Chess Detection

Lin Yusen, Yuan Yuan, Song Jiafei, Weng Rusong, Qin Xin

December 29, 2016

1 Introduction

In the project we implemented chess detection, using corner detection, intensity difference and Haar feature calculation enabling computer-human chess playing feasible, featuring chessboard detection and reconstruction, chess piece detection and timed based on game timer.

2 Filter and Cell Detection

Cell is the basic component of a chess board, and we know that there are 64 square cells in a chess board. Half is white and half is black.

2.1 Median filter

To detect cells, we should first use the median filter to remove noise in the chess board. After the median filtering, its time to detect cells.

2.2 Harris corner

Its easy to detect edges and corners of cells than cells themselves because gradients of them are huge, especially on a white and black chess board. So we use the MATLAB function corner to detect Harris corner points in the image.

Figure 1: detect Harris corner points

Many corner points have been labeled and only corner points inside the chess board are what we want. Points on the edge of the board are just noise and useless to reconstruct the chess board.

2.3 Haar filter

Some prior knowledge is useful in the problem: black cells and white cells are arranged regularly. Each intersection point between cells should be the corner of four cells, two black cells and two white cells. In this case, Haar feature is useful to represent this property. Obviously, this type of Haar feature is very similar to a chess board.

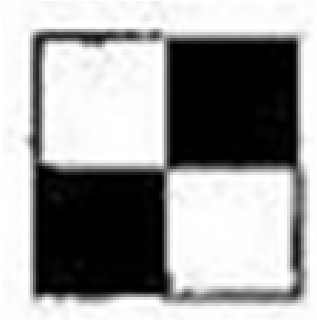


Figure 2: Cell arrangement

To the center point in figure, its Haar feature is the difference of the sum of white area pixels and the sum of black area pixels The size of the window and the threshold are parameters. Corner points with Haar feature smaller than the threshold are not target points. After the filtering, we get 49 points represent the intersection of cells.

3 Chess Board Reconstruction

After applying corner detection to the chess board, we get at least forty-nine corners. In this step, we want to assign index to these points.

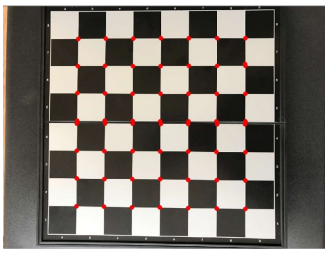


Figure 3: points

Left-Top corner: $\min(x+y)$
Left-Bottom corner: $\min(height - y + x)$
Right-Top corner: $\min(width - x + y)$
Right-Bottom corner: $\min(height - y + width - x)$

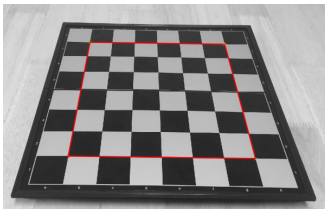


Figure 4: square

Other points are the linear combination of these four points. The origin is left-top point. For example: To the bottom line , we can divide this line with six part. So we can get the point in this line by :

$$X_k = k * \frac{X_{Right} - X_{Left}}{6} + X_{Left} Y_k = k * \frac{Y_{Right} - Y_{Left}}{6} + Y_{Left} \quad (1)$$

We can get each blocks index by using mod operation. By applying above method ,we can deal with alignment of the chessboard.

4 Chess Piece Detection

We find that if there is no chess on the cell the grayscale of the pixels in the cell should be similar. And if there is a white chess on the black cell or black chess on the white cell, it is easy to detect. The problem is to detect the chess on the same-color cell. We find that the shady face of the white chess will much darker than front face. Due to reflection, some are on the black chess will be much lighter.

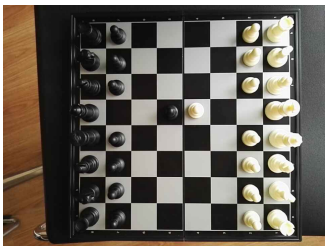


Figure 5: detect the chess on the same-color cell

So we find the max grayscale and min grayscale in the patch which is a little smaller than the cell. We judge there is a chess if

$$\max grayscale - \min grayscale > threshold \tag{2}$$

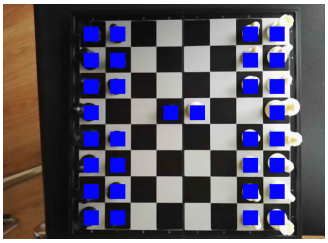


Figure 6: judgment and the patch

Figure above shows the judgment and the patch. We save the result of pieces detection in a binary state matrix.

1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	0	0	1	1	0	0	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1

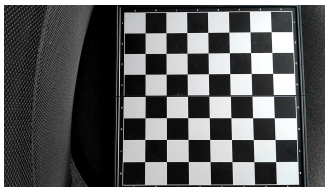
Figure 7: binary state matrix

5 The process of playing detection

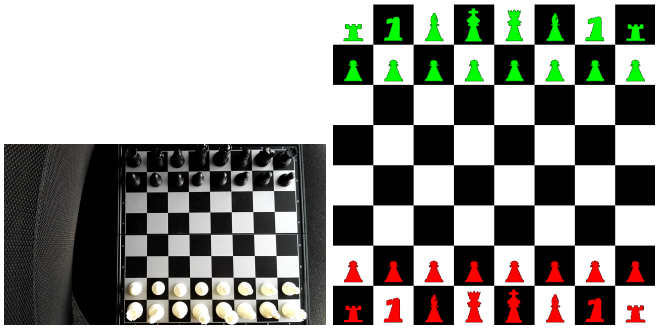
After the above steps, we can detect the location of each chess.

So we can make a state matrix with 8*8. 1 means existing chess in that block and 0 means not. After each move , we will get a new state matrix ,then minus it with the previous one. After minus , 1 means where the chess moving to and -1 means where the chess coming from

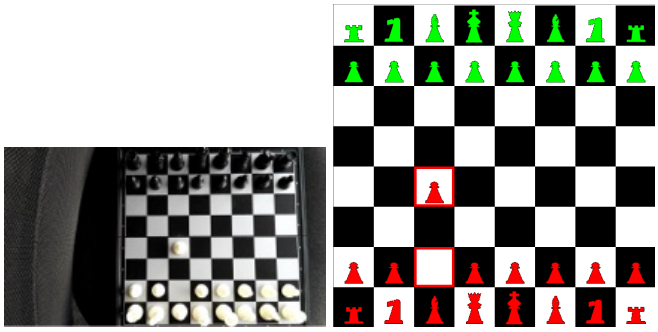
Step 1: detect the chessboard to locate the chessboard



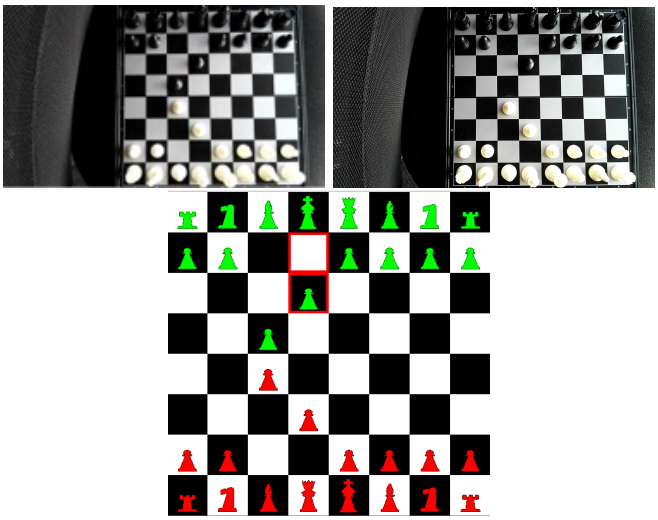
Step 2: initialize the chess

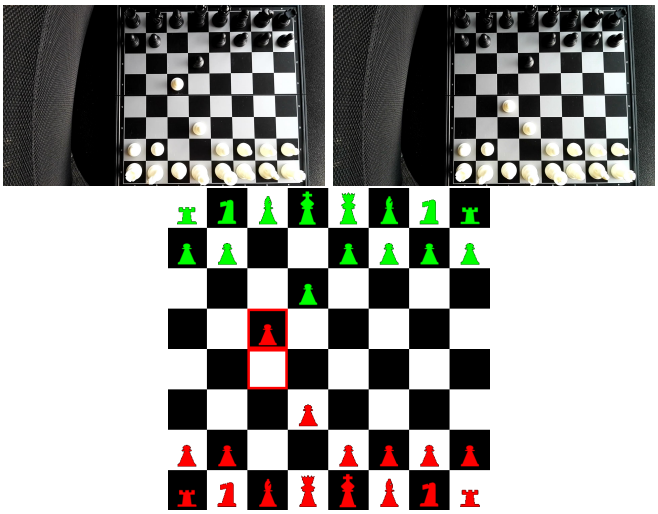


Step 3: move the chess



Step 4: Eat the chess First move the eaten chess and then move the eat chess to that position





5.1 Photo transformation

In fact, we take photos instead of videos. We use an APP called smart webcam. Code for matlab to use the smart webcam:

```
imread(    <IP address>\image    );
```

keyboard singal which symbolizing the action of pressing the chess timer. Therefore the whole process can be conclude to be

1. Take a photo of empty chess board

2. Take a photo of initial state

3. Take a photo after each movement.

6 Data Visualization

I save the position of each chess piece in matrix. I also use matrix in Matlab drawing the chess board. Use the alpha channel to draw the chess piece on the board with green and red color.

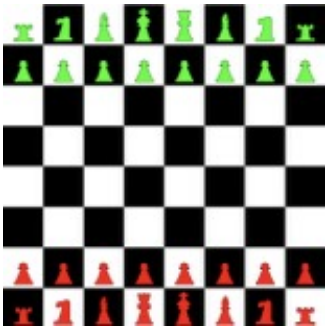


Figure 8: draw chess piece on the board

When a chess piece moved, red case will be drawn to highlight.

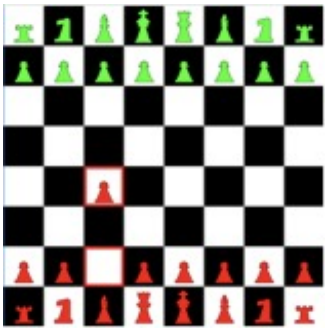


Figure 9: highlight red case

7 Real Time Demo Making

7.1 set the location of smart phone, chessboard

Figure 10: location of smart phone

Figure 11: location of chessboard

7.2 search for ideal light condition

Figure 12: light condition

Found out natural light alone was better.

8 Conclusion

8.1 Our improvement on chess board detection

According to a reference paper we search from Stanford, Hough transform is a method to detect the chess board. But the result of the paper is not good, only two third of chess lines are detected and the author completed the chess board according to known lines. Thats not precise enough and when we imitate the method the result is even worse. So we think if the intersection points of cells on the chess board can be precisely detected and other noise can be totally concluded, things will be easy. We learn face detection by Haar feature classifying in the computer vision class and it is nature to realize the similarity between the chess board and the Haar feature window. After a little adjustment on Haar feature, it works on the chess board detection, and this method has not been used before.

8.2 Improvement can be done on chess board construction

In the latter search, we can use ransac with standard chessboard to assign index to the chessboard without locating four points first.