21^{st} *Computer Vision Winter Workshop*
Luka Čehovin, Rok Mandeljc, Vitomir Štruc (eds.)
Rimske Toplice, Slovenia, February 3–5, 2016

A Computer Vision System for Chess Game Tracking

Can Koray
Department of Computer Engineering
Başkent University
Ankara, TURKEY
cannkorayy@gmail.com

Emre Sümer
Department of Computer Engineering
Başkent University
Ankara, TURKEY
esumer@baskent.edu.tr

Abstract. *In this paper, we present a real-time system that allows the detection of the moves of a chess game. In the proposed approach, each captured video frame, from a RGB webcam positioned over the chessboard, is processed through the following steps; the detection of the corner points of the chessboard grids, geometric rectification, chessboard position adjustment, automatic camera exposure adjustment, intensity adjustment, move detection and chessboard drawing. All steps were implemented in MATLAB programming environment without using any chess engine. The proposed approach correctly identified 162 of 164 moves in 3 games played under different illumination conditions.*

1. Introduction

There are many systems of computer vision, which require algorithms to be able to recognize different objects and scenes. Since, chess game has become an interesting issue in terms of human-computer interaction systems, a computer vision system is needed for chess playing and chessboard recognition system.

There are various published techniques related to chess-playing systems. Sokic and Ahic-Djokic [11] proposed a computer vision system for chess playing robot manipulator as a project-based learning system. The proposed algorithm detects chess moves by comparing frames captured before, during and after a move, and finds the difference between them. In a similar study, Ataş et al. [1] developed a chess playing robotic arm system composed of various modules such as main controller, image processing, machine learning, game engine and motion engine of robot arm. In their study, the top of the pieces are uniquely designed to be different from each other in order to track by the system.

On the other hand, in a study conducted by Bennet and Lasenby [2], the recognition of chessboards under deformation was carried out. Their method determined a grid structure to detected vertices of a chessboard projection. Further, the same authors developed a feature detector named 'Chess-board Extraction by Subtraction and Summation (ChESS)' to respond to chessboard vertices [3]. In a different study, a chessboard recognition system was proposed [8]. The proposed system was applied to chessboard in order to identify the name, location and the color of the pieces. Piskorec et al. [10] presented a computer vision system for chess game reconstruction. The system reconstructs a chessboard state based on video sequences obtained from two cameras.

The tracking of the chess moves can be regarded as the preliminary task before designing a robotic chess playing system. In the literature, there are several efforts that perform the chess move tracking. The studies conducted by Matuszek et al. [9], Urting and Berbers [12], Cour et al. [4] and Gonçvales et al. [7] use unique algorithms to identify the chessboard grids along with the classification of squares. These methods are not only based on corner detection but also rely on having a clean background.

In this paper, we propose a real-time chess game tracking system using a RGB webcam positioned over the chessboard. In general, the move is detected by comparing the occupancy grids based on average color information of the pieces and the squares. Before that, several pre-processing steps are employed including geometric rectification, intensity adjustment and chessboard position adjustment. The system also works successfully under different illumination conditions by means of automatic camera exposure adjustment. Besides been a tracking system, the

proposed system can also perform 2D reconstruction of the chessboard states and generate movement logs.

2. Equipment and Setup

In this work, a setup is prepared to detect the moves of the pieces during the game. The setup has the Logitech c310 webcam for the capturing footage. The camera which has 5 megapixels resolution is capable of HD 720p recording. The camera has no autofocus functionality. Only the exposure mode from the camera settings is changed to 'manual mode' for move detection process. The webcam was used on a mid-range notebook. The chessboard and pieces are selected to meet World Chess Federation (FIDE) requirements in terms of color and size [6]. The board and the pieces have different colors from each other. The colors of the pieces are black and white, while the board has dark and light brown colored squares. The camera is positioned over the chessboard by a long and flexible holder as shown in Figure 1.

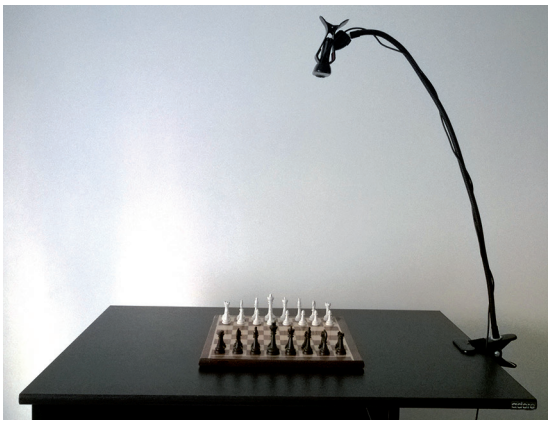


Figure 1. The image of the setup

3. The Overall Framework

The general block diagram of the proposed system is given in Figure 2. The details of the steps of the proposed framework are given in the further subsections.

3.1. Chessboard Grid Corner Detection

In this process, the first step is to find all grid corner points of the chessboard (Figure 3(a)) by using the snapshot of the camera. To find grid corners (Figure 3(b)), we used detectCheckerboardPoints function of MATLAB. The function that is particularly used in camera calibration gets an RGB image as an input and returns the located grid corners and the size of the board as an output. Until all grid corners are located, the saturation value of the captured image is increased gradually as a pre-process step. Once all grid corners are located, the second step is to locate the chessboard corners (point-C in Figure 3(c)). The grid corner points which are closest to the corners of the image are selected as pivot points. Point-A in Figure 3(c) is one of the pivot points. The diagonal closest inner point to the point-A is point-B, which is shown in Figure 3(c). The reflection of the point-B over the point-A is the point-C, which is the one of the chessboard corners as shown in Figure 3(c). This procedure is applied for all remaining corner points.

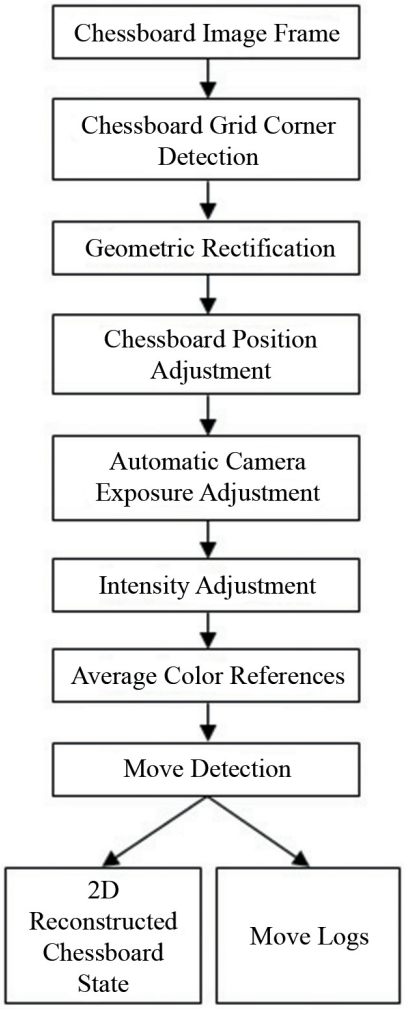
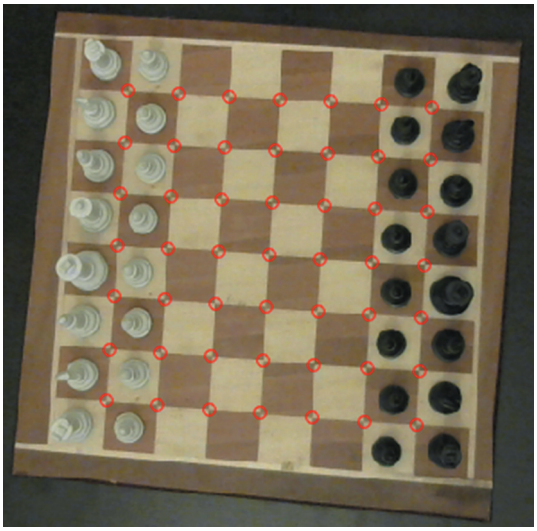


Figure 2. The overall framework

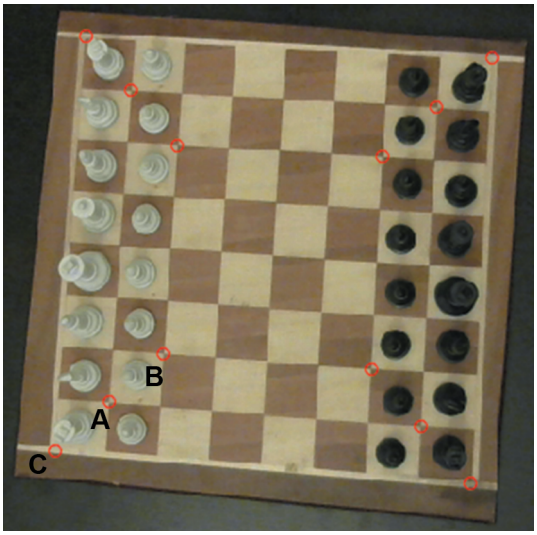
3.2. Geometric Rectification

The geometric rectification is an essential step to isolate the chessboard from the environment and correct the perspective distortion of the chessboard to pave the way for the other processes. The chessboard

(a)



(b)



(c)

Figure 3. (a) The original chessboard image, (b) the detected chessboard grid corners and (c) the related points to chessboard grid corner detection

is warped from its corner points which are located in the previous section to coincide with our predetermined size square corners (480x480px) (Figure 4).

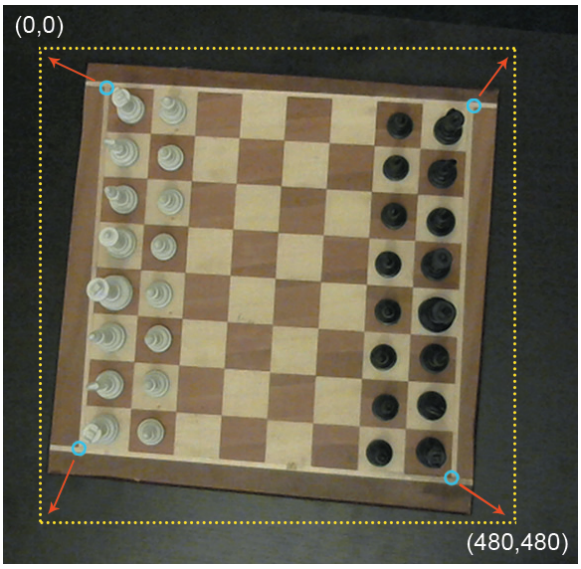


Figure 4. The chessboard before geometric rectification step

This process is applied only once before the game starts therefore, either the camera or the board should not be moved during the game. The geometrically corrected chessboard is presented in Figure 5.

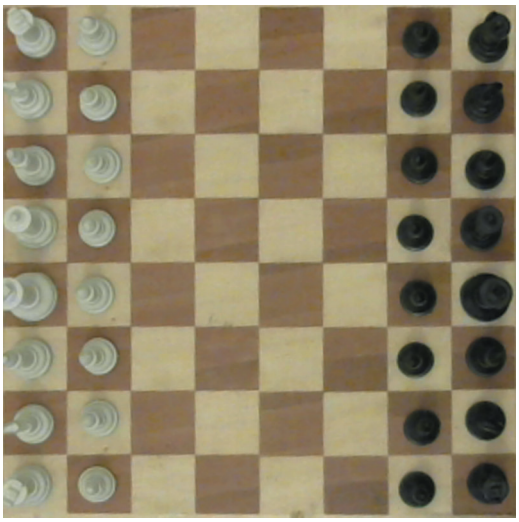


Figure 5. The chessboard after geometric rectification step

3.3. Chessboard Position Adjustment

To ease the calculations of the future processes, the white pieces are needed to be positioned at the bottom of the view. Thanks to camera position, we know that the positions of the pieces have to be on the left and right side of the camera. The comparison of the average colors of the both side's king square gives

us the position of the white pieces. According to the white side position, a new transformation matrix is computed to be used in the future warping processes. In Figure 6, the white pieces are located at the bottom while the black ones are at the top.

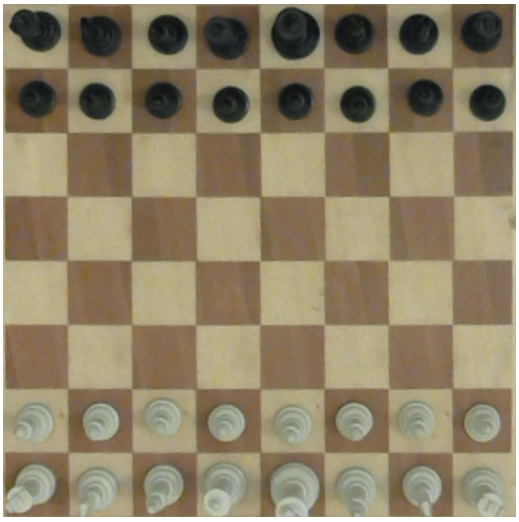


Figure 6. The chessboard after chessboard position adjustment

3.4. Automatic Camera Exposure Adjustment

The built-in automatic exposure mode of the camera may cause undesirable image acquisition for the move detection. In this mode, the camera continuously adjusts the exposure level according to the captured footage. Especially, whenever the player makes a move, the camera changes its exposure level due to the player hand on the captured image. In addition, the exposure level which is adjusted by built-in automatic exposure mode of the camera can be under or overexposure. In order to find optimum exposure level of the camera, it needs to be adjusted manually at the beginning of the game. The aim of this process is to get correct color values as much as possible by preventing under and overexposure situations. We proposed our automatic camera exposure algorithm that aims to find the optimum exposure level which maximizes the average of the color differences between light/dark piece and square (Figure 7). The calculated optimum exposure level is set to camera as a new exposure level for the following processes. In the present case the computed exposure level was computed to be -6 where the full range is between -9 (the darkest) and 0 (the lightest). The snapshot of the chessboard after applying the computed exposure level is given in Figure 8.

```
0  Begin Procedure: FindOptimumExposureLevel
1  FOR EACH Exposure Level
     2  CALCULATE the average color of dark squares, dark pieces, light squares and light pieces
     3  CALCULATE the color distance between the average color of dark squares and dark pieces
     4  CALCULATE the color distance between the average color of light squares and light pieces
     5  CALCULATE the average of the color distances
     6  IF the exposure level is the first exposure level THEN
          7  SET the average to the maximum average
          8  SET the exposure level to the optimum exposure level
     9  ELSE
          10  IF the average is greater than the max average THEN
                11  SET the average to the maximum average
                12  SET the exposure level to the optimum exposure level
          13  ENDIF
     14  ENDIF
15  ENDFOR
16  RETURN the optimum exposure level
17  End Procedure: FindOptimumExposureLevel
```

Figure 7. The pseudo-code of the automatic camera exposure adjustment

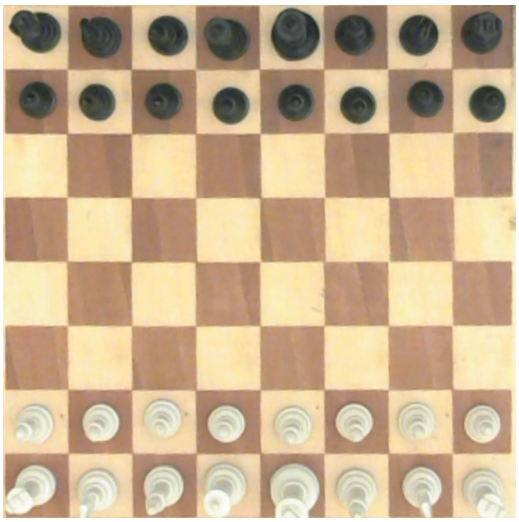


Figure 8. The chessboard after automatic camera exposure adjustment

3.5. Intensity Adjustment

To improve the image quality, a set of enhancements is applied to the snapshots of the camera. The first one is to reduce the noise problem. We used a 5x5 median filter to minimize the noise level of the images. The second one is to increase the saturation of the image to enhance colors. After this process, the average colors of pieces and squares are calcu-

lated to be used in further processes. The image of the chessboard after the intensity adjustment step is illustrated in Figure 9.

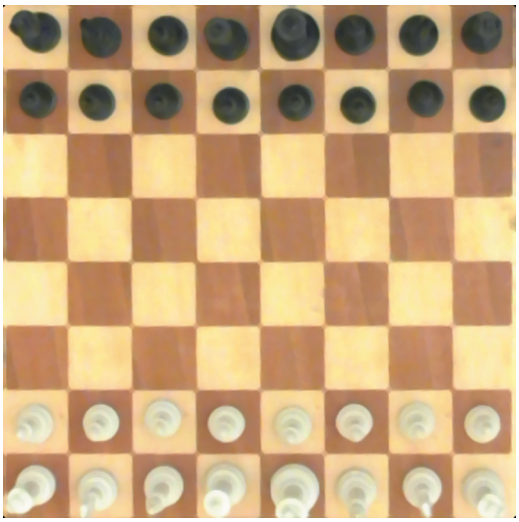


Figure 9. The chessboard after intensity adjustment

3.6. Average Color References

After all enhancements, in order to get color values of each square of the chessboard, the image of the chessboard (Figure 9) divided into 64 identical pieces each in correspondence to a square of the board. Therefore occupancy grids are created for the chessboard. After that, it is defined a region of interest (ROI) for each square (grid) of the chessboard. The primary aim of using ROIs is to get color information of the piece. ROI is defined as a 25x25px rectangle from the center of each square as shown in Figure 10.

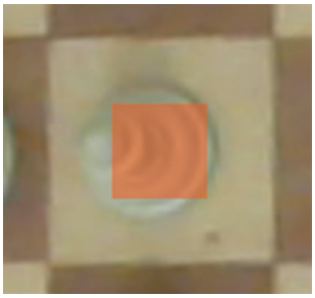


Figure 10. The orange colored region of interest superimposed on the pawn

At the beginning of the game, before the move detection, the average color values of the light/dark pieces and squares are received and recorded as reference values.

The reference colors of each type of piece and square calculated as follows:

- Reference color of the light pieces is calculated by taking the average of the 16 squares that are occupied by the light pieces.
- Reference color of the dark pieces is calculated by taking the average of the 16 squares that are occupied by the dark pieces.
- Reference color of the light squares is calculated by taking the average of the 16 light squares that are not occupied by any pieces.
- Reference color of the dark squares is calculated by taking the average of the 16 dark squares that are not occupied by any pieces.

3.7. Move Detection

The implementation of the move detection is based on a comparison between the reference image and the snapshot of the camera. For this process, the reference image is used as the first snapshot which is taken after each valid move. The first reference image is regarded as the first snapshot of the footage. During the game, the average color difference is calculated between the reference image and snapshots. Whenever the result of the calculation exceeds a predetermined threshold, we conclude that the player makes a move. After the result goes down below the threshold, we assume that the player finished the move.

At this point, the last snapshot is interpreted to determine the color and position of the pieces. Before this process, the last snapshot is warped and the enhancements are applied to the warped image of the chessboard. The ROI within each square of the image is compared with the four reference colors which are determined in section 3.6. In this comparison, the color differences are calculated in Lab color space by computing the deltaE value that represents the Euclidean distance of the related items. As a result of the comparisons, the reference color that gives the minimum deltaE value determines if a grid cell is a square or a piece with light or dark color. By applying this process to all squares of the chessboard, the chessboard state of the last snapshot is revealed. The state of the last snapshot and the previous chessboard state are compared to detect the move of the piece. The previous chessboard state represents the chessboard state of the last valid move. At the beginning of the game, the first state of the game is stored as the previous chessboard state.

When the state of the snapshot and the previous chessboard state are compared, six different outcomes can be obtained:

1. If there is no difference between previous and last states, this means there is no change in the game. For this reason, the color difference over the board is not a move.

2. If there are only one occupied and only one unoccupied squares difference with the same piece color then this is a move.

3. If there are two occupied and two unoccupied squares difference with the same piece color, then this is a special move called 'castling'.

4. If there are one occupied and one unoccupied squares difference with the same piece color and one unoccupied square difference with the other piece color, then this is another special move called 'en passant'.

5. If there is only one unoccupied square difference and if there is a piece color change to the previous piece color of the unoccupied cell in any other occupied square, then this is a capturing move.

6. For all other conditions, the result of the comparison is not a move.

If the result is a move then the state of the chessboard is updated as the last chessboard state. The move is added to the move list and the last state of the chessboard is reconstructed in 2D. An example 2D state reconstructed from a test game and the move list are presented in Figure 11. The moves are logged as standard algebraic notation which is the notation standardized by World Chess Federation (FIDE) [5]. Note that all the steps of the proposed methodology including the graphical user interface were implemented in MATLAB.

4. Experimental Evaluation and Discussion

In order to test the system, three chess games are played at different times having different illumination conditions. In these tests, 162 moves of all 164 moves are successfully detected by the system. The corner points of the chessboard are successfully located in all games. The system performance was found to be satisfactory to detect moves in real-time.

Figure 11. The reconstructed chessboard state with move list

The saturation enhancement which is applied to the images taken from capturing footage helped to increase the accuracy of the average color differences.

The combination of the lighting, camera settings and chess set are playing a big role in the success of detecting moves in a chess game. Although the proposed system works well under different illumination conditions, lighting environments (having a single light source) that cast strong shadows over the board are unsuitable for tracking.

On the other hand, shadows over the light pieces are another important problem. This makes difficult to separate the light pieces from the light squares, as in 2 of 164 undetected moves during the experimental evaluation. In addition, this problem may cause to get incorrect results from the automatic camera exposure adjustment method.

Shadows and specular reflections over a particular area of the chessboard can break the uniformity of the colors. In these conditions, a chess game cannot be tracked by the proposed system. Besides, due to the reference colors of pieces and squares are determined at the beginning of the game, the overall illumination of the environment should not change dramatically during the game. Otherwise, the move detection cannot be possible by the system.

5. Conclusion

In this paper, we have presented a real-time system that performs the detection of the chess moves. The preprocessing steps are found to be quite useful.

In particular, automatic camera exposure adjustment highly reduces the color ambiguities. The environments which are heavily under the influence of directional lights are not recommended because of casting strong shadows. The results of the played games indicate that the proposed system can be an affordable and efficient option among chess game tracking systems.

As an addition to the current system, a chess move validation system is under progress to interpret the player moves. By this way, the system not only tracks the position of the pieces but also validates the movements according to the type of the piece. Therefore, the future system can be used to help decision making and monitoring by referees and anti-cheat committee.

References

[1] M. Ataş, Y. Doğan, and İ. Ataş. Chess playing robotic arm. *Proceedings of the IEEE 22nd Signal Processing and Communications Applications Conference*, pages 1171–1174, 2014. 1

[2] S. Bennet and J. Lasenby. Robust recognition of chess-boards under deformation. *Proceedings of the 20th IEEE International Conference on Image Processing*, pages 2650–2654, 2013. 1

[3] S. Bennet and J. Lasenby. Chess – quick and robust detection of chess-board features. *Computer Vision and Image Understanding*, 118:197–210, 2014. 1

[4] T. Cour, R. Lauranson, and M. Vachette. Autonomous chess-playing robot, 2006. 1

[5] FIDE. Handbook, 2015. Laws Of Chess. 6

[6] FIDE. Handbook, 2015. Standards of Chess Equipment and Tournament Venue. 2

[7] J. Gonçalves, J. Lima, and P. Leitao. Chess robot system: A multi-disciplinary experience in automation. *Proceedings of the 9th Spanish-Portuguese Congress on Electrical Engineering*, 2005. 1

[8] I. M. Khater, A. S. Ghorab, and I. A. Aljarrah. Chessboard recognition system using signature, principle component analysis and color information. *Proceedings of the Second International Conference on Digital Information Processing and Communications*, pages 141–145, 2012. 1

[9] C. Matuszek, B. Mayton, R. Aimi, M. P. Deisenroth, L. Bo, R. Chu, M. Kung, L. LeGrand, J. R. Smith, and D. Fox. Gambit: A robust chess-playing robotic system. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4291–4297, 2011. 1

[10] M. Piskorec, N. Antulov-Fantulin, J. Curic, O. Dragoljevic, V. Ivanac, and L. Karlovic. Computer vision system for the chess game reconstruction. *Proceedings of the 34th International Convention*, pages 870–876, 2011. 1

[11] E. Sokic and M. Ahic-Dokic. Simple computer vision system for chess playing robot manipulator as a project-based learning example. *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology*, pages 75–79, 2008. 1

[12] D. Urting and Y. Berbers. Marineblue: A low-cost chess robot. *Proceedings of the International Conference Robotics and Applications*, pages 76–81, 2003. 1