Chess Board State Detection and Applications through Artificial Intelligence and Machine Learning, and Convolutional Neural Networks

Aditya Verma

Abstract

This paper addresses the enhancement of automated chess recognition systems, focusing on overcoming challenges such as poor lighting, diverse camera angles, and varying chess set designs. Through the use of cutting-edge convolutional neural networks (CNNs) trained on diverse datasets, this study employs advanced image processing techniques including edge, corner, and line detection algorithms, along with the RANSAC algorithm for robust corner identification. Pretrained CNN models are used for classifying board occupancy and piece types. The methods significantly improve system accuracy under controlled conditions, demonstrating high success rates in LED lighting on homogeneous surfaces. However, performance is still affected by extreme lighting variations, unconventional chess sets, non-standard camera angles, and glare reflected by the chessboard. The study presents promising advancements, highlighting the potential for further improvements to enhance universal applicability and robustness in real-world scenarios.

Keywords: Chess, Artificial Intelligence, Machine Learning, Image Processing, Neural Networks, Computer Science, Recognition Systems, Software, Coding

Introduction

By revolutionizing an ancient game, chess recognition systems have emerged as a remarkable application of modern technology. Since its inception in the 6th century, chess has captivated players worldwide, demanding quick thinking and strategic moves (Chess History, 2021). Traditionally, players recorded their moves for later analysis, a process often burdened by the manual entry of data into computers or notebooks. However, the invention of chess recognition systems has brought newfound convenience and accessibility to this old game.

Despite the existence of chess recognition systems, their accuracy has been a subject of consideration, especially regarding challenging conditions such as poor lighting, camera quality, and various camera angles. To overcome these obstacles, cutting-edge convolutional neural networks (CNNs) are now employed, and trained on diverse datasets comprising of chessboards captured from unique camera angles and varying conditions. As a result, these systems can promptly display an 8 by 8 grid on the screen, representing the board and its pieces in real-time, increasing user convenience. However, most of these systems do not perform well under certain circumstances such as using different types of chess board pieces than what the CNN has been trained on, or even the conditions of the camera and lighting.

This research paper discusses chess recognition systems, focusing on how they combine traditional gameplay with modern technology. It examines the challenges faced by previous models and highlights the improvements made possible through convolutional neural networks (CNNs) and intelligent move timers. The main goal of this study is to contribute to the field of automated chess recognition, by testing and modifying machine learning and recognition algorithms to find flaws and inaccuracies due to various experimental conditions. As a result, chess enthusiasts can enjoy a more accurate and enjoyable gaming experience.

Related Works

Can Koray, Emre Sumer, and Vitomir Štruc, associate professors from Başkent University in Turkey, created a chess recognition system using a chessboard's geometric shape to easily recognize the corners and grid lines on the board. The geometric rectification warps a tilted board to fit perfectly based on its corner points' coordinates. Their system takes a small region of interest (ROI) on each piece, capturing its color properties and averages it with the other occupied/colored squares of the same type (all the empty white squares, all the squares with black pieces, etc.). Due to their heavy dependance on the colors of the ROI, one of their limitations is that shadows across the board, such as the players' hands, can end up skewing the results. In addition, since the reference colors of the pieces are captured at the start of the game, if there is a drastic change in the lighting conditions the system will be unable to detect the pieces. However, in normal lighting, 162 out of 164 moves were properly recognized by the system, showing a 98.7% accuracy rate (Koray & Sumer, 2016).

Additionally, Georg Wölflein & Ognjen Arandjelovic from the University of St. Andrews School of Computer Science created an automated chess recognition system to assist users in their games (Wölflein & Arandjelović, 2021). The chess recognition system works by recognizing the board position and the piece's color and identity. Due to a chessboard always being an 8 by 8 square figure, computer vision techniques, such as the Canny edge detector, are used to distinguish the edges of the board and the lines between them. Next, the use of convolutional neural networks (CNNs) is needed to decide if a square is empty or occupied by a piece. If there is a piece on a square, then a separate CNN will analyze it to determine which piece and color it is. Their approach is successful with the datasets used to test the system out, seeing a 99.77% accuracy rate for successful piece recognition. However, when testing it out by submitting self-taken images of chessboards, many limitations were present, such as it not being able to distinguish between certain pieces, lighting conditions altering the results, and using a non-ariel camera angle. The recognition system uses Forsyth–Edwards Notation (FEN), a universally standardized chess notation to describe a board state, using alphabetical values for pieces (Berent, 2019). Lowercase represents black pieces, while uppercase represents white pieces.

Methods

In order to conduct a thorough analysis of the chessboard, a series of computational functions are applied. This process involves utilizing edge detection, corner detection, and line detection algorithms to enable the camera to identify and define key properties of a chessboard, and the positions of the pieces within the standard 8 by 8 grid. The algorithms used in this research study were inspired from the open-source code of Georg Wölflein, available on Github and mentioned in his research paper (Wölflein, 2021) & (Wölflein & Arandjelović, 2021).

The corner detector algorithm works by initially resizing the image and converting it to grayscale: a black and white image where each pixel represents only an amount of light (Christensson, 2011). Subsequently, the *_detect_edges_ and the _detect_lines_* functions are called to classify the lines as either horizontal or vertical. These functions work by verifying the image format and then applying a Canny edge detection technique. Canny edge detection works by smoothing out an image and finds areas of brightness changes to be able to sort out clear edges (Wang et al., 2017). Next, it selects the strongest (most distinguishable) out of the potential edges and isolates only those definite edges.

Furthermore, redundant lines are eliminated, and intersection points are determined using the *_get_intersection_points* function. This function analyzes information regarding the horizontal and vertical lines within an image, calculating the distance and angular values for each set of lines it detects. By generating a grid of possible combinations based on these values, the function then uses trigonometric calculations to find the intersection points between the vertical and horizontal lines, finally returning an array of these intersecting points.

In addition, the RANSAC (Random Sample Consensus) algorithm is used to identify a set of corner points that best represent the entirety of the chessboard, filtering out the outliers (Angst, 2014). RANSAC iterates through randomly chosen lines (from the edge and line detectors), hence the name, to continue refining its data. The iterative process continues until either 200 cycles are completed, or at least 30 inliers have been found. (Koray & Sumer, 2016).

Next, an occupancy classifier is utilized to determine whether each square on the chessboard contains a piece or not. The *_classify_occupancy* function takes in the image of the chessboard, its corner points, and information regarding the current player's turn. After warping the board's image based on the corner points, the function iterates through all 64 squares, extracting and processing an image of each one. Additionally, the function uses a pretrained model from Wölflein's provided GitHub directory, called *_occupancy_model,* to predict the occupancy status of each square, ultimately returning an array indicating the occupancy of each square (Wölflein, 2021).

In cases where a piece is detected on a square, the *_classify_pieces* function is used in order to determine its color and piece type (ie. black rook, white pawn). This function predicts the chessboard state using the four corners of the board, along with the occupancy and piece type of each square. If a piece is present, the function uses a pretrained model called *_pieces_model,* also based on Wölflein's GitHub documentation, to predict the type of piece (Wölflein, 2021). Finally, the function maps out the entire board, identifying the total number and position of pieces, which are returned as an array.

Results and Challenges

After several attempts of running the program with various angles and lighting, the results produced were diverse. In certain situations, the program successfully recognized the edges, borders, and corners of the chessboard. However, there are several reasons that cause the program to display skewed results. It is also important to keep in mind that these were tested on one type of chessboard, and may not hold true to every type in existence due to the vast variety. The chessboard configuration in these images is kept constant to ensure an accurate comparison. The position is Samuel Loyd's stalemate with all the pieces on the board. This is beneficial in testing the accuracy of piece detection functions, given that they have a larger dataset to run upon.

One of the impacting factors is the lighting of the image, the angle of that lighting and shadows it creates on the board. For instance, with a strong overhead lighting there are less shadows, so the results are expected to be more precise. However, in actuality, the ariel lighting was reflected off the chessboard, and the program detected it as a cluster of potential edges, as

seen in *Figure 1*. Consequently, the system was unable to render a proper layout of the chessboard, and instead raised an error.

In comparison, when there is a light source from the side, strong shadows are casted from the pieces onto the chessboard, creating a misinterpretation in the image. Interestingly, the edge and board detection of a side-lit board, as shown in *Figure 2*, was very similar to that of the aerial-lit board. This suggests that the lighting angle and shadows are not as responsible for the reflection effects caused by the light compared to the glare.

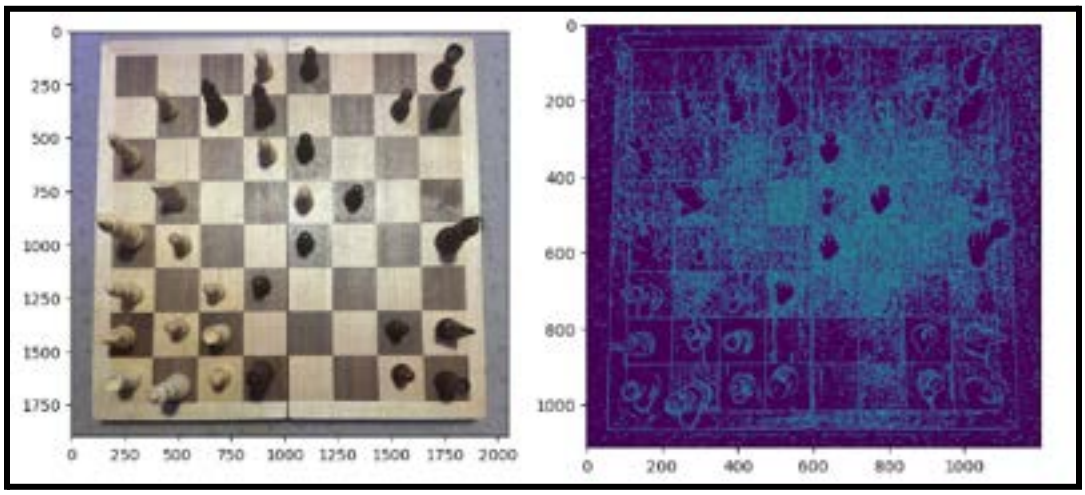

Figure 1: Arial White Lighting

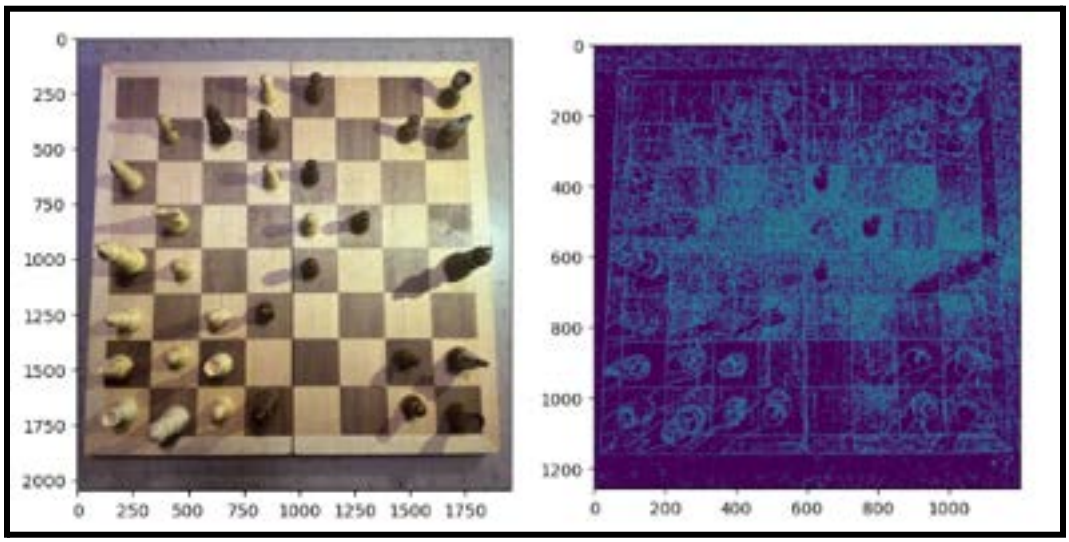

Figure 2: White Side-Angled Lighting

In continuation, most scenarios are made assuming the player is playing in a traditional white/yellow lighted area, but for those playing in LED/multicolored lights, the efficacy of the system differs. The image in *Figure 3* demonstrates this under a purple LED lighting. One benefit that was found through testing multi-colored lights is the reduction of glare and light reflection from the chessboard, ultimately assisting the detector in locating edges on the board. Consequently, the board state was recognized by the algorithm, and it was able to give a FEN representation of the status. However, many of the black pieces on black squares were not noticed due to the dim lighting, portraying a key limitation of the system. Although the system test was effective under purple lighting, there are millions of color combinations that could be made, requiring further testing to be done before making a generalized claim.

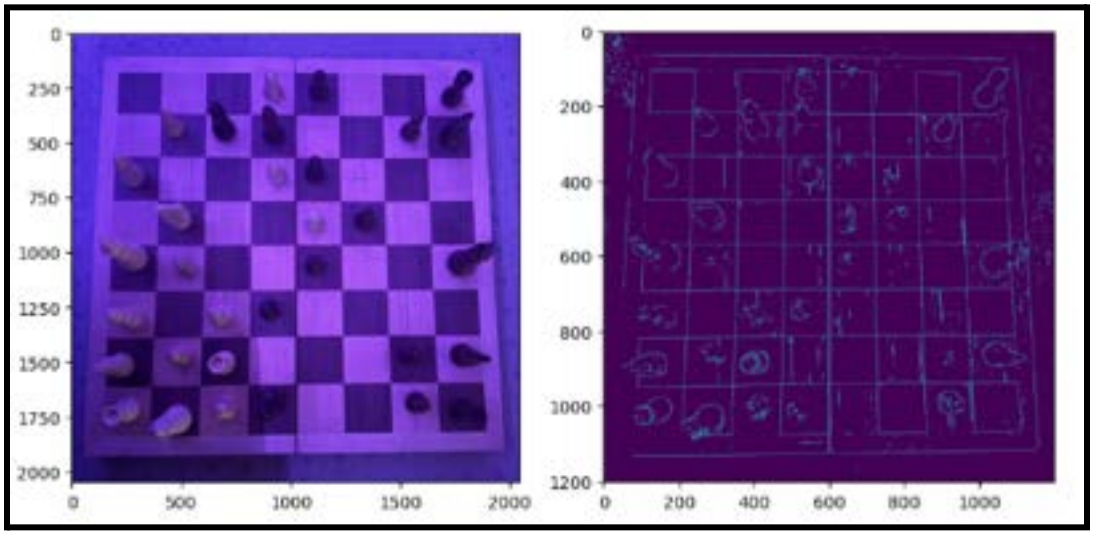

Figure 3: Arial Purple LED Lighting

Moreover, the board that was used to test the system had years of use, resulting in scratches and wood marks on it. The recognition systems detected them as edges in the model, lowering the accuracy of the given outputs. Extra lines are very problematic in machine imaging, as they are picked up by the edge and corner detectors and cause misinterpretations in the overall processing of the image. In certain cases the background of the image can alter the results as well. For instance, when the chessboard is on a carpeted surface, as seen in *Figure 4*, the carpet strands are mislabeled as chessboard edges, rendering the whole image useless. Users of the chess recognition program are not always going to be playing on an even and homogenous surface, highlighting a key limitation in the universality of the chess recognition system.

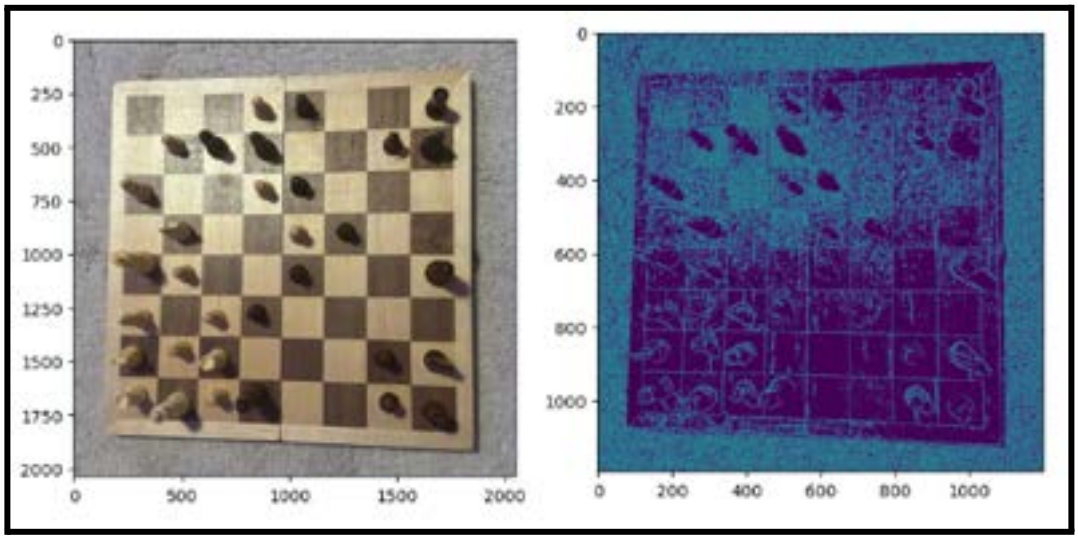

Figure 4: Arial White Lighting on Carpet

To add on, when a camera positon is non-arial, the recognition system is still generally effective in detecting board edges, corners, and piece outline. While *Figures 5 and 6* depict the struggles accidental carpet corner detection, their piece detection rates are much higher than the previous datasets. This is due to a lack of glare caused from an arial-lighting reflection by the chessboard.

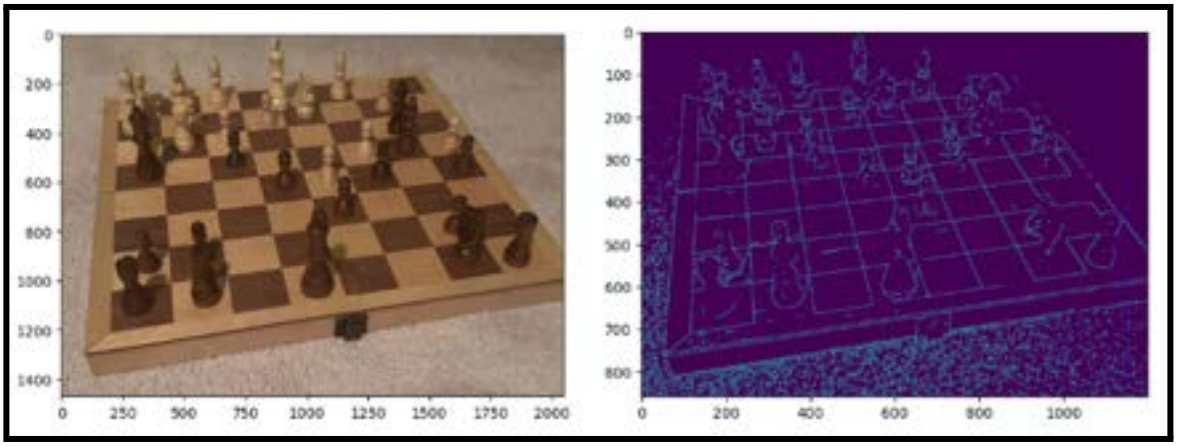

Figure 5: Non-Arial Yellow Lighting on Carpet

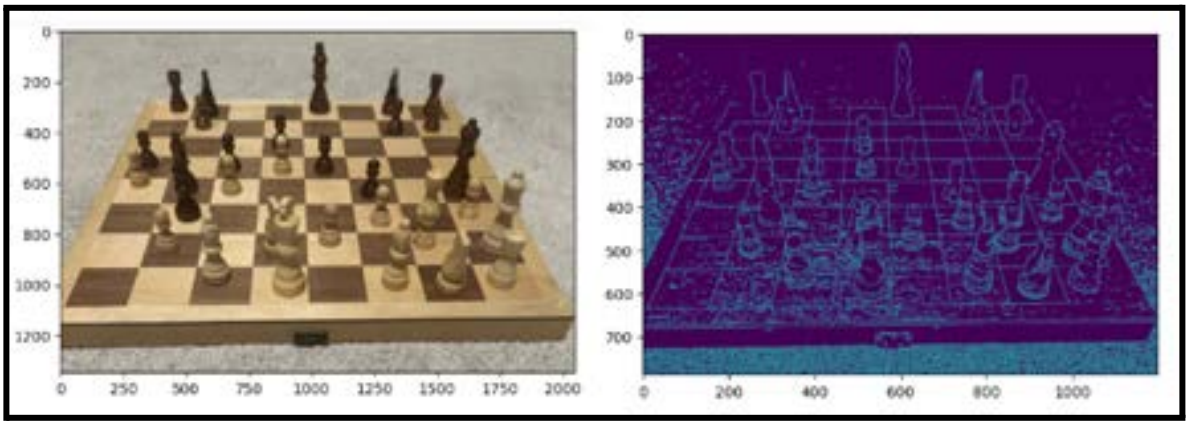

Figure 6: Non-Arial White Lighting on Carpet

Lastly, to ensure the theory that glare from lighting is the most culpable factor in skewing detection rates, a photo of a chess board was taken from an arial angle in yellow lighting, but with no direct glare on the board. As seen in *Figure 7*, the hypothesis was proven correct, as a significant amount of the pieces are accurately identified, and there are no external lines being detected by the canny edge detector. The detection software was also able to create an FEN diagram of the pieces it detected based on the board's state in *Figure 8*. In *Figure 9*, a visual representation of the FEN model is shown to simplify the results in a way that could be used as a UI for real-world integrations of the chess recognition system. Although the recognition software was not able to produce a high-accuracy representation of the chessboard state, it is still an achievement for it to be able to outline and attempt to predict the occupancy status, as compared to previous attempts. While the *_classify_pieces* and *_classify_occupancy* functions could definitely be improved, the successful board, edge, and piece detection proves that glare and heterogeneous surfaces were the most detrimental factors impacting the software.

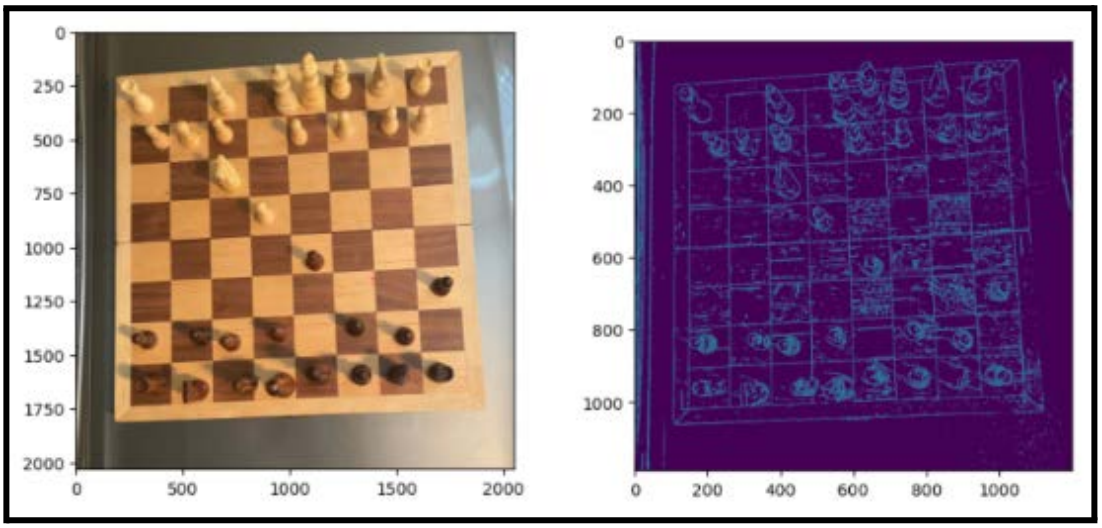

Figure 7: Arial Yellow Lighting with no Glare

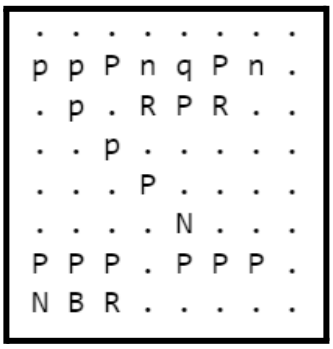

Figure 8: Detector-Produced FEN sequence

Figure 9: Visual Representation of FEN

The most common error that was produced by the recognition system was that the chessboard could not be located due to an excess amount of lines detected, as displayed by *Figure 10*. This accurately reflects the visual observations, as instances with numerous false lines detected did not produce a clear FEN visual, whereas those without such issues were successful in doing so (see *Figures 3 and 7*).

```
Cell In[16], line 24, in find_corners(img)
     22 #print(lines)
     23 if lines.shape[0] > 400:
---> 24     raise ChessboardNotLocatedException("too many lines in the image")
     25 all_horizontal_lines, all_vertical_lines = _cluster_horizontal_and_vertical_lines(
     26     lines)
     28 horizontal_lines = _eliminate_similar_lines(
     29     all_horizontal_lines, all_vertical_lines)

ChessboardNotLocatedException: chess recognition error: chessboard could not be located: too many lines in the image
```

Figure 10: Exception raised due to “too many lines in the image”

Due to the global popularity of chess, there are thousands of variations of pieces and boards, all having the same main components, but are differently represented. For instance, Figure 11 and 12 show two very different chess sets. Both are interpretable by a human chess player, but a computer will read them very differently, based on what types of pieces it was previously trained on. The lack of universal usage of the chess recognition software is a significant limitation, prohibiting millions of chess players from being able to benefit from it, illustrating another potential aspect to improve on.

Figure 11

Figure 12

Future Research and Applications

In addition to the research and results that were concluded, multiple features can be added to the program to enhance its user interactions. One of these ideas is the integration of a move timer in the chess recognition system. This innovative addition aims to revolutionize time management by being able to detect when a player's hand is on and off the piece. Unlike conventional timers that allow for delays and inaccuracies, the proposed move timer works in real-time to keep track of each players timing in the game, whereas it takes a few seconds of delay for the player to hit the timer after making their move. The successful implementation of the new algorithm heavily relies on a functioning occupancy classifier and piece detection mechanism. These components must work together seamlessly to accurately identify when a player is handling a piece. Future research will focus on refining these algorithms to ensure they can reliably detect hand movements and piece interactions under various visual and physical conditions, based on the styles of the chess pieces and the camera quality.

Additionally,the application of this technology could be expanded beyond traditional chess games to include online and virtual chess platforms. This would involve adapting the system to recognize and track digital representations of chess pieces and hand movements in a virtual environment, which can be done building upon the current progress of the recognition system. There could also be potential for a built in user interface so FEN notations can automatically be converted and saved as standard chess pieces, increasing the convenience for users.

Overall, these future enhancements have the potential to significantly improve the user experience, making chess more engaging and fair. Continued research and development in machine learning and artificial intelligence will be crucial in achieving these advancements and ensuring their practical implementation.

References

1. Acher, M., & Esnault, F. (2016). Large-scale analysis of chess games with chess engines: A preliminary report. *arXiv preprint* arXiv:1607.04186 [cs.AI]. Retrieved from https://arxiv.org/abs/1607.04186
2. Angst, R. (2014, April 30). *RANSAC: Random Sampling And Consensus* [PowerPoint slides]. Stanford Electrical Engineering. Retrieved February 15, 2024, from https://web.stanford.edu/class/cs231m/spring-2014/docs/ransac.pdf
3. Berent, A. (2019, March 2). *Forsyth-Edwards Notation*. Adam Berent Software & Hobbies. Retrieved August 18, 2023, from https://adamberent.com/forsyth-edwards-notation/
4. Bishop, C. M. (2013, February 13). *Model-based machine learning*. The Royal Society Publishing. Retrieved October 4, 2023, from https://royalsocietypublishing.org/doi/full/10.1098/rsta.2012.0222
5. *Chess board with chess set in opening position 2012*. (2012, November 3). Wikimedia Commons. Retrieved April 15, 2024, from https://commons.wikimedia.org/wiki/File:Chess_board_with_chess_set_in_opening_position_2012_PD_03.jpg
6. *Chess History: The Astonishing History of Chess (With Timeline)*. (2021, October 15). The Chess Journal. Retrieved January 3, 2024, from https://www.chessjournal.com/chess-history/
7. Christensson, P. (2011, April 1). *Grayscale*. TechTerms. Retrieved March 7, 2024, from https://techterms.com/definition/grayscale
8. Cicchetti, A., Ciccozzi, F., Mazzini, S., Puri, S., Panunzio, M., Zovi, A., & Vardanega, T. (2012). CHESS: A model-driven engineering tool environment for aiding the development of complex industrial systems. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering* (ASE '12) (pp. 362-365). https://doi.org/10.1145/2351676.2351748
9. Gusev, D. A. (2021, June 16). *Using Modern Chess Software for Opening Preparation*. ERIC Institute of Educational Sciences. Retrieved July 24, 2023, from https://eric.ed.gov/?id=ED617407
10. Holt, J. (2024). *25 Unique and Unusual Chess Sets*. Beautiful Life. Retrieved May 2, 2024, from https://www.beautifullife.info/industrial-design/top-15-original-chess-sets/
11. Khoche, K., Gurav, S., Pundir, R., Chrotiya, S., & Narooka, P. (2019). Application based smart chess board using interactive GUI design. *International Journal of Computer Science Trends and Technology (IJCST)*, 7(1), 49. Retrieved from http://www.ijcstjournal.org
12. Koray, C., & Sumer, E. (2016). A computer vision system for chess game tracking. In L. Cehovin, R. Mandeljc, & V. Štruc (Eds.), *21st Computer Vision Winter Workshop, Rimske*

Toplice, Slovenia, February 3–5, 2016 (pp. 1-7). Retrieved from https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/21.pdf

13. Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems, 33*(12), 6999-7019. https://doi.org/10.1109/TNNLS.2021.3084827
14. Mahesh, B. (2020). Machine learning algorithms - A review. *International Journal of Science and Research (IJSR)*, *9*(1). Retrieved from http://www.ijsr.net
15. Mendez, K. M., Pritchard, L., Reinke, S. N., & Broadhurst, D. I. (2019). Toward collaborative open data science in metabolomics using Jupyter Notebooks and cloud computing. *Metabolomics*, 15, Article 125. https://doi.org/10.1007/s11306-019-1588-0
16. Panchal, H., Mishra, S., & Shrivastava, V. (2021). Chess moves prediction using deep learning neural networks. In *2021 International Conference on Advances in Computing and Communications (ICACC)* (pp. Page numbers). IEEE. https://doi.org/10.1109/ICACC-202152719.2021.9708405
17. Wang, W., Tan-Torres, A., & Hamledari, H. (2017). *Lecture #06: Edge Detection*. Stanford Department of Computer Science. Retrieved November 20, 2023, from http://vision.stanford.edu/teaching/cs131_fall1718/files/06_notes.pdf
18. Wei, Y.-A., Huang, T.-W., Chen, H.-T., & Liu, J.-C. (2017). Chess recognition from a single depth image. In *2017 IEEE International Conference on Multimedia and Expo (ICME)* (pp. Page numbers). IEEE. https://doi.org/10.1109/ICME.2017.8019453
19. Wölflein, G., & Arandjelović, O. (2021). Determining chess game state from an image. *Journal of Imaging, 7*(6), 94. https://doi.org/10.3390/jimaging7060094
20. Wölflein, G. (2021, May 25). *Determining chess game state from an image*. GitHub. Retrieved September 22, 2023, from https://github.com/georg-wolflein/chesscog
21. Wölflein, G., & Arandjelović, O. (2021, April 25). *Dataset of Rendered Chess Game State Images*. OSF. Retrieved August 3, 2023, from https://osf.io/xf3ka/
22. Xie, Y., Tang, G., & Hoff, W. (2018). Chess piece recognition using oriented chamfer matching with a comparison to CNN. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. Page numbers). IEEE. https://doi.org/10.1109/WACV.2018.00221