

Computer Vision for Chess Position Identification

William Chapin and Toby Frick

CSC-262

Abstract

The ability of chess computers today greatly exceeds the ability of even the greatest grandmaster, but to get a computer's analysis of a position, a user typically needs to manually enter the location of each piece on the board into the computer. The goal of this project is to automate this process by generating the position on a chess board from only an image. To accomplish this, our algorithm finds the outermost four corners of the board and uses them to generate a homography estimate for the board, allowing us to find the location of each square of board in the image. Extracting this region, the algorithm then classifies the image region as one of thirteen possibilities (an empty square, one of six white pieces or one of six black pieces). The algorithm produced strong results, identifying on average 72% of the squares correctly on the images resembling the training data and 61% of the squares correctly for images we took of a slightly different board.

1. Introduction

The goal of this project is to be able to discern the locations of all pieces on a chessboard with only a photo. We do this by first leveraging the geometry of the image and the fixed structure of a chess board to create a homography estimate. With this, we are able to find and extract the image region for each square in the image. Using a pretrained CNN, we then identify the piece in the extracted region and thus identify the position of the entire board. The final output is an 8x8 categorical array containing our classification for each square.

The first step for the algorithm is feature matching to identify the outermost four corners of the board, which we will use to build a homography estimate. Since a chessboard is a repeating checkerboard pattern, locating specifically the outermost corners posed a challenge. Our feature matching method runs four filters over the entire image, one filter for each corner, eliminates all but the strongest responses, and then chooses the response closest to that corner of the image (i.e. closest to the bottom left corner of the image when detecting the bottom left corner of the board). Unfortunately, this feature matching technique imposed the precondition that input images must be taken from a particular angle. This challenge is described at greater length below.

Using the four located points, the algorithm produces a homography estimate, allowing us to find the location in the image of anywhere on the chessboard. An image region is then extracted for each of the 64 squares on the board. Finally, the image slice is identified with a Convolutional Neural Network (CNN), to determine what piece, if any, is on the extracted square. Which square the extracted image represents is already known when it is inputted into the CNN, so the output can be immediately added to the appropriate spot in an 8x8 array. As shown in Table 1, index (1,1) in the array contains the classification for a8, and (8,8) contains the classification for h1 in the final output.

| Square Classified at Each Index in the Output Array | | | | | | | |
|---|----|----|----|----|----|----|----|
| a8 | b8 | c8 | d8 | e8 | f8 | g8 | h8 |
| a7 | b7 | c7 | d7 | e7 | f7 | g7 | h7 |
| a6 | b6 | c6 | d6 | e6 | f6 | g6 | h6 |
| a5 | b5 | c5 | d5 | e5 | f5 | g5 | h5 |
| a4 | b4 | c4 | d4 | e4 | f4 | g4 | h5 |
| a3 | b3 | c3 | d3 | e3 | f3 | g3 | h3 |
| a2 | b2 | c2 | d2 | e2 | f2 | g2 | h2 |
| a1 | b1 | c1 | d1 | e1 | f1 | g1 | h1 |

Table 1.

2. Process

2.1 Generate Homography Matrix

We start by generating a homography matrix which encodes a linear transformation which can, given a 'top down' view of the chess board, convert points in the 'top down' view of the chess board to corresponding points on an image of a chess board. The 'top down' view of the chess board was synthetically generated by us and is a 130x130 element matrix of doubles in which white squares are represented by a 16x16 region of 1 values, black squares are represented by a 16x16 region of 0 values and a 1-pixel border of .5 values. We call this the mental board. The mental board is displayed below alongside an example image of a chess board we will try to identify the position of.

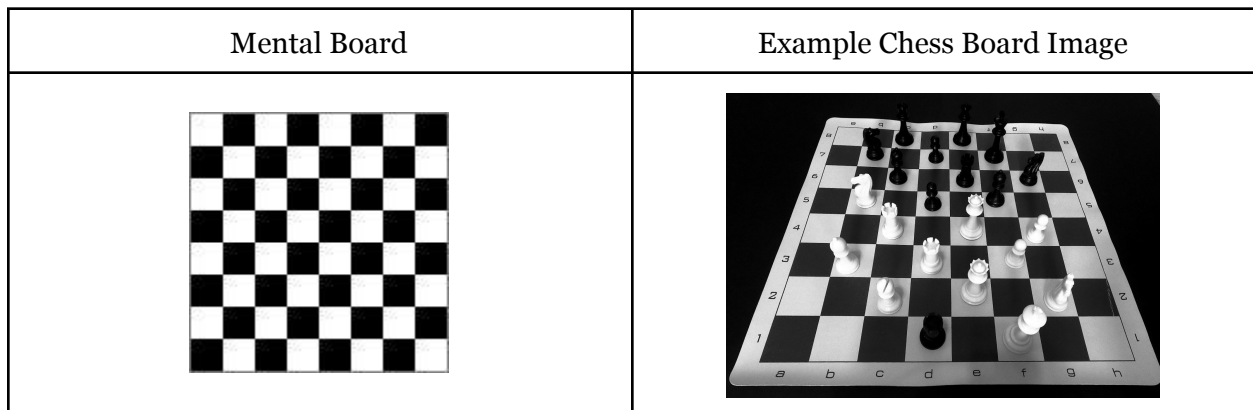


Fig 1.

The math behind generating the Homography matrix is explained in David Kriegman's Homography Estimation (2007). This generation method requires at least four corresponding points between the two images which we initially provided using hand-matched points. These hand matched points can also be found with an automated feature matching system.

2.2 Automated Feature Matching

In order to generate the Homography matrix automatically (without hand-matched points), we found that we needed to find a consistent way to find corresponding points between our mental board and the chess board image. Our solution to this problem was correlation. To generate the four matching points we need for the Homography calculation, we decided to try to identify the four corners of the chess board. We created four correlation filters that represented the four corners of the chess board. Then, for each filter, we performed correlation to find regions of the board with sufficiently high correlation. This generated matches across many regions on the board. Then, depending on the corner we were trying to find, we choose the match that is closest to that corner. For example, if we are trying to find the top left corner of the board, we pass our top-left corner filter over the board with correlation, and choose the top-leftmost match. With testing, we found that this method produces sufficient results as measured visually by the location of the four matching points (see Fig 3). Our handmade correlation filters are displayed below.

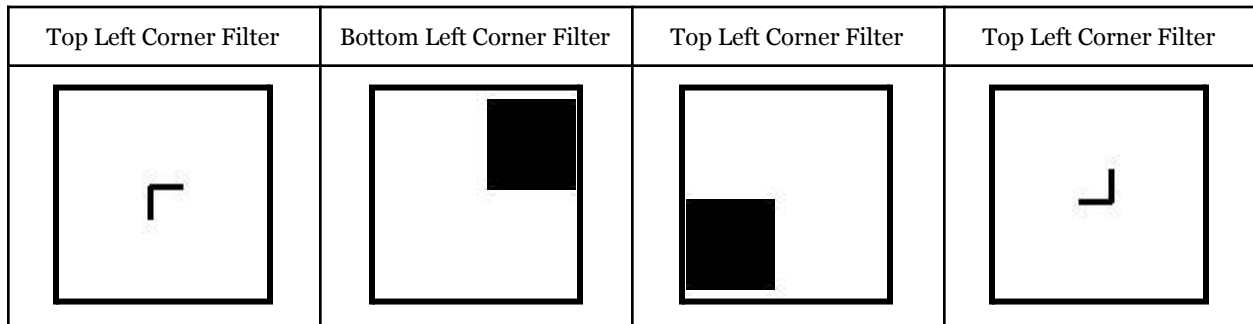


Fig 2.

Several issues arise with this method that require us to make a few assumptions about the image. The first of which is that the image of the board is taken with the camera behind the white side of the board (row 1). This is so that we can be assured that the top left and bottom right corners will be white squares and that the top right and bottom left corners are black squares. The second is that the orientation of the photo (i.e. the height of the photo off the ground plane and angle of the image plane relative to the ground plane) must be such that all four corners are visible but not transformed so much as to not work with the correlation filters. Additionally, this requires that the rows of the chess board are horizontal across the image. We must also ensure that the surface above which the board is placed is dark and that the board mat is not very wrinkled as this detracts from the planarity that we assume with image homography. If the surface that the board is placed on is not sufficiently dark, adjustments must be made to

the getHomography function to improve corner detection accuracy. Below is displayed the corners of the chess board we were able to identify using our filters above.

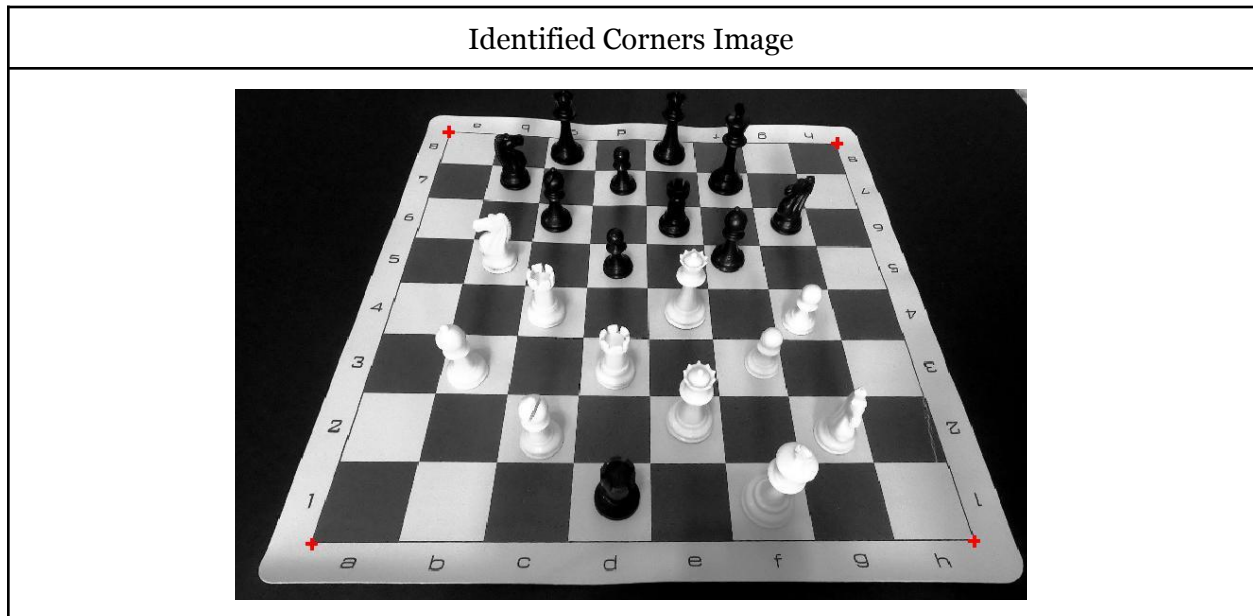


Fig 3.

With these assumptions and our corner filters, we were able to automate the homography generation process. The homography matrix produced is very noisy, but is close enough for our purposes to extract and identify each board square. Given a small sample of transformed points, we found our homography estimate has a RMSE of only 25 pixels, which for reference is less than 2.5% of the height of the image. Below is displayed four points in the mental board and their estimated corresponding location in a chess board image using the Homography Matrix.

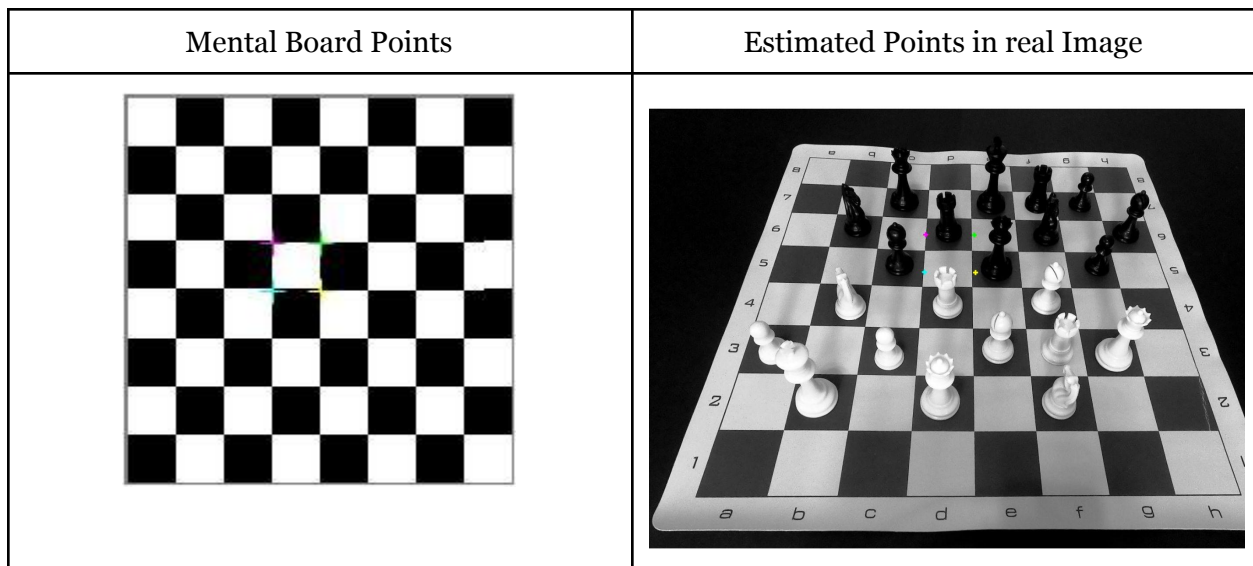
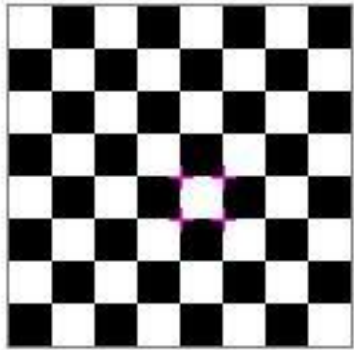
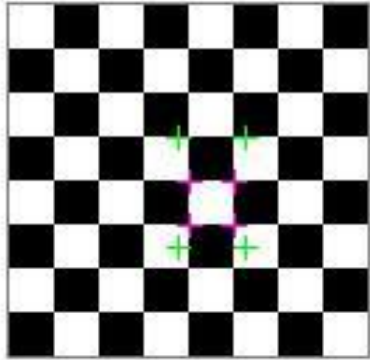


Fig 4.

2.3 Extract Board Squares

To extract an image of each square, we create a vector of the location of every corner in our mental image. We know the locations of these corners since this is the artificial board we created. For each square, we first take its four corners. In order to extract the region around the square and thus any piece on the square, we enlarge this area. Notice, pieces only occlude squares on the board immediately above or next to the square the piece is on in all the photos. Since we know how many squares a piece may block in the photo, we adjust our four corners so that they include these additional squares. We adjust the corners so it covers a quarter square in the horizontal directions, and the entire square above and half below (See Fig. 5 Upper Right). Accounting for the homography estimate error, this is ultimately more like a square and a half above and no squares below.

Sending these four points through the homography transformation yields four points enclosing the square as well as any piece on it (See Fig 5. Bottom Left). These points are cleaned to make sure they are in the bounds of the image, and then the rectangular region determined by these points is extracted from the image. An advantage of this method is that we will already know the location on the board of any piece we identify in the image. Examining the results of this method, we found that we successfully extracted an image of each square including any piece on it for all 128 squares in two separate input images. In other words, this method has a 100% accuracy in our evaluation.

| Mental Board Points Around e4 | Mental Board Expanded Region Containing e4 |
|---|--|
|  |  |



| Transformed Corners of the Expanded Region Containing e4 | Extracted Image for Square e4 |
|---|--|
|  |  |

Fig 5.

*Images are cropped for formatting.

2.4 Classify Extracted Board Squares

After all board squares have been successfully extracted, we can train a convolutional neural network to classify these extracted squares by piece type. There are 13 kinds of squares total (6 black pieces, 6 white pieces and empty squares), therefore, our neural network will be trained with 13 classification labels. The dataset we are using is relatively small and features under 3,000 total images. Therefore, to obtain optimal results with a limited dataset, we employed transfer learning to adapt the pretrained CNN SqueezeNet for our classification purposes. In order to do so, we replace the final convolutional layer with an untrained one featuring 13 1x1 filters and the final classification layer with an untrained one. We then slowed training on all of the remaining pretrained layers so that the vast majority of weight adjustment takes place in the two untrained layers.

3. Key Challenges

3.1 Automating Point Matching

We quickly discovered that hand-matching points was not an effective way to generate image Homography and an automatic point-matching system would be required. The difficulty in an automated point matching system is that by nature of the design of a chess board, there are not many unique features as the board is a repeated pattern. We tried a few techniques such as MatLab's detectCheckerboardPoints function, but found that because some of board square corners are occluded by pieces, we are not able to create an accurate method to generate matching points using this function. Our eventual solution arose from noticing that the four corners of the chessboard are unique filters and correlation could be used to find those corners

accurately across different images of chessBoards with slightly different setups within the bounds of the assumptions outlined above.

3.2 CNN Training

Initially, we wanted to train our own CNN, but quickly found that our dataset was not large enough to generate accurate results which encouraged us to research methods to train a network on a smaller dataset. This led us to our eventual solution of employing transfer learning on a pretrained network which leveraged the ability of a pretrained network to generate rich descriptions of images while also allowing our specific classifications (the original network had 1000 possible image classifications).

4. Experimental Setup

In order to evaluate the efficacy of our project, we first applied our algorithm to 7 images taken by the same person that gathered the dataset we used to train our algorithm. These 7 images are of a chess board and pieces that are exactly the same as those used to generate the dataset. We also use a number of our own images of a chess board that is similar to - but is also somewhat visually distinct from - the one used in the dataset.

The factors we wanted to analyze are the effect of image capture angle on accuracy, true positive result accuracy, false positive result accuracy, piece-wise identification accuracy and row-by-row accuracy which are all calculated and displayed below. We wanted to see the piece-wise identification accuracy to see if our system detected different pieces with different accuracies to determine what pieces would benefit from a larger training dataset. We also wanted to look at the row-by-row accuracy to see the effect that scale had on the accuracy of the system. However, all of our ground truths had to be generated by hand, so the number of results we are able to evaluate is limited.

5. Evaluation

5.1 Results from Provided Images

Our first evaluation is on the testing images provided by the dataset. These images are the ones we held in mind throughout the development of our algorithm and therefore should yield the highest success rates. These images also featured a constant image capture angle, so we are unable to measure the effect of a changing camera angle for these images.

In applying our algorithm, we found that it detected corners accurately for 6 of the 7 images. The following evaluation is conducted only on the 6 images with accurately detected corners.

Overall True Positive Rate: 72.77%

Provided Image Results by Piece

| Piece | True Positive Rate | False Positive Rate |
|--------------|--------------------|---------------------|
| Black King | 42.86% | 76.92% |
| Black Queen | 35.71% | 37.5% |
| Black Rook | 78.57% | 50% |
| Black Bishop | 85.71% | 58.62% |
| Black Knight | 85.71% | 47.83% |
| Black Pawn | 0% | Never Detected |
| Empty Square | 79.59% | 8.59% |
| White King | 85.71% | 62.5% |
| White Queen | 42.86% | 57.14% |
| White Rook | 64.29% | 18.18% |
| White Bishop | 85.71% | 50% |
| White Knight | 71.43% | 23.08% |
| White Pawn | 42.86% | 68.42% |

Table 2.

Provided Image Results by Row

| Row | True Positive Rate |
|--------------------------|--------------------|
| 1 (Furthest From Camera) | 66.07% |
| 2 | 53.57% |
| 3 | 60.71% |
| 4 | 71.43% |
| 5 | 66.07% |
| 6 | 75% |
| 7 | 89.29% |
| 8 (Closest To Camera) | 100% |

Table 3.

We find that in general, we classify more accurately squares which appear closer to the camera with a perfect success rate on the closest row. This is likely due to a lack of occlusion along this row. Each other row is subject to occlusion from pieces located on the row below it which often led to inaccurate classifications. We also note that there were no chess pieces placed on the closest row, so every true classification was empty square.

5.2 Results From Our Own Images

Overall True Positive Rate: 60.63%

Our Own Image Results by Piece

| Piece | True Positive Rate | False Positive Rate |
|--------------|--------------------|---------------------|
| Black King | 60% | 84.21% |
| Black Queen | 0% | Never Detected |
| Black Rook | 40% | 0% |
| Black Bishop | 0% | 100% |
| Black Knight | 80% | 46.67% |
| Black Pawn | 20% | 42.86% |
| Empty Square | 75.31% | 0% |
| White King | 60% | 50% |
| White Queen | 100% | 68.75% |
| White Rook | 10% | 0% |
| White Bishop | 10% | 95.45% |
| White Knight | 44.44% | 33.33% |
| White Pawn | 87.5% | 46.97% |

Table 4.

Our Own Picture Results by Row

| Row | True Positive Rate |
|--------------------------|--------------------|
| 1 (Furthest From Camera) | 45% |
| 2 | 17.5% |
| 3 | 65% |
| 4 | 85% |
| 5 | 77.5% |
| 6 | 70% |
| 7 | 85% |
| 8 (Closest To Camera) | 40% |

Table 5.

Our second evaluation is on images we obtained ourselves. These images show overall worse results than the test images from the dataset. This is likely because the chess set we were able to obtain featured pieces that were slightly different than those used to train the CNN. As a result, there are some pieces that were very difficult for the CNN to classify correctly. Empty squares, however, were detected at similar rates to the given board images.

5.3 Results from different camera angles

Finally, we took two images of the starting board position from two different camera angle: one from an angle similar to our given images and one from a more top-down perspective. These images are displayed below each with their overall true positive rate.

Results from Different View Angles



| | |
|---|--|
|  |  |
| True Positive Rate: 70.31% | True Positive Rate: 79.69% |

Fig 6.

As these images are taken of the same board on the same surface in the same position, we can conclude that it is likely that images taken from a more top-down angle produce more accurate results.

6. Conclusion

We have shown that using image homography and a CNN trained through transfer learning and a relatively small dataset, we can accurately classify most board squares in images of chess boards. The two major bottlenecks for this algorithm are our corner detection algorithm which limits the variety of input images that our algorithm can process successfully and the relatively small volume of data on which the CNN was trained. We believe strongly that results will improve if a larger dataset is used to train our CNN that features different looking pieces which will allow for more accurate image classification.

References

General:

Matlab Deep Learning Toolbox

<https://www.mathworks.com/products/deep-learning.html>

Andrew Underwood's Board Game Image Recognition using Neural Networks (2020)

<https://towardsdatascience.com/board-game-image-recognition-using-neural-networks-116fc876dafa>

Matlab Transfer Learning Tutorial

<https://www.mathworks.com/help/deeplearning/gs/get-started-with-transfer-learning.html>

SqueezeNet

<https://arxiv.org/abs/1602.07360>

Dataset:

Andrew Underwood's Chess Dataset

<https://www.dropbox.com/sh/8s4tvir5zbotseq/AACAQypmuFb6j-Yww9x9Q6Gta?dl=0>

Academic Papers (APA):

Kriegman, D. (2007). Homography estimation. *Lecture computer vision I, CSE a*, 252.