Coordinate and Characters Recognition of Chinese Chess

Lingxiao Zhang
Electrical Engineering
Wuhan University
Wuhan, Hubei, China
zhanglingxiao@whu.edu.cn

Zimeng Qiu
Electrical Engineering
Wuhan University
Wuhan, Hubei, China
ChrisQiu@whu.edu.cn

Muwei Zheng
Electrical Engineering
Wuhan University
Wuhan, Hubei, China
jamewayne@whu.edu.cn

***Abstract*—Chessboards arise frequently in computer vision theory and practice because their highly structured geometry is well-suited for algorithmic detection and processing. In this paper, we present an algorithm that can correctly recognize the state of a Chinese chess game by processing a photo of the chessboard. Some major steps of the algorithm include chessboard rectification using Hough transformation, geometric operation and edge detection, chess piece detection using circular Hough transformation, chess piece recognition using color classification and template matching.**

Keyword: Edge Detection, Hough Transform, Template Matching

I. Introduction

Chinese chess is a very popular strategic board game in many Asian countries. It is played between two players, each in control of one side, black or red. Chinese chessboard has 9 horizontal lines and 10 vertical lines, which intersect 90 points.

On each side, there are 16 chess pieces in 7 different types. So in total, there are 32 chess pieces in 14 different types. The goal of the project is to develop an algorithm that can automatically recognize the state of the chess game from a photo.An example input is given in Fig. 1.

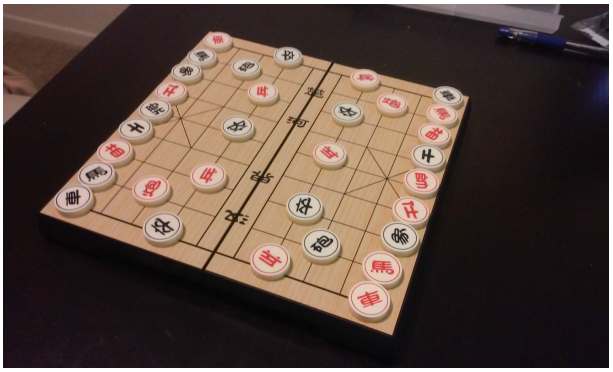

Fig. 1. An example input

At the same time, we has another photo without any chess piece as a reference image, showed as Fig. 2.

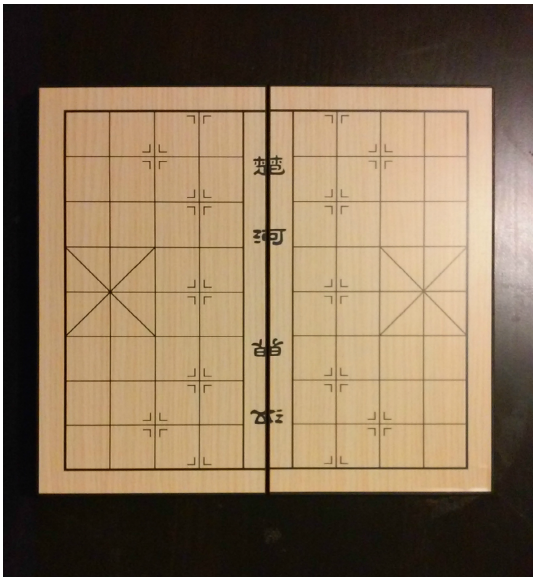

Fig. 2. A reference image input

II. Algorithm

In this project, we provides a specific method for Coordinate and state recognition of Chinese chess. The algorithm details are presented in the following sections.

A. Chessboard Segmentation

We first transform a color image to a gray image, then use Sobel Operator to achieve edge detection.

1) RGB Image to Gray Image: We use MATLAB function:rgb2gray() to finish the transformation. The result is as Figure. 3.

2) Edge Detection: Edge detection includes a variety of mathematical methods that aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges. The same problem of finding discontinuities in one-dimensional signals is known as step detection and

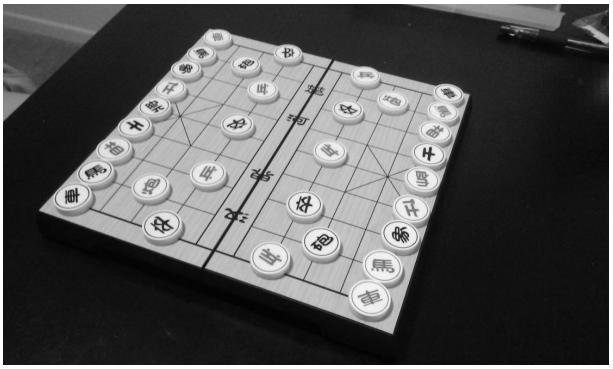

Fig. 3. Result of gray transformation

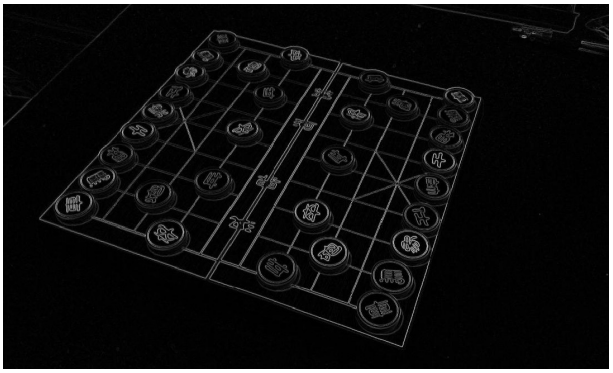

Fig. 4. Result after edge detection

the problem of finding signal discontinuities over time is known as change detection. Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction.

The Sobel operator, sometimes called the Sobel operator or Sobel filter, is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasising edges. It is named after Irwin Sobel and Gary Feldman, colleagues at the Stanford Artificial Intelligence Laboratory (SAIL). It was co-developed with Gary Feldman at SAIL. Sobel and Feldman presented the idea of an "Isotropic 3x3 Image Gradient Operator" at a talk at SAIL in 1968. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation that it produces is relatively crude, in particular for high-frequency variations in the image.

In this case, respectively, we set Sobel operator as:

$$sobelKernelY = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{1}$$

$$sobelKernelX = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2}$$

Using these two templates to extract the horizontal and vertical edges of the grayscale map and then to superimpose two graphs. The result are as Figure. 4.

B. Geometric operation

In geometry, an affine transformation, affine map or an affinity (from the Latin, affinis, "connected with") is a function between affine spaces which preserves points, straight lines and planes. Also, sets of parallel lines remain parallel after an affine transformation. An affine transformation does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line.

Examples of affine transformations include translation, caling, homothety, similarity transformation, reflection, rotation, shear mapping, and compositions of them in any combination and sequence.

If X and Y are affine spaces, then every affine transformation f: $X \to Y$ is of the form $x \to Mx + b$, where M is a linear transformation on X and b is a vector in Y. Unlike a purely linear transformation, an affine map need not preserve the zero point in a linear space. Thus, every linear transformation is affine, but not every affine transformation is linear.

$$(x, y, z) = (\hat{u}, \hat{v}, \hat{w}) \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{3}$$

All Euclidean spaces are affine, but there are affine spaces that are non-Euclidean. In affine coordinates, which include Cartesian coordinates in Euclidean spaces, each output coordinate of an affine map is a linear function (in the sense of calculus) of all input coordinates. Another way to deal with affine transformations systematically is to select a point as the origin; then, any affine transformation is equivalent to a linear transformation (of position vectors) followed by a translation.

In this process, first of all, we need to determine the corners coordinates of the above two chessboard. By using the line detection, we can get the longest line of four ends which are the four corners of chessboard.

Use the two corners as the target point for tilt correction to change an oblique rectangle into a non-oblique rectangle. A transformation matrix is obtained from the four points of the original rectangle and the four points of the new rectangle, then to transform the global image by using this transform matrix. The nearest neighbor

interpolation is used to fill the image vacancy and adjust the direction of the image. Fig. 5 shows the result after Affine transformation.

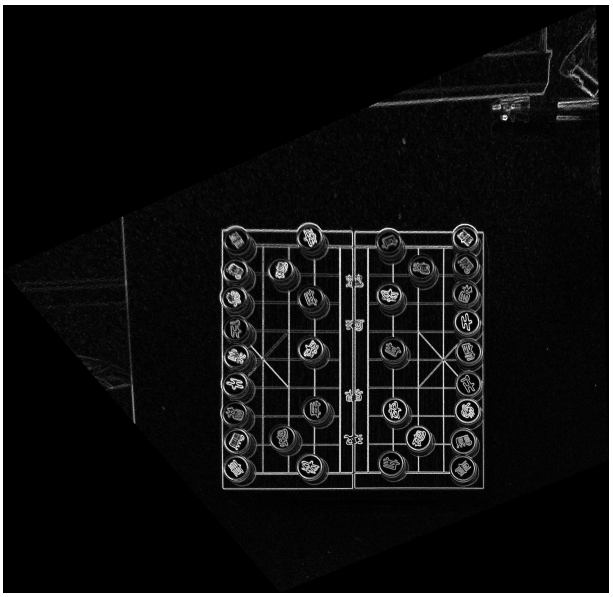

Fig. 5. Result after Affine transformation

C. Chessboard Rectification

In order to finish next steps of detection, we need to first rectificate the chessboard. Firstly, using erosion processing, we can make the chessboard lines more clear. Secondly, we binarize the image using Otsus Method. Here, we assume the background is darker than the chessboard and a global threshold is used. Tiredly, we use median filter to get rid of noisy points in the image.

Figure. 6 shows the result of the rectification.

D. Chess Lines and Points Detection

Lines are another natural local image feature exploited in many computer vision systems. Geometrically, the set of all lines in a 2D image can be parametrized by polar coordinates (ρ, θ) describing the distance and angle, respectively, of their normal vectors with respect to the origin. The discrete Hough transform exploits this idea by transforming a spatial image into a matrix in (ρ,θ)-space whose (i,j)-th entry counts the number of image edge points that lie on the line parametrized by (ρ_i, θ_j) As such, one can detect lines in an image by simply searching for local maxima of its discrete Hough transform.

Hough transform on the original image, and select the largest 24 points ,which representing the 24 chessboard lines (9 horizontal +10 vertical +5 Border lines). At the same time, we obtain the interval between checkerboard points by using the Ref processing results. In this case, the transverse interval is 114, and the longitudinal interval is 125. At the same time, the grid coordinates of

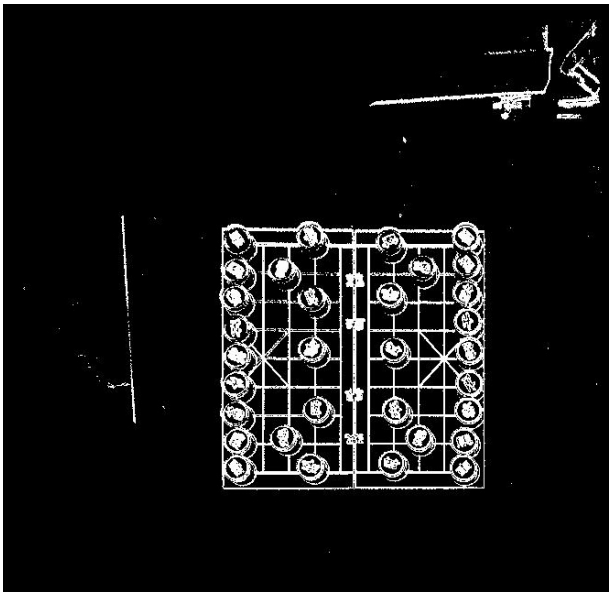

Fig. 6. Result after Chessboard Rectification

(1,1) are determined by the processing results of this Test image, where the (1,1) point coordinates are (1014, 1035).

Figure. 7 shows the result of Hough transform for the reference image.

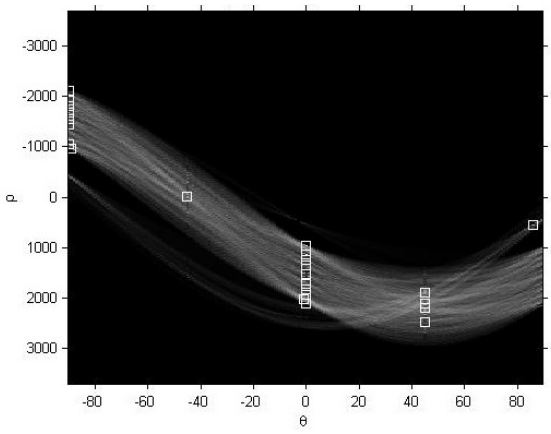

Fig. 7. Result after Hough transform

E. Chess Pieces Detection

By using the (1,1) grid coordinates, the horizontal spacing of the grid points and the longitudinal spacing of the grid points, the position coordinates of each of the 10*9 grid points on the chessboard can be obtained. At the same time, imfindcircles() function is used to detect the location of the round chess pieces, and the location of the center of the circle and the coordinate position of each grid point are determined as a certain range, so as to determine the location of the chess pieces. In Figure 8, a red circle is used to indicate the chess pieces, and the green line indicates the chessboard line and the

blue star indicates the chessboard spot. Thus, we can recognise the coordinate of every chess pieces by judging if chessboard spots and centers of chess pieces coincide in certain range.

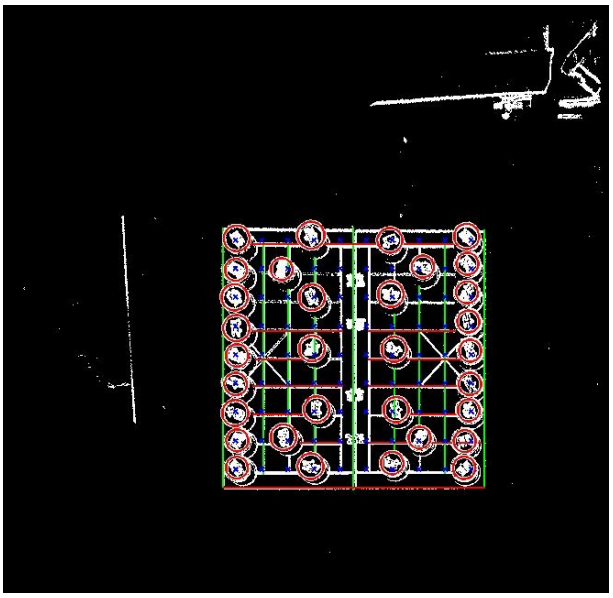

Fig. 8. Plot of chess pieces, grid points and chessboard lines

F. Color Classification

Gray and red channel image of original image are respectively prosessed by edge detection and affine transformation in the same way to ensure the two picture coordinates coincide.

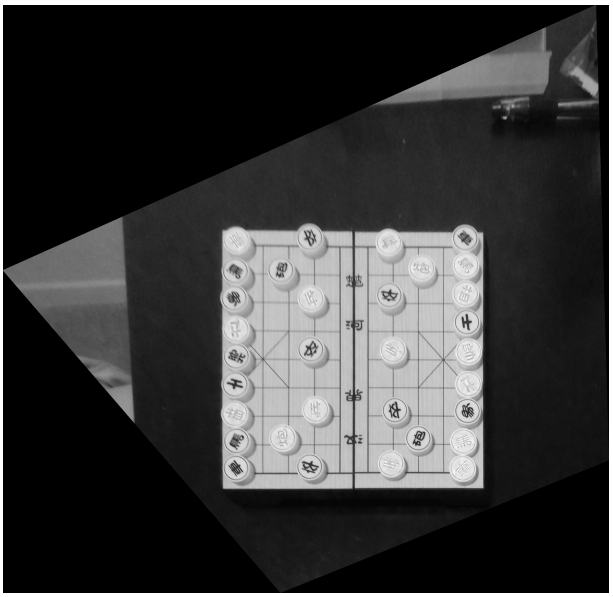

Fig. 9. Red channel image after affine transformation

Then binaryzation processing is used on the red channel image. At this point, in the red channel, the red pieces almost close to the white and the black pieces are still black, so the effect of binaryzation is very good.

Because of the same edge detection and affine transform processing on the grayscale image, we can use imfindcircles() function to find all circles in the image.

Since the coordinates of the two images are the same, we can also calibrate the circles found in the grayscale image into the red channel image, and get Figure. 10.

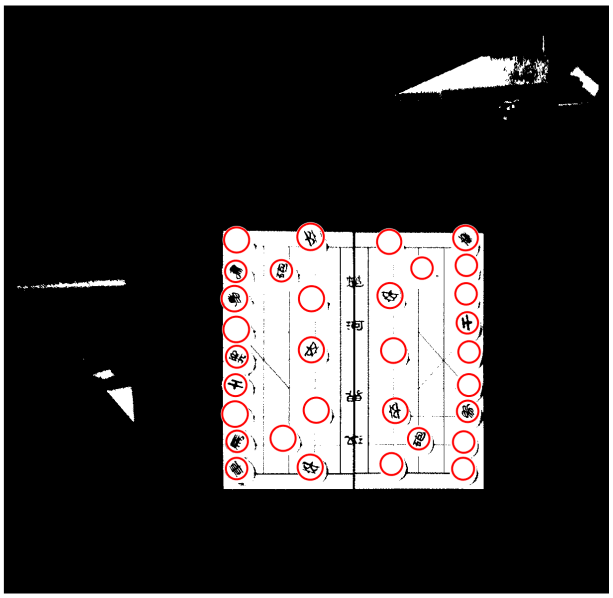

Fig. 10. After red channel image binarization, the circle will be calibrated in the image

Fig. 11. Recognition of a black chess piece

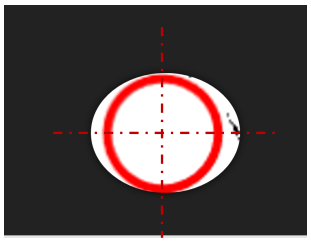

Fig. 12. Recognition of a red chess piece

From the image above, we can get the center coordinates and radius of all pieces (the circle in this image).

We can observe that the radius of Chinese characters circumcircle is about the 3/4 of whole piece. But in order to prevent the error caused by light during the binarization process (such as the light is too dark, the binarization may cause a huge area of black).

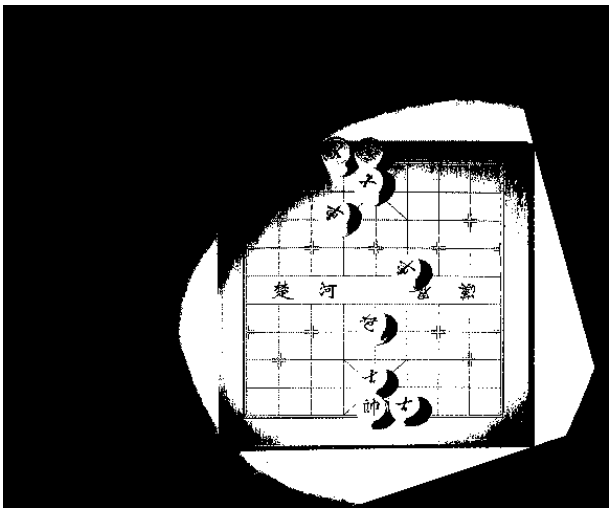

Fig. 13. An example of image binarization with errors

$$Pixels = \sum_{t=-\frac{r}{2}}^{t=\frac{r}{2}} P_{(x,y+t)} \tag{4}$$

Using the chess board coordinate system dealing with red channel map. And then binarization processing the red channel map, at this time, because in the red channel, the red pieces almost close to white, and black pieces are still black, so the binarization effect is very good. So now the focus is coming, through the above figure we can get the coordinates of the center points and radius of all pieces (ie, circles in the figure) It can be seen by observation that the radius of the circumcircle of the middle Chinese characters of the chess pieces is about 3/4 of the radius of the whole piece. But in order to prevent errors caused by light during the binarization process, for example, some areas too dark, the binarization processing will cause a large area on the pieces are black. So here take half of the average radius of the pieces as a statistical range, mark its length as r, within this range, assuming the coordinates of the center point of the piece is (x, y), traverse all the pixels on this diameter, from (x, y-r) to (x, y+r). If the number of black pixels on this line is greater than the number of white pixels, we think that this piece appears as black. Otherwise, it is a red chess piece. The results are as Figure. 11 and Figure. 12.

G. Characters Recognition

After the last step to obtain the chess coordinates, the greyscale image is directly carried out with the same affine transformation as before, and then process by binarization, as shown in Figure. 14.

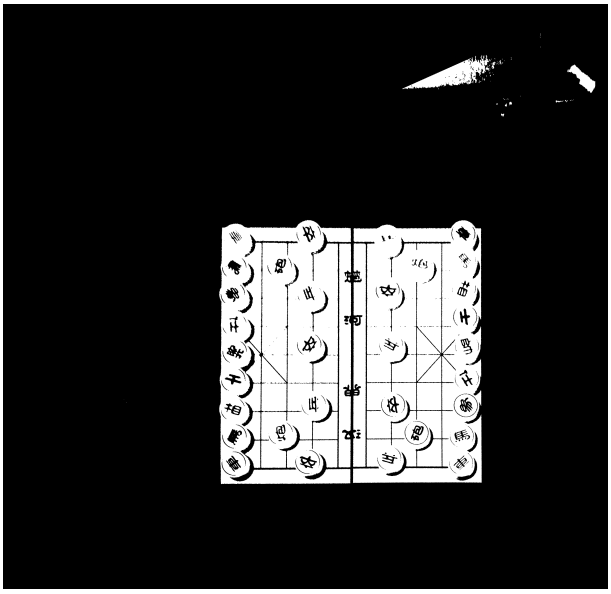

Fig. 14. Binarization grayscale images after affine transformation

Character template library for a variety of fonts through the same processing on multiple images, has the size of 20x20 pixels, black background and white Chinese characters.

Due to the limitation of the template, we need to process the binarization grayscale images after affine transformation. First of all, according to the center coordinates and radius of chesses, we use the imcrop () function to intercept a 3/4r x 3/4r rectangular of the image, which must ensure that contains the entire character. Then use the imresize () function to change its size into 20x20, which is consistent with the template size. Then the image will be inverted by black and white to obtain the character to be recognized.

Fig. 15. Samples of the character to be identified

Next, perform ”and” operation on the pretreatment character image imageU and character image imageT which comes from template library to get the common portion of the image imageV; then perform ” XOR ” operation on imageV and the image obtained with the character to be recognized, to obtain the excess part of recognition character image, imageX; perform ” XOR ” operation on imageV and the template character to get the excess part of the template image, imageW.

Then we calculate the white pixel number (T) of each template character image (imageT), the white pixels number (U) of imageU, the white pixels number (V) of and the number of common part (imageV) of imageU and imageT, the white pixels number (W) of imageW; the white pixel number (X) of imageX;

The structure discriminant function is expressed as:

$$\frac{V_i}{\frac{W_i X_i}{T_i U}\sqrt{\frac{(T_i - TUV_i)^2 + (U - TUV_i)^2 + (V_i - TUV_i)^2)}{2}}} \tag{5}$$

In this equation,

$$TUV = \frac{T + U + V}{3} \tag{6}$$

The template M, corresponding to the maximum similarity coefficient with max (Y), is the character to be recognized.

III. Conclusion

According to this algorithm, we can identify the coordinates of each of the red or black pieces of test image provided with 100% accuracy,

We also identified two other example. One example has a recognition rate of 100%,

While the rate of successful recognition of another test image is much lower, which is under 60% as shown in figure 18. It because that there are too much shadow in this image. It reveals that there still remains much work for us to do to identify chess board and pieces in more complex situation.

References

[1] Comparison of Edge Detectors:A Methodology and Initial Study. Health A.,Sarkar S,Sanocki T.et al. Computer Vision and Image Understanding . 1998

[2] Use of the Hough transformation to detect lines and curves in pictures[J]. Richard O. Duda,Peter E. Hart. Communications of the ACM . 1972 (1)

[3] A Road Detection Technology Based on Reverse Perspective Transformation[C]. Liu, Xinlei ; Cheng, Zeng Mu ; Yi, Feng Yan. 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). 2016

[4] Research on Automatic Visual Inspection Method for Character on Cartridge Fuse Based on Template Matching[C]. Yuanyuan, Zou. 2016 3rd International Conference on Information Science & Control Engineering (ICISCE); 2016, p527-531, 5p. 2016

[5] An Examination of Character Recognition on ID card using Template Matching Approach[C]. Ryan, Michael Hanafiah, Novita. In International Conference on Computer Science and Computational Intelligence (ICCSCI 2015), Procedia Computer Science 2015 59:520-529. 2016


```
R,Ju(1,1)
B,Ma(1,2)
B,XiangB(1,3)
R,ShiB(1,4)
B,Jiang(1,5)
B,ShiB(1,6)
R,XiangR(1,7)
B,Ma(1,8)
B,Ju(1,9)
B,PaoB(3,2)
R,PaoR(3,8)
B,Zu(4,1)
R,Bing(4,3)
B,Zu(4,5)
R,Bing(4,7)
B,Zu(4,9)
R,Bing(7,1)
B,Zu(7,3)
R,Bing(7,5)
B,Zu(7,7)
R,Bing(7,9)
R,PaoR(8,2)
B,PaoB(8,8)
B,Ju(10,1)
R,Ma(10,2)
R,XiangR(10,3)
B,ShiB(10,4)
R,Shuai(10,5)
R,ShiR(10,6)
B,XiangB(10,7)
R,Ma(10,8)
R,Ju(10,9)
```

Fig. 16. Binarization processed grayscale images after affine transformation

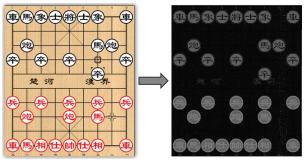

Fig. 17. Binarization processed grayscale images after affine transformation

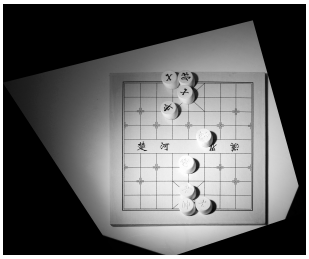

Fig. 18. Binarization processed red channel map after affine transformation