A Case Study on Logging Visual Activities: Chess Game

Şükrü Ozan and Şevket Gümüştekin

Izmir Institute of Technology, Urla Izmir 35430 Turkey
sukruozan@iyte.edu.tr, sevketgumustekin@iyte.edu.tr
www.iyte.edu.tr

Abstract. Automatically recognizing and analyzing visual activities in complex environments is a challenging and open-ended problem. In this study this task is performed in a chess game scenario where the rules, actions and the environment are well defined. The purpose here is to detect and observe a FIDE (Fédération International des Ėchecs) compatible chess board, generating a log file of the moves made by human players. A series of basic image processing operations have been applied to perform the desired task. The first step of automatically detecting a chess board is followed by locating the positions of the pieces. After the initial setup is established every move made by a player is automatically detected and verified. Intel® Open Source Computer Vision Library (OpenCV) is used in the current software implementation.

1 Introduction

Interpreting visual activities is an open-ended research problem in Computer Vision. The recent availability of necessary computational power to process vast amount of video data motivated research on several aspects of video processing. The temporal component of video introduces an additional dimension compared to static images enabling us extraction of meaningful events. In [1] temporal segmentation techniques are reviewed in a general terms. Several interrelated applications such as content based image retrieval [2], video annotation [3], video indexing [4] has been a major focus of research. Most of these applications are based on low level processing of pixel data. Since there is an unlimited number of visual events that need to be recognized it is impossible to expect a general purpose algorithm to extract the meaning of all the actions in a video stream. The high level information in video data can be better interpreted if the application domain is limited. Several successful studies has been carried out in sports video applications (e.g. [5][6]).

In this paper, a board game: chess has been considered as the domain of study. Compared to the other board games chess is more interesting. Besides its popularity it introduces some challenge as a vision problem with its three dimensional pieces and relatively complex rules. Most of the work on chess has been in the area of Artificial Intelligence aiming to create computer chess players. The progress of these studies ended up as a computer [7] defeating a human world champion. Other related work involves reading printed chess moves [8]. The studies similar to the one in this paper also attack the problem of interpreting a chess game in action but they are focused on designing and implementing robots [9][10] that can move pieces in the chess board. Here, our main concern is understanding the actions on the chess board.

By observing a standard chess board, the time instants where moves are performed by players have been detected, the moves are recognized using the visual data as well as the rules of the game and a log file of the moves is generated as a result. The camera is expected to be located with a clear view of the complete board and all pieces, but there is no further constraint on the perspective of the scene.

2 Chessboard Detection

A chessboard has a unique and uniform structure which makes it easily identifiable even in complex scenes. The first step of our scheme involves detection of the chessboard and extracting information about its position and orientation. A built in tool in Intel® Open Source Computer Vision Library (OpenCV) [11] is used to detect the chessboard and identify 49 inner corners.

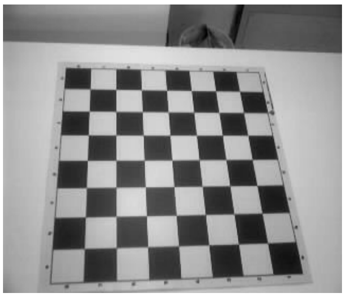


Fig. 1. Chess Board view example 1

The chessboard detection algorithm accepts grayscale images like the one in Figure 1. Initially, the algorithm applies a gray scale dilation operation [12]. By this operation touching corners of the squares are separated from each other. After the separation thresholding is applied to the image in order to convert the grayscale image to binary format. Generated binary image is used for a contour retrieving routine. While getting all the contour information, algorithm rejects contours with perimeters that are too small. So, contours with too small perimeters are regarded as noise in the image.

Contour analysis continues with searching for quadrangles. All the contours are checked and non-quadrangle contours are rejected. After these operations all the quadrangles and their corresponding contours in the source image are used for the next step which involves corner finding.

By using the retrieved contour plot as input, a corner finding algorithm is applied looking for sudden turns. The midpoint between the detected corners that are very close to each other are interpreted as the inner corners in the chessboard.

The 49 inner corners are verified and other corners are rejected by checking with the expected 7x7 uniform structure of the corner set.

3 Projection of the View

The only constraint on the position of the chessboard is that the angle between the surface normal of the board and the inverted viewing direction is sufficiently small (i.e. less than 30^o). Each piece is expected to be visible. But, some undesired perspective effects are allowed. Among such effects are: slight occlusion and coverage of more than one squares by a single piece. In this section we describe how the view of the chess board is rectified.

After the inner corners of the chessboard are detected and ordered as in Figure 2, the far corner points labeled as 1,7,43 and 49 are used for the perspective projection.

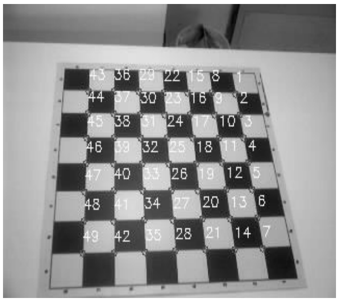


Fig. 2. Detected and labeled corners after chessboard detection algorithm applied

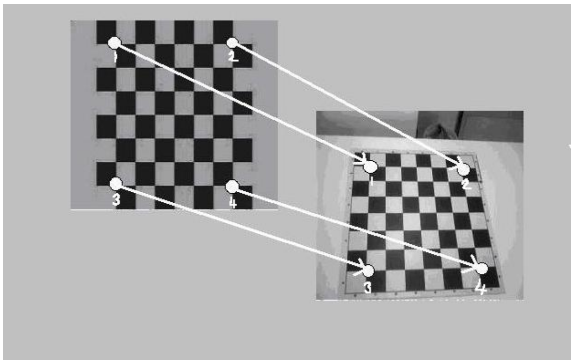


Fig. 3. 4 point correspondences for remapping

3.1 Finding the Parameters of the Perspective Transformation

A perspective projection can be formulized by the homographies [13]:

$$x_{im} = \frac{a * X + b * Y + c}{g * X + h * Y + 1}$$

$$y_{im} = \frac{d*X+e*Y+f}{g*X+h*Y+1} \tag{1}$$

Where in a backwards projection scenario (X,Y) are the coordinates on the desired projected view image and (x_{im},y_{im}) is the coordinates on the source image. Matching 4 points (8 coordinate parameters) in the target and source images allow us to find the unknown parameters of the perspective transformation.

8 linearly independent equations can be written in matrix representation as:

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -X_1*x_{im1} & -Y_1*x_{im1} \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -X_2*x_{im2} & -Y_1*x_{im2} \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -X_3*x_{im3} & -Y_1*x_{im3} \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -X_4*x_{im4} & -Y_1*x_{im4} \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -X_1*y_{im1} & -Y_1*y_{im1} \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -X_1*y_{im2} & -Y_1*y_{im2} \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -X_1*y_{im3} & -Y_1*y_{im3} \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -X_1*y_{im4} & -Y_1*y_{im4} \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x_{im1} \\ x_{im2} \\ x_{im3} \\ x_{im4} \\ y_{im1} \\ y_{im2} \\ y_{im3} \\ y_{im4} \end{bmatrix} \tag{2}$$

The 8x8 matrix can be represented as P for convenience, hence the unknown parameters can be computed as:

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = P^{-1} \cdot \begin{bmatrix} x_{im1} \\ x_{im2} \\ x_{im3} \\ x_{im4} \\ y_{im1} \\ y_{im2} \\ y_{im3} \\ y_{im4} \end{bmatrix} \tag{3}$$

After the chessboard is detected and the projected view is calculated, players put their pieces in their places, and the orientation of the board is analyzed. This analysis is done by checking the horizontal and vertical projection of pixels to see if the

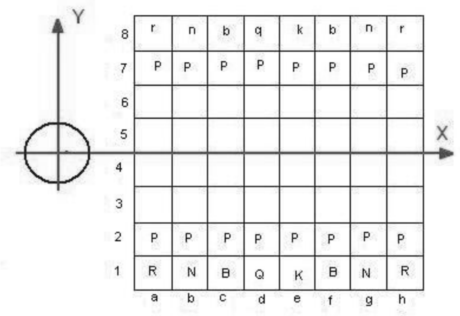


Fig. 4. Assumed chessboard standard, small letters are for black pieces, capital letters are for white pieces

horizontal projection shows an uneven distribution with a high number of black pixels towards the top of the board. If its orientation is not compatible with the standard given in Figure 4, then the projection is repeated as to show black pieces at the top and white pieces at the bottom in the projected view.

This is done by changing the orders of (x_{im1},y_{im1}), (x_{im2},y_{im2}), (x_{im3},y_{im3}) and (x_{im4},y_{im4}) in circular order, i.e. Instead of giving the reference points in the order:

$\{ (x_{im1},y_{im1}), (x_{im2},y_{im2}), (x_{im3},y_{im3}),(x_{im4},y_{im4}) \}$

One of these sets is used:

$\{ (x_{im2},y_{im2}), (x_{im3},y_{im3}), (x_{im4},y_{im4}),(x_{im1},y_{im1}) \}$
$\{ (x_{im3},y_{im3}), (x_{im4},y_{im4}), (x_{im1},y_{im1}),(x_{im2},y_{im2}) \}$
$\{ (x_{im4},y_{im4}), (x_{im1},y_{im1}), (x_{im2},y_{im2}),(x_{im3},y_{im3}) \}$

4 Move Detection

After the pieces are placed in order a typical scene looks like Figure 5.a. Figure 5.b is the projected view of the scene which is obtained by perspective transformation described earlier.

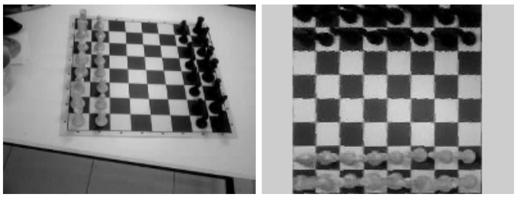

(a) (b)

Fig. 5. (a) A starting view (b) Projected view of the chess board

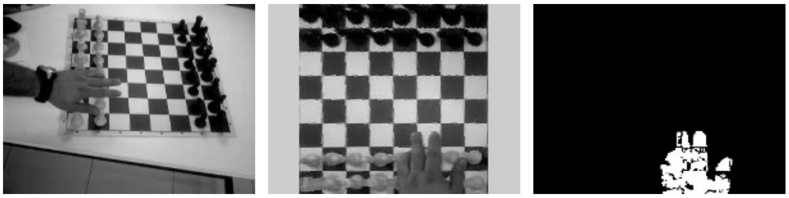

(a) (b) (c)

Fig. 6. (a)The hand in the scene (b)Projected view (c)Difference image

When the initial setup of pieces is complete the system enters in a “wait” mode for a play action. The next mode is the “alert” mode which is the state entered when a human activity is observed. The expected activity is the movement of a human hand on the chessboard. In Figure 6.a, player's hand is in the view of camera. Figure 6.b is the projected view. The difference between Figure 5.b and 6.b is shown in Figure 6.c.

A typical activity is detected from the video stream by observing the difference from the reference frame. The difference images are thresholded and number of foreground pixels are recorded. This operation forms a plot shown in Figure 7. The horizontal axis here is the time axis or frame number and the vertical axis is the number of the white pixels in difference images. This plot can easily be interpreted within the context of the chess move. An increase at the beginning of the graph indicates the beginning of action. A peak value suddenly drops to low levels when human hand stops to pick up a piece. Another peak is observed when it stops to put the piece on the board. When the initial level is reached we can assume that the move has ended.

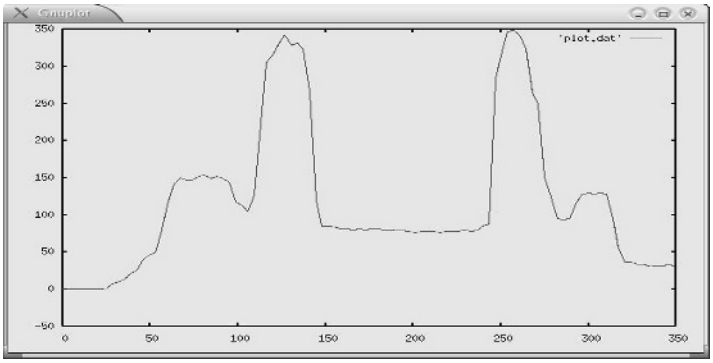
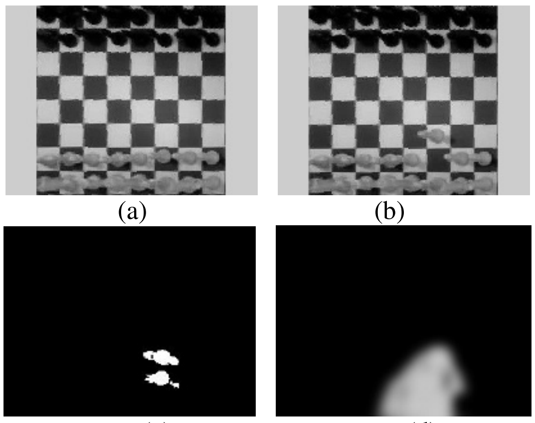


Fig. 7. A typical temporal plot of a move



(c) (d)

Fig. 8. (a) Before the move (b) After the move (c) Difference image (d) Cumulative difference

When the move is detected the system enters into the state of "analyze move". This step verifies whether or not a move is made and it determines the new placement of pieces on the board. This is done by using a cumulative difference image (Figure 8.d) which shows the position of the hand during the observed interval as well as the static frames before (Figure 8.a) and after the move (Figure 8.b). Figure 8. c is the difference between Figs (a) and (b).

5 Move Analysis

When a possible move is detected the next step is to verify the move and determine the new positions of pieces. The decision is based on the information about the static (e.g. in Figure 8.c) and dynamic (e.g. in Figure 8.d) changes in the scene. The ambiguity in these images is encoded by representing each square of the chessboard with a degree of compliance with the possible move. These degrees are computed by normalizing the brightness values corresponding to a square by maximum brightness value observed in the difference images since the beginning of the game (E.g. Figure 10.b & c).

The possible movements indicated by difference images are checked to see if they are valid by using the moves allowed by chess rules at the current state of the game. This is done by using a move table representation (E.g. Figure 9) for each piece which is updated after each move.

8	0	0	1	0	0	0	1	0
7	0	0	1	0	0	1	0	0
6	1	0	1	0	1	0	0	0
5	0	1	1	1	0	0	0	0
4	1	1	1	1	1	1	1	1
3	0	1	1	1	0	0	0	0
2	1	0	1	0	1	0	0	0
1	0	0	1	0	0	1	0	0
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**

Fig. 9. Move table of a queen located at c4 with no other piece blocking

Move tables are constructed according to the chess rules (E.g. pawns only move forward; the pieces can not jump with the exception of knights, etc.). After a move table is constructed it is stored in an array of 8 unsigned characters using C language. This is made by representing each line in the move table as decimal numbers. Making this conversion reduces the memory usage and simplifies computation. For example the table in figure 9 can be stored as:

[34 36 168 112 255 112 168 36 34]

Here every number stands for a raw of packed bits converted to decimal form.

This high level decision making mechanism involving knowledge representation of chess domain improves the reliability of the algorithm and it enables us to recognize special moves like en-passant or castling.

The algorithm comprises the following steps:

- Combine the information encoded in two types of difference images (e.g. in Figure 10.b and c). We use MINIMUM operator which stand for fuzzy AND [14].

- For each position with a value greater than a threshold (e.g. 0.5) combine the array resulting from previous step by the array representing the rules for the piece at that position.
- Compare the pairs of possible moves and accept the one with highest degree of likelihood.

In Figure 10 (a)Original view of a sample game can be seen. Figures 10.b and 10.c shows the movement of white queen in d1 to d3 respectively.

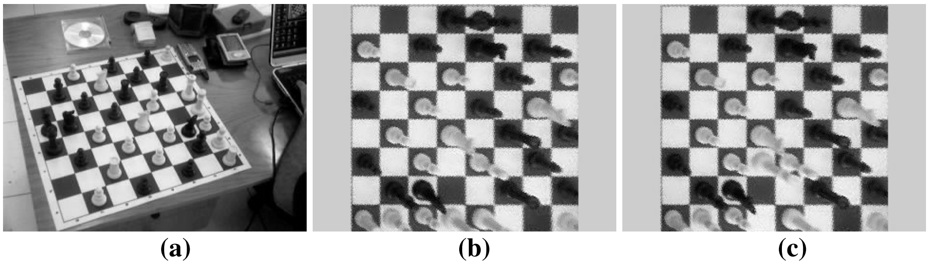


Fig. 10. (a) original view of a sample game (b) reference image taken before the move (c) reference image taken after the move

Figure 11 shows the difference image between 10.b and 10.c. To extract position of a piece before and after a move occurrence, bottom of the piece gives accurate information. In the difference images (E.g. 11.a), bottom of a piece can be distinguished as a circular shape. By using this morphology piece positions can be approximated. By using a circular structuring element (shown in figure 11.b), correlation for each square can be calculated. This structuring element has 30x30 pixels in size.

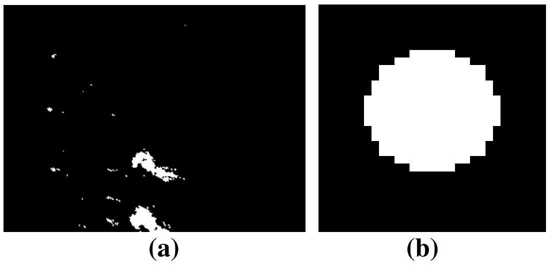


Fig. 11. (a) Image of the difference between 10.b 10.c (b) morphological element

Due to the perspective of the camera, in the difference image pieces like Queen and King generates some difference in neighbor squares. This situation may cause some faulty results. In order to eliminate these results, a little modification has been added to the correlation mechanism. Simple correlation requires a convolution of difference image with the structuring element for each square on the chessboard. The modification reduces correlation of a square if a black pixel in the element coincides with a white pixel (a difference pixel). By this way the faulty characteristic due to the

8	0	0	0	0	0.0021	0	0	0
7	0.0231	0	0	0	0	0	0	0
6	0	0.0042	0	0	0	0	0	0
5	0.0378	0.0021	0.0084	0	0	0	0	0
4	0	0	0	0	0	0	0	0
3	0.0462	0	0.0252	0.721	0.275	0	0	0
2	0.0021	0.0042	0.042	0.0084	0.174	0.0063	0	0
1	0	0.00196	0.00313	1	0.197	0	0	0
	a	b	c	d	e	f	g	h

Fig. 12. (a) correlation table of the difference image in 11.a

perspective is eliminated. In figure 12, the correlation table, which has been calculated by using the difference image 11.b and structuring element 11.b, can be seen.

By using the result obtained from the table in Figure 12 in parallel with move table information and the cumulative difference information, the move can be predicted accurately.

6 Conclusion

A computer vision system which identifies a chessboard, and interprets the actions on the board is described. The domain knowledge is represented and updated at each state of the game. The visual data is combined with the encoded domain knowledge in order to recognize actions. This study aims to be a first step into task oriented understanding of actions in video data. The well defined environment and the rules of the chess game provide us a good framework in order to use domain knowledge in a vision guided decision making process. The experience gained here can be extended to more complicated vision tasks involving more complex environments. Besides attacking more complicated problems, the future work involves making the algorithm more robust in terms of extraction of reliable data and decision making.

References

1. I. Koprinska, S. Carrato, "*Temporal video segmentation: A survey*"; Signal Processing: Image Communication, Volume 16, Issue 5, January 2001, Pages 477-500
2. A. W. M. Smoulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "*Content based image retrieval at the end of the early years*" IEEE Trans. Pattern Anal. Machine Intell., vol. 22, pp. 1349–1380, Dec. 2000.
3. A. Dorado, J. Calic, E. Izquierdo, "*A rule-based video annotation system*"; IEEE Transactions on Circuits and Systems for Video Technology, Volume 14, Issue 5, May 2004 Page(s): 622 – 633
4. R. Brunelli, O. Mich and C. M. Modena, "*A Survey on the Automatic Indexing of Video Data*", Journal of Visual Communication and Image Representation, Volume 10, Issue 2, June 1999, Pages 78-112

5. J. Assfalg, M. Bertini, C. Colombo, A. D. Bimbo, W. Nunziati, "*Semantic annotation of soccer videos: automatic highlights identification*". Computer Vision and Image Understanding, Volume 92, Issues 2-3, November-December 2003, Pages 285-305
6. A. Ekin, A.M. Tekalp, R. Mehrotra, "*Automatic soccer video analysis and summarization*"; IEEE Transactions on Image Processing, Volume 12, Issue 7, July 2003 Page(s):796 - 807
7. M. Campbell, A. J. Hoane, and F.H. Hsu, "*Deep Blue*", Artificial Intelligence, Volume 134, Issues 1-2, January 2002, Pages 57-83
8. H.S. Baird, K. Thompson, "*Reading chess*", IEEE Trans. Pattern Anal. Machine Intell., Volume 12, Issue 6, June 1990 Page(s):552 - 559
9. F.C.A. Groen, G.A. Den Boer, A. Van Inge, R. Stam, "*A chess-playing robot: lab course in robot sensor integration*", IEEE Transactions on Instrumentation and Measurement, Volume 41, Issue 6, Dec. 1992 Page(s):911 - 914
10. E. Uyar, Ş. Gümüştekin, Ş. Ozan, L. Çetin, "*A Computer Controlled Vision Oriented Robot Manipulator for Chess Game*", Proc. of Workshop on Research and Education in Control and Signal Processing, REDISCOVER 2004, Cavtat, Croatia
11. http://www.intel.com/research/mrl/research/opencv/
12. R.C. Gonzales, R.E. Woods, "Digital Image Processing", Addison Wesley, 1993
13. E. Trucco, A. Verri, "*Introductory Techniques for 3D Computer Vision*", Prentice Hall, 1998.
14. J.Z. Zimmermann, "Fuzzy Set Theory and its Applications", Kluwer Academic Publishers, 1993.