Geometry-Based Populated Chessboard Recognition

Conference Paper · November 2017

DOI: 10.1117/12.2310081

CITATION
1

READS
91

3 authors:

Youye Xie
Colorado School of Mines
9 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Gongguo Tang
Colorado School of Mines
54 PUBLICATIONS **1,662** CITATIONS
SEE PROFILE

William A Hoff
Colorado School of Mines
77 PUBLICATIONS **2,366** CITATIONS
SEE PROFILE

Geometry-Based Populated Chessboard Recognition

Youye Xie[1], Gongguo Tang[1], William Hoff[2,3]
[1]Department of Electrical Engineering, Colorado School of Mines, Golden, Colorado USA
[2]Department of Computer Science, Colorado School of Mines, Golden, Colorado USA
[3]DAQRI Holographics, Ltd, Vienna, Austria

ABSTRACT

Chessboards are commonly used to calibrate cameras, and many robust methods have been developed to recognize the unoccupied boards. However, when the chessboard is populated with chess pieces, such as during an actual game, the problem of recognizing the board is much harder. Challenges include occlusion caused by the chess pieces, the presence of outlier lines and low viewing angles of the chessboard. In this paper, we present a novel approach to address the above challenges and recognize the chessboard. The Canny edge detector and Hough transform are used to capture all possible lines in the scene. The k-means clustering and a k-nearest-neighbors inspired algorithm are applied to cluster and reject the outlier lines based on their Euclidean distances to the nearest neighbors in a scaled Hough transform space. Finally, based on prior knowledge of the chessboard structure, a geometric constraint is used to find the correspondences between image lines and the lines on the chessboard through the homography transformation. The proposed algorithm works for a wide range of the operating angles and achieves high accuracy in experiments.

Keywords: Chessboard recognition, geometric transformation, Hough transform, K-means.

1. INTRODUCTION

1.1 Background

Chess is a popular intellectual and entertaining game all over the world. Chessboards are also commonly used in computer vision for tasks such as camera calibration, due to the regular pattern of the lines on the board. By viewing a chessboard from several camera poses, the line features in different images are used to estimate the extrinsic and intrinsic parameters of the camera [1,2]. Those parameters include camera poses and the distortion coefficients. In addition, three dimensional scene reconstruction [3] and lens distortion correction [4] both rely on the accurate estimation of camera parameters. Many chessboard recognition methods for camera calibration have been developed.

However, when the chessboard is populated with chess pieces, such as during an actual game, the problem of recognizing the board is much harder. Challenges include occlusion caused by the chess pieces on the board, the presence of outlier lines in the scene and low viewing angles of the chessboard. Recognizing the board is a necessary first step for an automated chess playing system [5,6], such as a chess playing robot or an augmented reality chess assistant. This paper focuses on chessboard recognition under game conditions using a geometric constraint-based approach. Our approach permits flexible operating angles and allows different occlusion conditions.

1.2 Previous Work

There are several approaches to recognize a chessboard. One straightforward method is using a magnetic board [7] or embedding sensors in the chessboard. A sensor detector can then recognize the chessboard based on the received signal. However, a modified chessboard is very expensive and needs to be maintained regularly. Alternatively, computer vision methods can be used to recognize the chessboard using a single chessboard image. This is more portable and transferable to different types of chessboards. The MarineBlue chess robot [6] recognizes the chessboard based on hue, saturation and brightness (HSB) information and compares them with the true chessboard squares to find the board. Another approach that uses pixel information can be found in [8].

Another class of methods [2,9,10], recognizes the chessboard by using corner detection to determine the board lines. Those methods achieve promising performance when the camera is directly above the chessboard or there is no occlusion. But these constraints are undesirable in practice. Alternatively, the Hough transform based line recognition method allows higher angles and occlusion flexibility since the local occlusion is unlikely to cover the whole lines. One of such approaches is proposed by Tam et al. [11]. In their approach, edge recognition is implemented first to find all

possible edge points in the image and the Radon transform is used to determine the line locations. However, false edge intersections may hurt its ability to find the true diagonal lines of the board.

The paper is organized as follows. In section 2, we present our geometry-based populated chessboard recognition approach. In section 3, we show experimental results on several boards from different viewing angles under different occlusion conditions. Finally, we conclude this paper in section 4.

2. GEOMETRY-BASED CHESSBOARD RECOGNITION

2.1 Overview of the Approach

This section provides an overview of the geometry-based chessboard recognition process as summarized in Fig. 1.

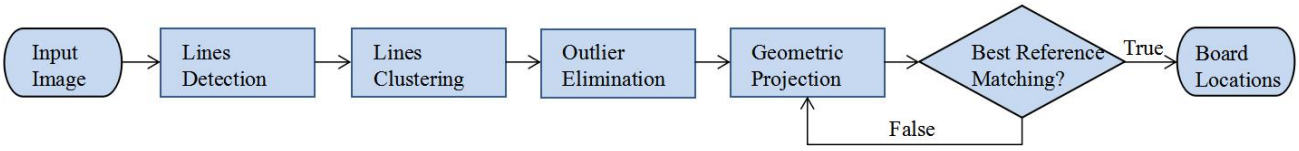


Figure 1. The overview of the geometric-based chessboard recognition approach.

Given a chessboard input image, the Canny edge detector and the Hough transform are used to find all possible lines. These lines are then clustered into two groups based on their locations in the Hough transform space. The two groups represent the two sets of lines on a chessboard (horizontal and vertical). Lines that do not appear to belong to either group are filtered out in the outlier elimination step, and the intersections of two groups of remaining lines will be calculated and recorded. Finally, possible chessboard candidates will be projected and matched to a chessboard reference model. The solution with largest number of matching corners and the smallest matching residual error is chosen as the final result. The details of each step will be discussed in the following subsections.

2.2 Line Detection

Our approach takes gray scale images as its inputs. We assume that the chessboard is completely contained within the image and occupies most of the image. The Canny edge detector is applied to find possible chessboard edge points. The threshold of the detector is adaptively chosen to obtain a number of edge points that is approximately 0.7% of the input image's size. After getting the edge image in Fig. 2 (b), the Hough transform is used to detect lines. Each edge point votes for the lines that pass through it, to create a Hough parameter array (H-space), where each bin stores the number of edges that lie on the line corresponding to that bin. The H-space is indexed by θ and ρ, which are the parameters of a line using the equation $\rho = x\cos(\theta) + y\sin(\theta)$. The width of the H-space is 180, corresponding to values of θ from -90 to 89 degrees. The height of the H-space is equal to 2d, where d is the length of the image diagonal. A simple threshold and non-maximum suppression are implemented in the H-space to filter out weak lines and remove lines that are too close to stronger lines. Finally, the remaining lines with large counts are recorded and enter the next step for clustering. The size of the H-space depends on the image resolution. In the next section, we will show how to handle images with different sizes.

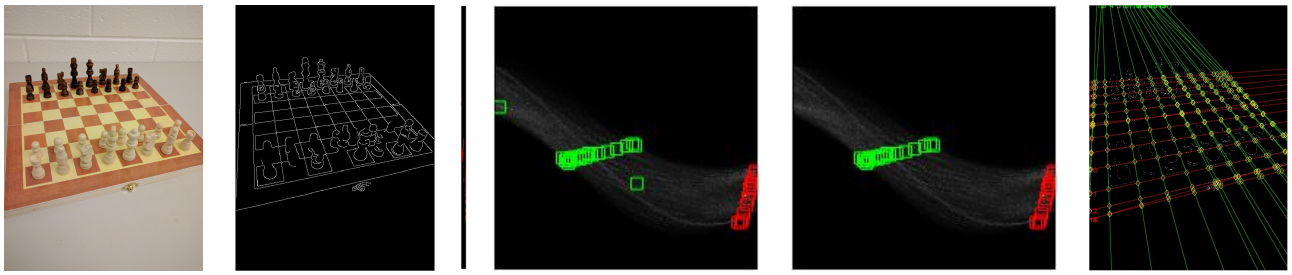
2.3 Lines Clustering & Outlier Elimination

Both line clustering and outlier elimination are performed in the H-space. However, since the height of the H-space (corresponds to the ρ parameter) varies depending on the input image size, the clustering result may be affected by the image resolution. In order to avoid that, Tam et al. [11], cluster the lines based on only the θ values which is sensitive to the background noise. To pursue a more stable clustering performance, both the ρ and θ values are taken into account in our approach. The k-means algorithm with a special scaling distance metric is customized to cluster the lines in the H-space. With this metric, adjacent lines which belong to the same group have a small distance between them as compared to the distance to a line from the other group.

Specifically, the scaling distance is equivalent to the Euclidean distance in a scaled H-space so that the distance between two adjacent lines in the ρ direction and the θ direction are weighted equally. We looked at the largest $\Delta\theta$ and the largest $\Delta\rho$ between two adjacent lines under typical viewing conditions, such that the viewing angle from the vertical direction is 45 degrees. Empirically we found that if we scale the H-space so that it is square, then these distances are

roughly equal. To visualize the scaling process, the original H-space and the scaled H-space are shown in Fig. 2 (c) and (d) respectively.

Edge points on the chess pieces and edges from the background may induce spurious chessboard lines. In order to eliminate the outlier lines and reduce the search complexity in the following steps, we eliminate any line that is far from the other lines in its own group, using a k-nearest-neighbor (KNN) inspired algorithm. This is done as follows. For each group in H-space, we find the distance between each point and its nearest neighbor. Points whose distances to their nearest neighbors are much larger than the mean value are marked as outliers and eliminated. As an example, two false lines were eliminated as shown in Fig. 2 (e). At the end of this step, all possible intersection points between the two groups of lines are calculated and are marked in Fig. 2 (f) using yellow diamonds.

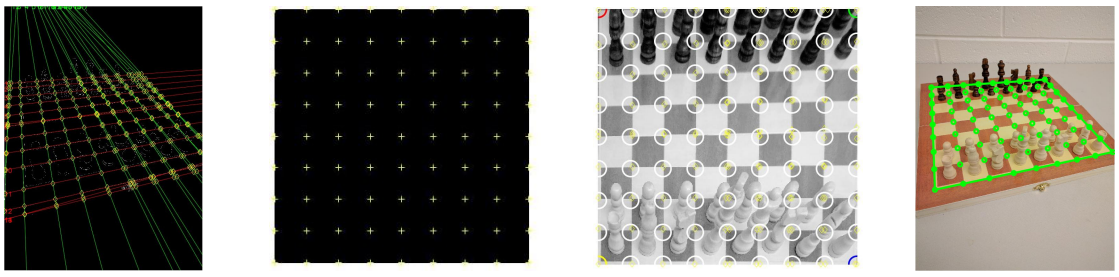


(a) (b) (c) (d) (e) (f)

Figure 2. Lines clustering and outlier elimination process. (a) The input image. (b) The edge image (4032x3024 pixels). (c) The raw H-space (10079x180 pixels). (d) The scaled H-space (180x180 pixels), with clustering results marked by red and green rectangles, corresponding to the two clusters. (e) The outlier elimination result shown in the scaled H-space. (f) The remaining lines are projected onto the image and intersections between the two groups of lines are marked by yellow diamonds.

2.4 Geometric Projection & Reference Matching

The next step is to find the correspondence between the image lines and the lines on the chessboard. This is done using a geometric projection constraint. Our model for the chessboard consists of 9 horizontal lines and 9 vertical lines. There are 81 intersection points, as shown in Fig. 3 (b). We search for a set of four points from the intersections found in the previous section, which correspond to the outer corner points of the model. To find the correct set, we compute the homography transformation that maps the four image points to the outer four points in the model. Then we transform all remaining points to the model, using the same transformation. Matching points (those with low residual error) are kept as inliers. An example of inlier points projected onto the model is shown in Fig. 3 (c), and the same points are shown on the original image in Fig. 3 (d). The transformation with largest number of inlier points and smallest residual error is used as the estimated transformation.



(a) (b) (c) (d)

Figure 3. The geometric projection and reference matching process. (a) The detected intersection points. (b) True intersection points from the model. (c) Projection of inlier intersection points onto the model (the input image has also been projected onto the model). (d) The detected chessboard intersection points on the input image. The outer boundary is marked using green lines.

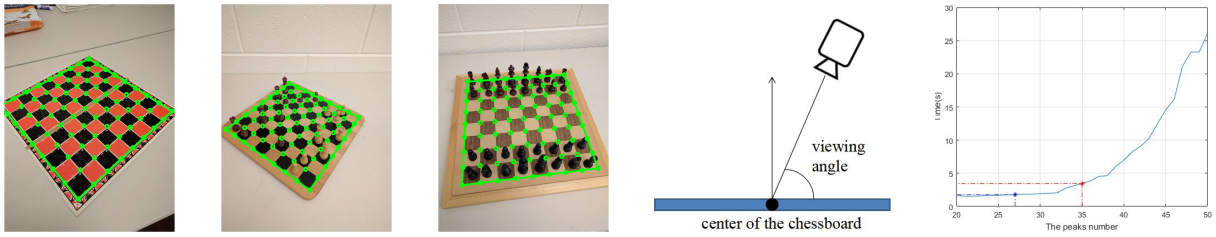
To search for the set of four detected points that correspond to the board corner points, we simply try all possible combinations of detected points, starting with the outermost points and proceeding towards the image center. The true points should be among the outermost points. We cap the number of combinations to be tested.

3. EXPERIMENTS

3.1 Experimental Setup

We test our approach with different kinds of boards as shown in Fig. 3 (d) and Fig. 4 (a)-(c) under different occlusion conditions and viewing angles. Specifically, three different occlusion conditions are considered. First, there are no pieces on the board as shown in Fig. 4 (a). Second, only some of the pieces are on the board and their locations are determined by a chess contest middle game as shown in Fig. 4 (b). Third, all the pieces are on their initial positions as shown in Fig. 4 (c). Moreover, the test images are divided into several classes based on the viewing angles from 10 to 90 degrees. The viewing angle is determined by the angle between the line of sight from the camera to the board center and the board's normal vector as shown in Fig. 4 (d).

The algorithm was implemented in MATLAB on an i7-4710HQ CPU. The processing time is mainly determined by the number of detected lines to be searched (i.e., the lines found in section 2.2). With a relative clean background as shown in Fig. 4 (a)-(c), our approach achieves good performance when the number of remaining lines is 27, and in this case, the average processing time is 2 seconds, as marked by a blue star in Fig. 4 (e). A larger number of remaining lines means more resistance to false edges and noisy background. However, it requires a longer processing time. The number of remaining lines should be set to a number such that all the true board lines remain. In our final implementation, we keep the top 35 strongest lines, and the processing time is around 4 seconds. This is marked in Fig. 4 (e) by a red star.

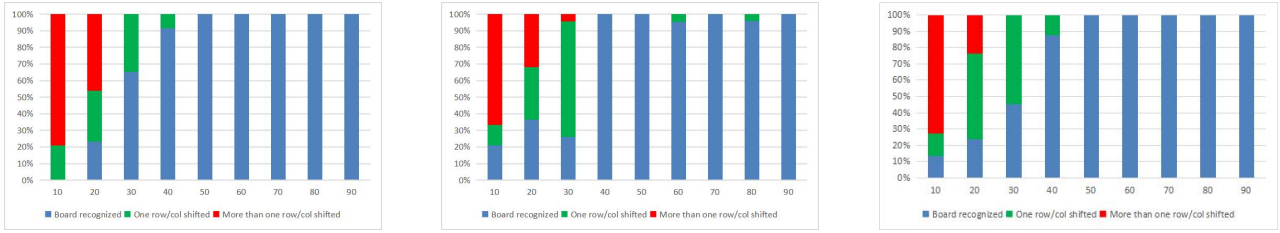


(a) (b) (c) (d) (e)

Figure 4. (a)-(c) Different test board examples with recognition results. (d) The viewing angle. (e) The relation between remaining lines and the processing time.

3.2 Results and Analysis

In order to provide a more detailed result, the output is classified into three cases: success, one row or column shifted and failure. The case of "one row or column shifted" occurs when there is a background line that is parallel to the board lines, and spaced from the outermost board line by about the same distance as the distance between the valid board lines. For every fixed occlusion condition and viewing angle, 20 to 30 board images taken at different viewpoints are tested and we summarize the recognition success rate in Fig. 5 (a)-(c). To further limit the variables, the board type in Fig. 4 (b) is used and we would expect similar result for different kind of boards based on our experiments with mixed boards.



(a) Board without pieces. (b) Middle game (c) Board with pieces on their initial positions.

Figure 5. The recognition success rate.

From the figures above we can see that our approach achieves almost 100% success rate when the viewing angle is equal or greater than 40 degrees. When the viewing angle is relatively small such as 30 degrees, we still get good results, although there are many cases of "one row or column shifted". More importantly, our workable viewing angle range matches the angles that a person would naturally view the chessboard during a game. If the viewing angle is less than 30

degrees, many of the pieces will be occluded by other pieces on the board and it would be difficult to recognize all the pieces with only one viewpoint.

Now we want to point out two common reasons that lead to failure. The first reason is a low angle. From Fig. 5, our approach fails when the viewing angle is 10 degrees. When the image is taken from an extremely low angle, the chessboard lines in the far end are too close to each other and some of them are suppressed in the edge recognition step. Another major reason for failure is occlusion. If the viewing angle is below 30 degrees, with two rows of pieces in front of the chessboard boundary, the boundary edges can barely be detected and the edges on the chess pieces induce many false lines. Moreover, since we keep a fixed number of strongest edge lines in the line recognition step, if the background contains strong spurious edges, we might miss some of the true chessboard edges. Fortunately, this failure can be overcome by making the chessboard occupy the most part of the image or simply by increasing the threshold for remaining lines.

4. CONCLUSION

In this paper, a geometry and line-detection based algorithm is developed for populated chessboard recognition. The Canny edge detector and Hough transform are used to detect the possible chessboard lines in the input image. The clustering and outlier elimination steps are performed in the H-space to exclude false lines and reduce the geometric transformation searching complexity. Finally, a geometric projection constraint is applied to find the correct correspondence between image lines and chessboard lines. Our approach achieves high accuracy within a specific viewing angle range and is time efficient which only takes around 4 seconds for the whole process with a high resolution (4032x3042 pixels) image.

REFERENCES

[1] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on pattern analysis and machine intelligence* 22(11), pp. 1330 – 1334, 2000.

[2] A. De la Escalera and J. M. Armingol, “Automatic chessboard recognition for intrinsic and extrinsic camera parameter calibration,” *Sensors* 10(3), pp. 2027 – 2044, 2010. 6

[3] A. Geiger, J. Ziegler, and C. Stiller, “Stereoscan: Dense 3d reconstruction in real-time,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 963 – 968, Ieee, 2011.

[4] H. S. Sawhney and R. Kumar, “True multi-image alignment and its application to mosaicing and lens distortion correction,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(3), pp. 235 – 243, 1999.

[5] C. Matuszek, B. Mayton, R. Aimi, M. P. Deisenroth, L. Bo, R. Chu, M. Kung, L. LeGrand, J. R. Smith, and D. Fox, “Gambit: An autonomous chess-playing robotic system,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference* on, pp. 4291 – 4297, IEEE, 2011.

[6] D. Urting and Y. Berbers, “Marineblue: A low-cost chess robot.,” in *Robotics and Applications*, pp. 76-81, Citeseer, 2003.

[7] L. Miolo, “Magnetic chessboard with self-centering pieces,” Nov. 10 1981. US Patent 4,299,389.

[8] J. E. Neufeld and T. S. Hall, “Probabilistic location of a populated chessboard using computer vision,” in *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on*, pp. 616-619, IEEE, 2010.

[9] N. Banerjee, D. Saha, A. Singh, and G. Sanyal, “A simple autonomous robotic manipulator for playing chess against any opponent in real time,” in *Proceedings of the International Conference on Computa- tional Vision and Robotics*, 2011.

[10] A. Geiger, F. Moosmann,Ö. Car, and B. Schuster, “Automatic camera and range sensor calibration using a single shot,” in *Robotics and Automation (ICRA)*, *2012 IEEE International Conference on*, pp. 3936 – 3943, IEEE, 2012.

[11] K. Y. Tam, J. A. Lay, and D. Levy, “Automatic grid segmentation of populated chessboard taken at a lower angle view,” in *Computing: Techniques and Applications, 2008. DICTA ’ 08. Digital Image*, pp. 294 – 299, IEEE, 2008.