate how involved the robot's mechanical design would be, I had little time to flesh out the programming logic involved. Many features such as kinging were left out, and I was ultimately unable to even code for jumping. In the end, the robot was able to scan the board and make a valid move (but not a jump).

8. Experimental Layout and Results

The platform of the robot was completed far behind schedule, as its mechanical complexity was something that I had not accounted for. Regardless, some testing was completed in terms of the sensors.

- ☒ Sensor test: I enabled the stepper motor array and did many laps across the board with the hall-effect array enabled. In the end it was determined that the array was mostly accurate with an intermittent issue in column 2, where the microprocessor would report back seeing a north piece where there was none there.

- ☒ Motor test: As the stepper motor coupled to the threaded rod would be moving with no reference beyond the limit switches that trigger at the limits, I needed to know that I could move accurately. I had the stepper motor move the threaded rod from one limit to the other and count the number of steps.

Results:

Calibration Testing	
1	11663
2	11665
3	11664
4	11666
5	11663
6	11665
7	11662
8	11662
9	11667
10	11670

This table shows that the stepper motor is very accurate. However, the variable I chose for to keep track of the number of steps must have overflowed, because when I went to actually move the stepper motor, it took about 14,000 steps to move the servo platform across one square!

9. Conclusion

Although Deep Azure was not completed fully by the end of the semester, I would like to continue working on it. As I will need to return Tim's motors and I will lose access to the machinery in MIL once I graduate, this may be a difficult task. Regardless, the project taught me so much. Although I had an incredible amount of support from the TAs and my friends, I was also on my own for most of the project. Unlike my other classes, when I started a new C project for the programming, there was no direction from a professor or helpful tips on what to do. It was me and the datasheets, trying to figure out what I wanted to do and how to accomplish that. Although I was not always successful, I feel much more confident that I can do these things.

Deep Azure was ultimately able to scan the board, communicate what move it wanted to make, and make that move. There were some glitches, but I would call it a success.

10. Documentation

[1] Datasheet for Unipolar Hall-Effect Sensor -- http://www.diodes.com/datasheets/AH182_AH183.pdf

[2] Datasheet for Omnipolar Hall-Effect Sensor -- http://www.allegromicro.com/en/Products/Part_Numbers/3212/3212.pdf