Team 10:

Chexter

Shane Brewer, Rahul Jinturkar
smb2197, rj2512

Final Report

Robotic Manipulation
MECE 6615 Spring 2018

I. Introduction

Chexter is a portmanteau of Checkers and Baxter and indicates the goal of the project: play checkers with the Baxter robot. The project involves computer vision, robotic joint and pose control, feedback loops, a simple “head-up” GUI, and close integration with an implementation of Arthur Samuel’s AI checkers engine.

II. System Description

As expected, the system utilizes Baxter’s cameras as inputs for running the game. This begins with setup. A preset standby position was created with each of Baxters hands placed a few feet above the environment. From this position, one of Baxter’s cameras was used the survey the environment.

First the checkerboard is placed in front of the Baxter on a table and the position and orientation of the board is determined. Given the size of the board and the limits of Baxter’s range, there is a small area in which the board can be placed. To assist the person in placing the board, an image is displayed on Baxter’s head to indicate unreachable corners.

Once the board is placed, the pieces can be set and the game can begin. The same standby camera position surveys the board for the board state - i.e. which pieces are in which squares. During a human’s turn, all valid moves are extracted from the checkers engine. To assist the human, the valid moves are graphically represented on Baxter’s head. The vision module waits until the board state matches one of these moves before alternating to Baxter’s turn.

Baxter waits for the checkers engine to select its move and then executes. This operation contains a sequence of operations. Given a piece to move (an origin and destination square), Baxter moves its gripping arm to an approach position. This approach position is calculated based on the origin square and the initial board position. The original goal involved calculating the position of a game piece using the surveying camera. Unfortunately, the accuracy of this calculated position has too much error (up to 1+ inch) to grasp the piece in question. Instead, a fine adjustment is made using feedback from an image from the hand camera of the gripping arm. This pre-grasp image is much closer to the piece and is sufficiently accurate for calculating a grasp position. Baxter is then able to grasp the piece and move it to an approach position for the destination square. A similar fine adjustment is made at the destination square before placing the piece.

If the move involves a jump, Baxter picks up the jumped piece and removes it from the board.

The checkers engine is an implementation of a depth-adjustable search using minimax functions ranking possible future board states. Per Arthur Samuel’s groundbreaking AI implementation, it uses a 36-bit strings to represent pieces and uses bitwise operations to search for and execute moves.

Rather than create custom game pieces as originally intended, Baxter plays the game with an off-the-shelf set of checkers pieces. However, the grip distance of the stock Baxter gripper is too large for the pieces. Custom fingers were designed and 3D printed to allow the Baxter to grasp the game pieces.

III. Qualitative Results

Ultimately, the goal of the project was to create a full checkers game-playing Baxter. Given this, the project was completely successful. A checkers game with a human was played from start to finish (Baxter won…).

The game is not without its kinks. The most critical flaw involves the board state detection. Based on lighting conditions, piece placement, and a variety of functional parameters, the board state detected is susceptible to false positives and false negatives. The lighting in the environment requires a high camera gain, which creates very light and very dark sections of the board. Some image processing is done to improve this, but it isn't error free. To mitigate these errors, the vision program is tightly integrated with with the checkers engine. This means the program knows what board states it is looking for and only accepts one of those states. The downside of this is it forces a very structured game. The program cannot start in a random game state and does not allow invalid moves.

The inaccuracy of the Baxter's motions also forced delays in execution. When the Baxter approached a piece, it then needed to capture an image to calculate a fine adjustment. There is a small length of time after the Baxter finishes its move command that it is still moving. Without waiting for a steady hand, this extra motion led to large errors in the fine adjustment.

IV. Quantitative Results

The initial goal metrics are outlined here:

- The Baxter will succeed in identifying 75% of objects.
- The Baxter will identify the positions of objects within 0.5 inch.
- The Baxter will succeed in grasping 75% of objects.
- The Baxter will succeed in moving objects to the desired position within 0.5 inch.

These goals were overly conservative and would not have enabled successful gameplay. The results were as follows:

- The Baxter successfully identifies 100% of pieces.
- The Baxter identifies positions of objects with 0.1 inch.
- The Baxter successfully grasps 98% of pieces.
- The Baxter successfully moved objects to desired position within 0.25 inch.

The first point is caveated by the problem mentioned in section III. There were several instances when the vision program did not successfully recognize all game pieces in a single still image. However, given general noise and the refresh rate of the camera (20+ Hz), this did not manifest as a critical problem during gameplay.

A note on the third point: three full games were played with the Baxter, each of which contain over 30 (Baxter) pick operations. Of these games, the Baxter bumped one piece before grasping it, but was still able to successfully grasp, move, and place the piece.

V. Extensions

There are a number of areas where the project can be improved.

i. Faster Gameplay

Baxter's motions may have room for optimization via velocity control. If the Baxter slowed on approach, it may be able to capture images sooner that still enable fine adjustment.

ii. Game Setup

If the pieces were all placed it range of the Baxter arm, a script could be written to enable the Baxter to set up the game

iii. Kinged Pieces

With the current finger, two pieces can be moved at a time. It would be possible for Baxter to find jumped pieces off the board. It would then need to precisely place the second piece on top of the existing piece to be "kinged." This would enable more intuitive gameplay but would require a higher level of precision than currently exists in the program.

iv. Moving Board

The board is assumed to be fixed once it has been detected during setup. A more robust implementation would allow the board to move during gameplay without issue. Early attempts to implement this feature resulted in slower execution of other operations.

v. Random Start

Allowing a random starting board state may be useful in learning to play the game.

VI. Conclusion

The result of the project is a successful implementation of a Checkers playing robot. Computer Vision and Robotic Manipulation are combined to create a seamless game-playing experience. Though rudimentary, Chexter was successful in achieving its goals as initially formulated.

VII. Appendix

Images: