Chess Robot with Visual Recognition

Background

Currently, there are numerous chess robots available. These robots have robotic arms that are capable of moving chess pieces based on an AI algorithm. However, none of them are actually automatic in that they do not detect chess pieces as humans do. Most of the chess robots require manual input of chess piece locations, which evidently shows that they are not automatic. Some robots use specialized equipment, such as DGI (Digital Game Technology) chess boards, that allow detection of locations based on electromagnetic interaction between the pieces and the board. However, these specialized boards and pieces have cost limitations, so for my project, I avoided digital chess sets and instead used image processing to detect and identify pieces. The main objective in this project is to create an artificial intelligent robot that can visually recognize and move chess pieces just as humans do by developing a visual recognition algorithm and implementing it to an actual chess robot.

Methodology

There are many challenges in visual recognition of chess pieces, as suggested by Cheryl Danner and Mai Kafafy from Stanford University suggested in their research paper on visual chess recognition. Since chess pieces do not have texture, matching through simple edge detection or even through SIFT/SURF descriptors will not give good results. Camera angle has also been a problem. An overhead view of the chessboard has limitations in that it is impossible to distinguish a piece from another. A horizontal view is also a poor choice since it would be impossible to check on which squares the pieces lie. Even a diagonal camera view poses a problem that the pieces seem to overlap, as shown in Figure 1.

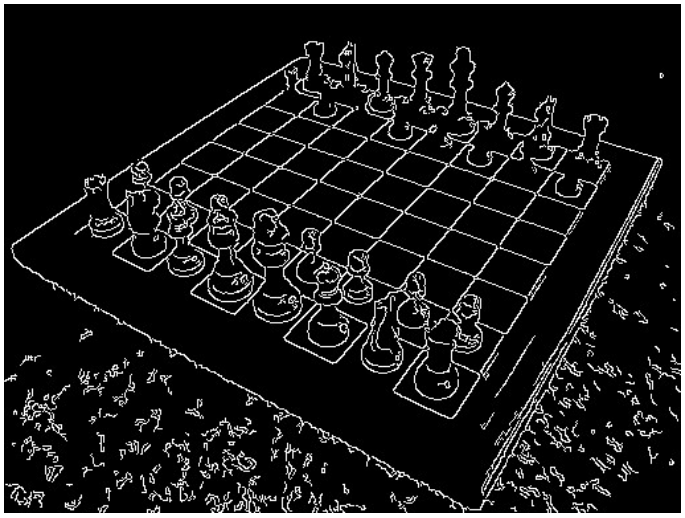

Figure 1. Canny edge detection of chessboard showing overlaps of pieces.

Previous chess robots, instead of trying to identify each piece, have used a simple algorithm to just check if a piece exists at a given square. Since players can only move one piece per turn (two in castling), an overhead visualizer is able to recognize which piece was moved assuming the game started from a standard setup. Thus, the Scale Invariant Feature Transform algorithm, which uses Euclidean vector distances and Gaussian functions to scale images and compare objects, have been used as a viable method to compare the images of an empty square and a nonempty square to determine the locations of the chess pieces [1].

The limitation of using the visual detection method above is that the robot cannot directly detect pieces, since it relies on tracing the movements. Thus, in situations like advancing a pawn, it is difficult to detect into what piece the pawn has changed. This tracing technique is also limited in that it relies heavily on the standard initial board setup. As a result, identifying and analyzing positions of a specific configuration of pieces would be impossible without the entire record of chess moves.

The solution to all the problems posed above can be handled by Danner and Kafafy's approach. Danner and Kafafy argued that polygonal approximation and shape context are not

good shape representation of chess pieces since they all have similar elongated shapes (when pictured from a diagonal view), with only fine variations in their boundaries [2]. Thus, the optimal shape representation method for identifying chess pieces is 1-D shape signature, which represents the 2-D objects with a 1-D function. By using Fourier coefficients of this signature, we will be able to distinguish pieces based on the image contours.

The algorithm used for this project can be split into two main parts. The first part is obtaining image contours of the pieces using various methods. For this project, I used Canny edge detection, color detection, and Otsu's method of segmentation. The second part is calculating normalized Fourier coefficients and comparing them to the chess piece database. The identification process involves comparing the Fourier coefficients for all three types of features and using majority rule. For example, if a particular piece is identified to be Queen by canny edge detection but is identified to be King by color detection and Otsu's method, the final decision will be King by two-thirds majority.

Results

The following images show the best 50 feature matchings of a queen (Figure 2) and a collection of several chess pieces (Figure 3) determined using the Canny edge detection (Figure 4), color detection (Figure 5), and Otsu's method of segmentation (Figure 6). The threshold value for Otsu's method was adjusted to separate the white pieces from the black.

Figure 2. Queen.

Figure 3. Collection of pieces (in clockwise direction from the left: black pawn, white rook, black king, white queen, white king).

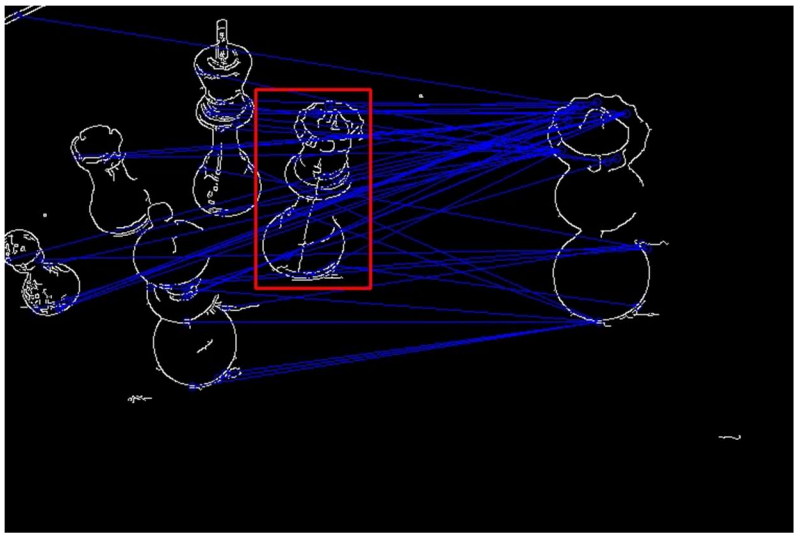

Figure 4. Feature matchings of the queen using Canny edge detection.

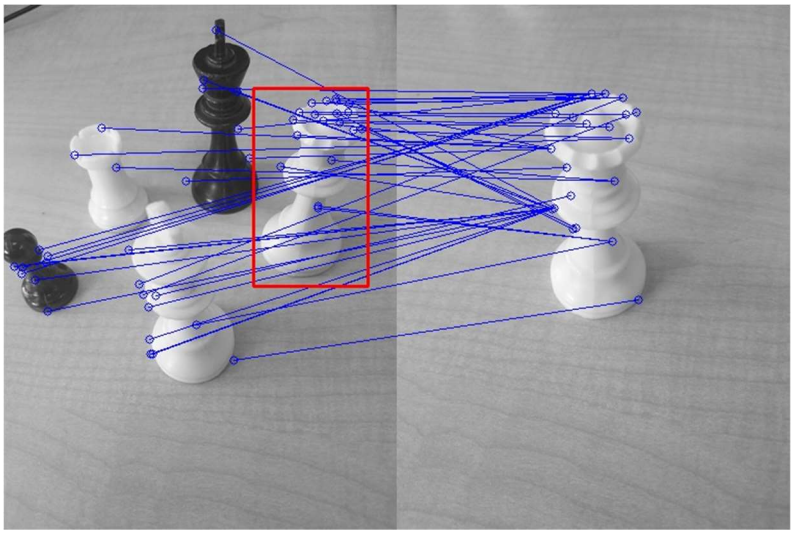

Figure 5. Feature matchings of the queen using color detection.

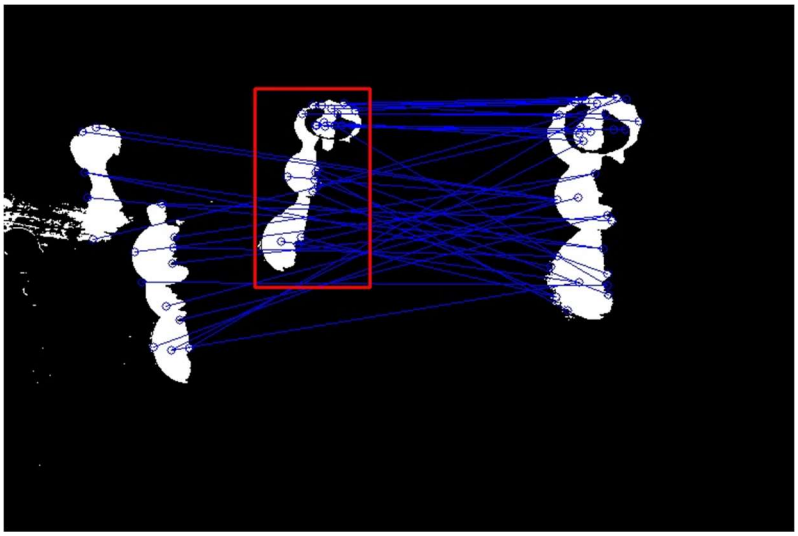

Figure 6. Feature matchings of the queen using Otsu's method of segmentation.

The images above indicate how difficult identifying a piece is. Similar pieces, such as a king and a queen, show several overlapping key points that may cause errors and misidentifications. Table 1 shows the number of matches out of 50 to specific pieces for each type of feature matching. The pieces with most matches for each feature is boxed in red.

Table 1. Number of feature matches.

	Canny Edge Detection	Color Detection	Otsu's Method
White King	14	11	16
Black King	12	7	0
White Queen	12	21	28
White Rook	3	3	6
Black Pawn	8	8	0

Analysis of the images of feature matchings suggest that the best way to determine the type of chess piece is to observe the top half of each piece. Notice that all the chess pieces have a similar bottom in that they are all rounded and circular. The top portion is what makes each piece unique. Thus, I modified my code to focus on the top side in order to improve my algorithm, which produced the following table.

Table 2. Number of feature matches using top portion of the pieces.

	Canny Edge Detection	Color Detection	Otsu's Method
White King	6	2	8

Black King	6	4	0
White Queen	7	19	23
White Rook	3	3	4
Black Pawn	3	5	0

The table above demonstrates that the identification process is improved drastically when we use only the top portion of the pieces to match the features of the pieces. Instead of 2 out of 3 features associating the piece to a queen in the original feature matching, all 3 out of 3 features indicated queen in the improved algorithm.

Discussions

The images and the tables of the feature matchings of the pieces demonstrate that image processing approach is a viable option in detecting chess pieces. My algorithm was able to successfully identify pieces and was even more accurate after the implementation of using only the top portion of the pieces. The only problem was that there were inevitable issues in detecting pieces on a chess board. Since I was using only one camera with a diagonal view, pieces that seemed to overlap were impossible to identify.

There are a lot of improvements that can be done to my algorithm. One is to simply increase the number of methods used to obtain image contours so that the probability of misidentifying is decreased. Another improvement is to conduct weighted evaluations to the feature matches. In the case of ambiguous or indecisive feature matches, the feature with most

number of matches shall be used. For instance, if a piece is identified rook with 3 out of 10 matches for Canny edge detection, bishop with 6 out of 10 matches for color detection, and pawn with 9 out of 10 matches for Otsu's method, it should be identified as a pawn.

The knowledge of how many pieces are on the chessboard can also be utilized for improvement. For instance, if there are 3 Queens and 1 King detected on a regular chessboard game, then we know that one of the Queens must be a King. Once these errors are detected, backtracking and overriding the decisions based on percentage error can significantly reduce misidentifying the pieces.

For further research, I would like to apply my algorithm to an actual chessboard. After resolving the issue with overlapping pieces, I would like to implement my code to a chess robot. By working on the mechanical portion of the chess robot, as well as the software that connects the robotic arm to my algorithm, I would be able to develop the desired robot that can take pictures, identify pieces, calculate its moves, and then execute the moves.

References

[1] Lowe, David G. “Object Recognition from Local Scale-Invariant Features.”

<http://www.cs.ubc.ca/~lowe/>

[2] Danner, Cheryl and Kafafy, Mai. “Visual Chess Recognition.”

<https://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Danner_Kafafy.pdf>