ChessVision: Chess Board and Piece Recognition

Jialin Ding
Stanford University
jding09@stanford.edu

Abstract

This paper details a method to take an image of a chess board and output a reconstructed computer representation of the board through board and piece recognition. Though techniques for board recognition have been thoroughly explored in the past, especially in relation to chessboard calibration, previous works on piece recognition often focus on non-robust segmentation procedures that rely heavily on the exact color of customized chess boards and pieces. The method presented in this paper improves upon the segmentation-based approaches of previous work in piece recognition by introducing a novel approach using classifiers trained on feature descriptors, which is more robust to the similarities in color seen in real-life chess boards. This work is important for both automating the recording of moves in human chess games and improving the ability of chess-playing AI that depend on vision.

1. Introduction

The recording of moves during a chess game is often a tedious manual task that impedes the flow of the game, especially in formats such as speed chess where manually recording moves hinders efficient use of time. At the professional level, specialized chess sets have been developed to record moves automatically. However, this equipment is expensive and not easily accessible to recreational players. Further, in an age when much analysis and storage of chess games is done on computers, chess players at all levels can benefit from the ability to input games into a computer-readable format by simply taking a picture of a real-life board, as opposed to manual input. More recently, chess-playing AI have relied on either specialized equipment or human assistance in order to function. To truly create an autonomous chess-playing robot would require the ability to detect a chessboard and pieces through vision.

This paper outlines an approach that takes as input an image which contains a chessboard, and outputs a computer representation of that board. The approach was created with the best use case for recreational chess players in mind. Such players most likely do not have access to multiple cameras. Therefore an approach that takes advantage of multiple images of the same scene was ruled out. In additional, a stable overhead view of the board, though optimal for board recognition, is difficult to configure in real settings, and so was also ruled out. Therefore, this paper was written under the assumption that users would take one picture, mostly likely from a smartphone camera, from an angle to one side of the board, in order to minimize obstruction from the players themselves.

One constraint faced in this approach was the lack of existing databases of classified chess board and piece images. Therefore, the database used in this paper was manually constructed from one particular chess set and therefore does not offer robustness to intra-class variation among chessboards and pieces. In order to fit the best use case, the particular chess set used in this paper was chosen to be representative of the type of sets used by recreational chess players.

The remainder of this paper will first outline previous work on chess recognition and highlight a novel improvement to a sub-component of the established algorithm; provide a technical explanation of the approach; discuss the experimental setup and results; and present conclusions.

2. Previous work

The realm of chess recognition can be broadly separated into two major areas. Board recognition refers to the detection of the chess board within the image and the identification of board characteristics, such as the orientation, the location of squares, etc. This usually involves finding a projective transformation that rectifies the image into a format where the location of the board is known. Piece recognition refers to the detection of pieces on the board and the localization and classification of those pieces. Since piece recognition relies on knowledge about the board, board recognition is typically a prerequisite step to piece recognition.

2.1. Board Recognition

Most of the previous computer vision work related to chess has focused on the area of board recognition. Gen-

eral techniques for board recognition can be separated into corner-based approaches and line-based approaches.

Corner-based approaches use corner detection on the input image to identify the corners of the chess board, then either perform Hough transforms to identify the lines of the board or assign coordinates to the corners directly. These approaches either assume a plain background, so as to reduce the number of corner-generating artifacts; use a top-down overhead view [5]; or require the absence of pieces from the board, in order to prevent the occlusion of corners [1]. These approaches work well for general game-tracking applications, where the board can be identified before the start of the game, when there are no pieces on the board. However, corner detection fail under occlusion and background noise, which occur in real-life settings.

Line-based approaches use edge detection on the input image to identify lines of the chess board. Domain knowledge, such as the fact that the board can be identified by 18 total lines and the orientations of half those lines will be orthogonal to the other half, makes line-based approaches more robust to noise, and is therefore the more popular technique [2, 6, 7].

2.2. Piece Recognition

Not as much work has gone into piece recognition. Game-tracking applications assume the starting positions of the pieces, and there can use differences in intensity values after each move to track the movement of pieces [1, 5]. Techniques that do not assume a starting position focus on color segmentation to detect pieces and then use shape descriptors to identify them [2, 3, 4]. However, color segmentation often relies heavily on the ability to distinguish the four major colors on a chessboard – the colors of white squares, black squares, white pieces, and black pieces. Often, squares and pieces of the same color are difficult to distinguish, so unreasonable constraints must be placed, such as the usage of a red-and-green chessboard [2] or a side-view that relies on depth but occludes most of the pieces [3]. A more general solution should be able to perform piece recognition on a standard, unmodified set.

The main purpose of this paper will be to offer a novel approach to piece recognition that improves upon the state-of-the-art techniques outlined above. Instead of the color segmentation-based approach, which is often non-robust and places unnecessary constraints on the board, this paper uses a machine learning approach based on training classifiers with gradient-based feature descriptors. The next section will explain the entirety of this approach in detail.

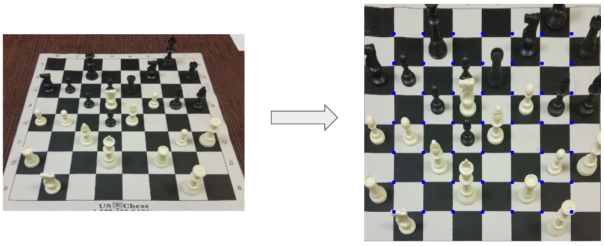

Figure 1. In the board recognition stage, a projective transformation matrix is constructed that maps the the input image (left) to a rectified image (right). The blue dots on the rectified image indicate the predicted square corners.

3. Technical Solution

3.1. Board Recognition

The original ChessVision program was implemented using the line-based methods of de la Escalera *et al.* [6]. However, this technique produced a sizable error rate due to noise and frequently was unable to correctly detect the chessboard. Given that the main purpose of this paper was to introduce a novel approach to piece recognition, the decision was made to simplify board recognition by adding a minimal amount of user interaction. This eliminates error in board recognition and allows all experimentation to focus on the accuracy of piece recognition, without having to account for the accuracy of board recognition.

The implementation of board recognition proceeds as follows. The user is presented the image and must manually select the four corners of the board, in a specified order (clockwise from the top left). The 2D coordinates of these corners are then used to calculate a projective transformation that maps the board in the image to a 640 pixel by 640 pixel rectified board (Figure 1).

Calculation of the projective transformation matrix is conducted as follows. Let P_i and $(u, v, 1)_i^T$ be the corresponding corner points in homogeneous coordinates on the input image and the rectified image, respectively, for $i \in \{1, 2, 3, 4\}$. Then the projective transformation matrix $H \in \mathbb{R}^{3x3}$ can be found up to a scale factor by solving the system of equations $Ph = 0$, where

$$P := \begin{bmatrix} P_1^T & 0 & -u_1 P_1^T \\ 0 & P_1^T & -v_1 P_1^T \\ \vdots & \vdots & \vdots \\ P_4^T & 0 & -u_4 P_4^T \\ 0 & P_4^T & -v_4 P_4^T \end{bmatrix}$$

and h is H reshaped into a column vector. The unknown h can then be found by taking the first column vector of the third returned matrix of applying SVD to P.

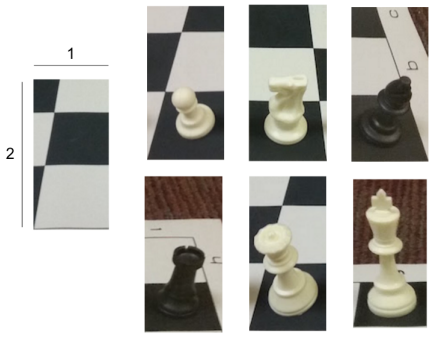

Figure 2. Examples of database images for each class. Note the 1:2 aspect ratio.

Class	Size of training set (images)
Empty	64
Pawn	128
Knight	32
Bishop	32
Rook	32
Queen	32
King	16

Table 1. Size of training set by class.

3.2. Piece Recognition

Constructing the database There does not currently exist an extensive database of labeled chess pieces. For the purposes of this paper, a database of chess pieces was manually constructed from one specific chess set. The database consisted of images of individual pieces placed on the board (Figure 2). All images were taken with a 1:2 aspect ratio, in order to guarantee that the entirety of the pieces were captured.

Images in the database were manually labelled with one of seven classes, corresponding to the different chess pieces – these were pawn, knight, bishop, rook, queen, king, and empty. Images labelled empty did not contain a piece, but rather an empty square. White and black pieces of the same type were given the same label in the database. Distinguishing pieces by color occurred at a later stage. The distribution of images in the database is shown in Table 1.

Feature extraction Features were constructed over the images in the database using two separate descriptors – scale-invariant feature transform (SIFT) [9] and histogram of oriented gradients (HOG) [10]. The relative performance of using one over the other is examined in experimentation.

In order to ensure the resulting features were of the same size, all images in the database were resized to 64 pixels by 128 pixels.

Features were extracted with SIFT using a bag-of-words model, as follows. SIFT keypoints were first extracted from all database images. K-means was run over these keypoints to extract n centroids, which we refer to as *words*. In experimentation, n was varied. Then the database was iterated through again. For each image, SIFT keypoints were extracted and mapped to the nearest word. A histogram was then constructed over these mapped keypoints and normalized to sum to 1. These histograms served as the feature for their respective image.

Features were extracted with HOG directly on each resized image. Block size, block stride, cell size, and the number of bins were varied in experimentation.

The above process of extracting features with SIFT and HOG was repeated five times, once at one of five possible aspect ratios – 1:1, 1:1.25, 1:1.5, 1:1.75, and 1:2. During each iteration, database images were cropped from the top to fit the given aspect ratio. (For example, for the 1:1.5 aspect ratio, only the bottom $3/4$ of each image, 64 pixels by 96 pixels, was used to compute features. The top $1/4$ was discarded.) Note that this means no cropped was necessary for the iteration with aspect ratio 1:2.

In the end, computing features using both SIFT and HOG over five aspect ratios for all database images produced $2 \cdot 5 \cdot 336 = 3360$ feature descriptors. The remainder of piece recognition can use either the SIFT features or the HOG features; the process is identical for both. Therefore, we will proceed under the assumption that HOG feature descriptors are used. In experimentation, the performance of both SIFT and HOG was explored.

Training SVMs were used to train binary one-vs-rest classifiers for each class. The positive training set consisted of features of that class, and the negative training set consisted of all other features. Each classifier was trained only on the feature descriptors of the aspect ratio corresponding to its class (Table 2). Therefore seven classifiers were produced. Given the unbalanced size of the data set for each class, weights were given to each class c:

$$w_1 = \frac{\sum_{i \neq c} \#\text{ image for class } i}{\sum_i \#\text{ image for class } i}$$

$$w_0 = \frac{\#\text{ image for class } c}{\sum_i \#\text{ image for class } i}$$

Piece classification Pieces were detected and classified using a modified sliding windows technique. We take advantage of our domain knowledge of the chessboard as well as the projective transformation between the 640 pixel by 640 pixel rectified image produced from board recognition and the input image to slide exactly over the 64 squares of the chessboard.

Class	Aspect ratio
Empty	1:1
Pawn	1:1
Knight	1:1.25
Bishop	1:1.5
Rook	1:1.25
Queen	1:1.75
King	1:2

Table 2. Aspect ratios of training set and bounding boxes by class

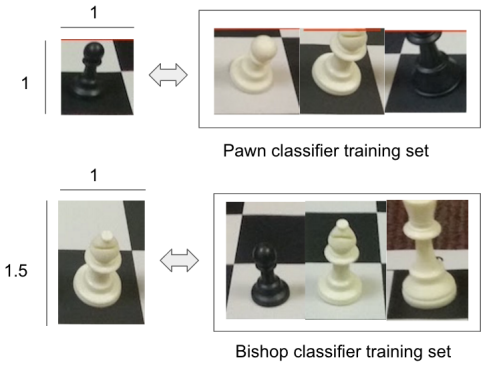

Figure 3. The pawn classifier used bounding boxes with a 1:1 aspect ratio, while the bishop classifier used bounding boxes with a 1:1.5 aspect ratio. The difference in resulting bounding box images is shown through sample images.

At each square, each of the seven classifiers is run over a window with aspect ratio corresponding to the class (Figure 3). This was done in order to incorporate information about the shape of the piece into classification, as well as to reduce the possibility of artifacts in the bounding box (i.e., pieces in adjoining squares). The width of the window is set to the width of the base of the square, and the height is adjusted accordingly. (For example, a bishop classifier that is run on a square with a base width of 100 pixels in the input image would have a window with height 150 pixels.) The window is resized to a width of 64 pixels before being run through the classifier.

Instead of outputting a prediction, the classifiers output a class membership probability estimate. Therefore each classifier produces a probability matrix over the 64 squares, with each square being assigned a probability of being occupied by the piece of the classifier (Figure 4). Overall, the seven classifiers output seven probability matrices. For each square, we take the class with the highest probability estimate in that square as our classification prediction.

Piece color Since the rectified image is 640 pixels by 640 pixels, each square is 80 pixels by 80 pixels. Further, using domain knowledge we know the color of each square. Piece color is determined by taking the central 40 pixels by 40 pixels of a square, binarizing, taking the ratio of white pixels to black pixels, and comparing to the expected ratio of an empty square. (For example, if the empty square is known to be white, then a sizable black to white pixel ratio indicates that the piece is black.)

4. Experiments and results

Board recognition and determination of the color of each piece each had near perfect accuracy, and so were not extensively tested in experimentation.

Piece recognition was tested on 30 images of chessboards with a variable number of pieces placed in randomized configurations. Three metrics were measured on each image. To formally define these metrics, let $B^p \in \mathbb{R}^{8x8}$ be the board predicted by the program, and let $B^{gt} \in \mathbb{R}^{8x8}$ be the ground truth board, where each element of B^p and B^{gt} contains the class label of the corresponding square. Labels for all classes are positive, except for the label for the empty class, which is negative. Let P^c be the probability matrix produced by the classifier for class c, so that P^c_{ij} is the probability estimate of the square in row i and column j of the chessboard having class c.

- **Detection Accuracy**: The fraction of squares correctly identified as empty or non-empty.

$$DA = \frac{\sum_{i,j=1}^{8} 1\{sign(B^p_{ij}) == sign(B^{gt}_{ij})\}}{64}$$

- **Classification Accuracy**: The fraction of correctly classified squares.

$$CA = \frac{\sum_{i,j=1}^{8} 1\{B^p_{ij} == B^{gt}_{ij}\}}{64}$$

- **Cross Entropy of the Chessboard**: An extension of the idea of cross entropy to the chess setting. It is intended as a more descriptive form of classification accuracy, since it takes into account the margin of probability scores.

$$CE = \frac{\sum_{i,j=1}^{8} -\ln P^c_{ij}}{64}$$

The optimal parameters for SIFT and HOG feature extraction were found by varying the parameters and measuring the quality of the resulting predictions on the 30 test images. The optimal number of bins for SIFT was found to be 10. Any further increase in the number of bins did not appreciably increase accuracy, but did increase computation time. For HOG, the block size, block stride, cell size, and number of bins were varied, with the constraints that block stride and cell size were set equal. The results are shown in Table 3. Smaller cell sizes produced better results, up

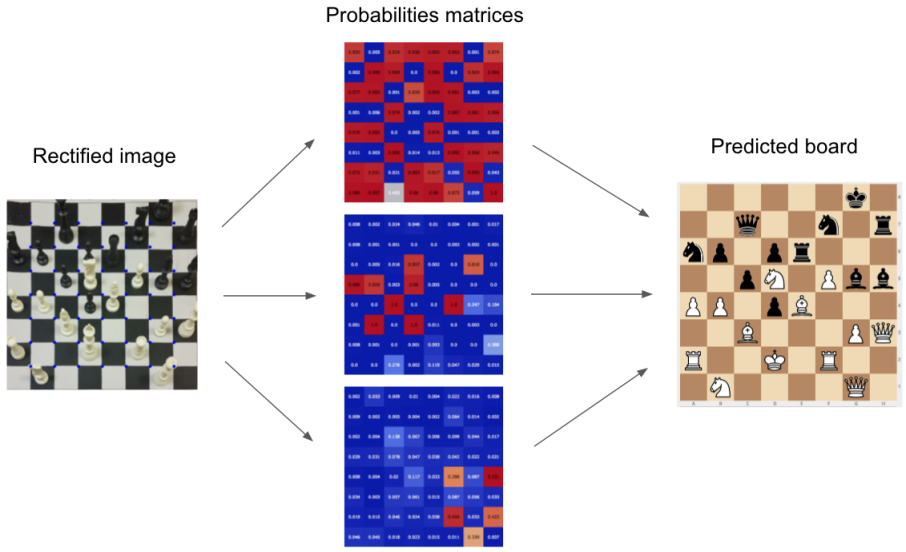

Figure 4. Sliding windows produces seven probability matrices (only three are shown here). Red squares in these matrices denote high probability of a piece. Combining the results produces the final predicted board.

to a point, likely because greater granularity better captures the relevant differentiating features of pieces that typically occupy small portions of the board (e.g., the crosses on top of kings and the crenellations on rooks). Also, increasing the number of bins produced better results, up to a point. The optimal configuration of parameters would determined to be a block size of 32 pixels, a block stride of 16 pixels, a cell cize of 16 pixels, and 9 bins. Optimizing further did not produce appreciably better results, but did increase computation time. The class weights for SVM also had to be adjusted slightly to account for the small size of the data set.

Results of running the program with the optimized parameters are shown in figure 5. The confusion matrix over all 30 images using HOG features is shown in table 4. There are several interesting things to note in these results.

- HOG performs better than SIFT. There are several explanations for this. First, the different classes of pieces are similar in shape, so a collection of keypoints that describe one class may be very similar to the keypoints that describe another class. Instead, pieces are better described by a holistic description. Further, HOG is fundamentally well-suited to detection problems, which is essential what each classifier is doing. Due to its holistic nature, HOG is also more robust to occlusions.

- Accuracy decreases and cross entropy increases as the number of pieces on the board increases. This is likely due to a combination of reasons. First, differentiating an empty square from a non-empty square is fundamentally simpler than differentiating two pieces. Second, the rate of occlusions of pieces by other pieces increases as more pieces appear on the board. Due to the inadequate data set, the trained classifiers are not robust to occlusions.

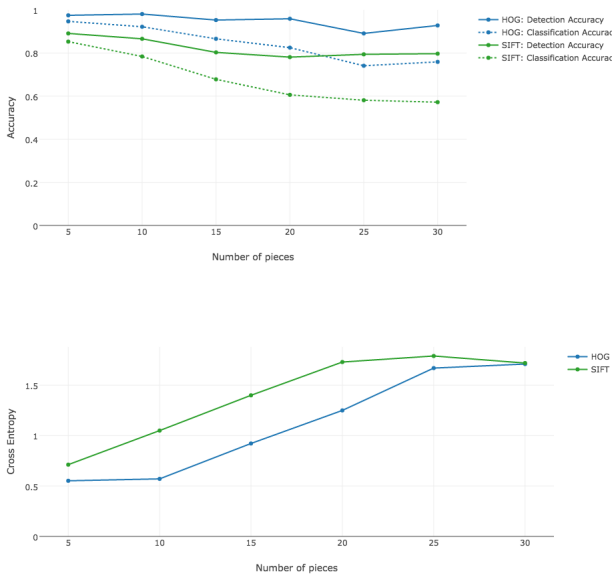

Figure 5. Detection and classification accuracy for SIFT and HOG on top. Cross entropy for SIFT and HOG on bottom.

- Some pairs of pieces are difficult to differentiate. For

Config. No.	HOG parameters				Measured performance		
	Block size	Block stride	Cell size	Num. of bins	Cross Entropy	Detection Acc.	Classification Acc.
1	16	8	8	9	1.18	0.931	0.803
2	32	16	16	9	1.34	0.884	0.768
3	32	4	4	9	1.19	0.925	0.809
4	**16**	**4**	**4**	**9**	**1.04**	**0.950**	**0.846**
5	16	2	2	9	1.03	0.949	0.843
6	8	4	4	9	1.06	0.948	0.840
7	4	2	2	9	1.03	0.943	0.836
8	16	8	8	2	1.38	0.868	0.748
9	16	8	8	20	1.08	0.939	0.843

Table 3. Parameter configurations of HOG and quality of resulting predictions. Configuration 4 was determined to be optimal.

		Predicted Class						
		Empty	**Pawn**	**Knight**	**Bishop**	**Rook**	**Queen**	**King**
Actual Class	**Empty**	1211	115	46	4	12	6	1
	Pawn	19	204	6	0	3	4	0
	Knight	14	22	25	2	2	6	1
	Bishop	6	33	6	23	2	1	0
	Rook	13	15	9	7	19	6	0
	Queen	4	5	5	4	0	9	3
	King	5	8	8	7	0	5	15

Table 4. Confusion matrix over all test images.

example, bishops were more often predicted as pawns than bishops. This is due to the similar nature of the two pieces, and is something that could be addressed by a more comprehensive data set.

5. Conclusions

The source code for this project can be found at `https://github.com/jialinding/ChessVision`. This paper presented a proof of concept of using a machine learning approach to piece recognition that is much more robust to real-life chess boards than segmentation-based approaches of past works. We determined that training classifiers on HOG feature descriptors produced test results with a detection accuracy of 95% and a classification accuracy of 85%. Key details to enhance results included using different aspect ratios in order to better utilize the shape of the piece in classification and reduce artifacts; and using a cell size for HOG that is small enough to capture the distinguishing attributes of pieces.

However, improvements must be made to the program before it can be used in real game scenarios. The main bottleneck to improvement, as was apparent throughout the process of implementing this paper, was the lack of a comprehensive labelled data set of images of chess boards and pieces. With a more comprehensive data set, the classifiers would likely be more accurate and more robust to occlusions and intra-class variations, and classification techniques that require large amounts of training data, such as neural networks, could be explored. These areas should be the focus of future research.

References

[1] J. Hack and P. Ramakrishnan. (2014). *CVChess: Computer Vision Chess Analytics* [Online]. Available: `http://web.stanford.edu/class/cs231a/prev_projects/chess.pdf`

[2] C. Danner and M. Kafafy. (2015). *Visual Chess Recognition* [Online]. Available: `https://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Danner_Kafafy.pdf`

[3] N. Loewke. (2015). *Depth-based image segmentation* [Online]. Available: `http://stanford.edu/class/ee367/Winter2015/report_loewke.pdf`

[4] I. A. Aljarrah, A. S. Ghorab, and I. M. Khater, "Object Recognition System using Template Matching Based Signature and Principal Component Analysis", *International Journal of Digital Information and Wireless Communications* vol. 2, no. 2, pp. 156-163. The Society of Digital Information and Wireless Communications, 2012

[5] C. Koray and E. Sumer, "A Computer Vision System for Chess Game Tracking," presented at the 21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia, 2016

[6] A. De la Escalera and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," *Sensors*, vol. 10, no. 3, pp. 20272044, 2010.

[7] K. Y. Tam, J. A. Lay, and D. Levy, "Automatic grid segmentation of populated chessboard taken at a lower angle view,"

in *Digital Image Computing: Techniques and Applications (DICTA), 2008*. IEEE, 2008, pp. 294299.

[8] J. E. Neufeld and T. S. Hall, "Probabilistic location of a populated chessboard using computer vision," *Midwest Symposium on Circuits and Systems*, pp. 616619, 2010.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91-110, Nov. 2004

[10] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005