Chess Lens: Digital Chess Projection

1. Problem Statement

Develop a computer vision system that analyzes over-the-board chess matches to detect and classify each move made by the players.
We have been watching chess games form childhood and even now, over the board chess even at the highest level is not automated for online stream.
Identifying the chess pieces is the biggest challenge in the assessment of this problem statement.

2. Dataset

The following is the information about chesscog which used object detection techniques for recognizing a board position efficiently. The chess positions have been taken from standard open library of chess from 2851 games.
For testing we are using a Video dataset and sampling on each move time trigger to get the image and run the inference on.

2.1. Synthetic Dataset Generation

In this study, a pioneering methodology for generating datasets is introduced. It leverages highly detailed 3D models of chess pieces and boards to craft a vast synthetic dataset. This dataset aims to closely emulate real-world scenarios by integrating nuances such as variations in lighting conditions, angles of observation, and diverse styles of chess pieces. By utilizing advanced 3D modeling techniques, the generated dataset not only reflects the complexity of real-world environments but also offers a controlled setting for training and testing machine learning models.

2.2. Augmentation Techniques

To fortify the robustness of the models developed using the synthetic dataset, a series of augmentation techniques are applied. These techniques encompass a spectrum of transformations, including random rotations, scaling adjustments, and the introduction of noise. By subjecting the dataset to such diverse alterations, the models are trained to adapt to a wide array of real-world scenarios, ensuring their ability to generalize effectively. This augmentation process serves as a crucial step in enhancing the resilience and versatility of the trained models, enabling them to perform optimally across various environments and conditions.

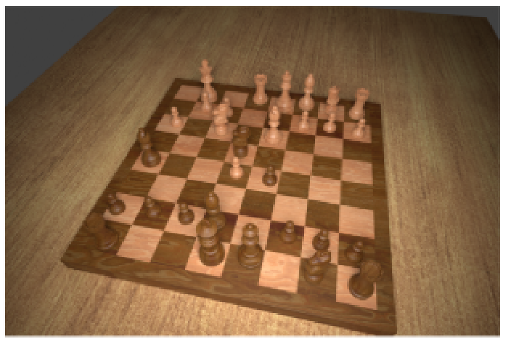

Figure 1. Camera Flash.

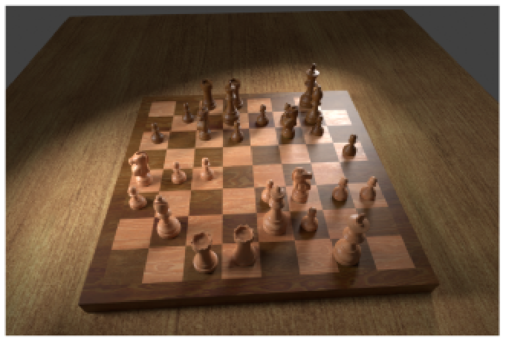

Figure 2. Spotlight

3. Chessboard Localization and Transformation

To determine the location of the chessboard's corners, we rely on its regular geometric nature. Each square on the physical chessboard has the same width and height, even though their observed dimensions in the input image vary due to 3D perspective distortion. A chessboard consists of 64 squares arranged in an 8×8 grid, so there are nine horizontal and nine vertical lines.

3.1. Intersection Points Detection

The algorithm initially detects horizontal and vertical lines by converting the image to greyscale and applying the Canny edge detector. Subsequently, the Hough transform identifies edge-formed lines, which are then split into hor-

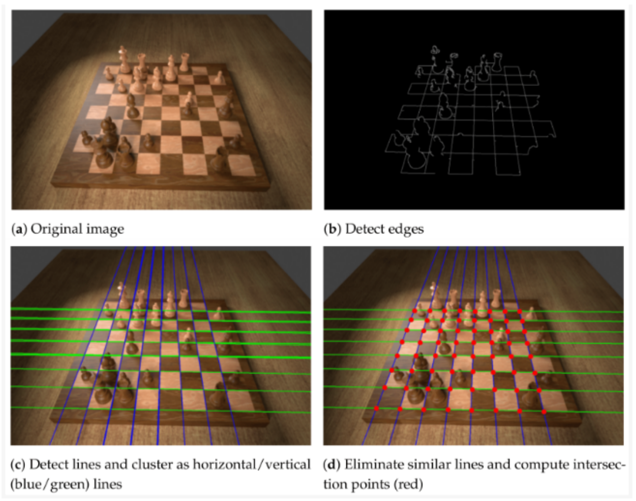
(a) Original image
(b) Detect edges
(c) Detect lines and cluster as horizontal/vertical (blue/green) lines
(d) Eliminate similar lines and compute intersection points (red)

Figure 3. Intersection Point Section

izontal and vertical categories and similar ones are eliminated. To robustly classify lines, an agglomerative clustering algorithm is utilized, considering the smallest angle between lines as the distance metric. The mean angle of top-level clusters determines vertical and horizontal lines. To remove similar lines, the mean vertical line is identified, and DBSCAN clustering groups similar horizontal lines based on intersection points, retaining only the mean horizontal line from each group. The process is repeated vice-versa for vertical lines, computing all intersection points

3.2. Perspective Correction:

When fewer than nine horizontal and vertical lines are detected, additional lines are needed. To determine their likely positions, the input image is warped so that intersection points form a regular grid of squares, simplifying analysis. A homography matrix H is computed using a robust RANSAC-based algorithm, even effective when lines are missing or extra ones are detected. The input image and inlier intersection points are then warped according to H, yielding a transformed image for further processing.

3.3. Occupancy and Piece Classification:

We find that performing piece classification directly after detecting the four corner points with no intermediate step yields a large number of false positives, i.e., empty squares being classified as containing a chess piece (see Figure 5). To solve this problem, we first train a binary classifier on cropped squares to decide whether they are empty or not.

Implements two CNN layers: The first CNN detects if a square is occupied or not (occupancy classification). The second CNN classifies the type of piece on each square. This dual-stage approach helps in reducing classification

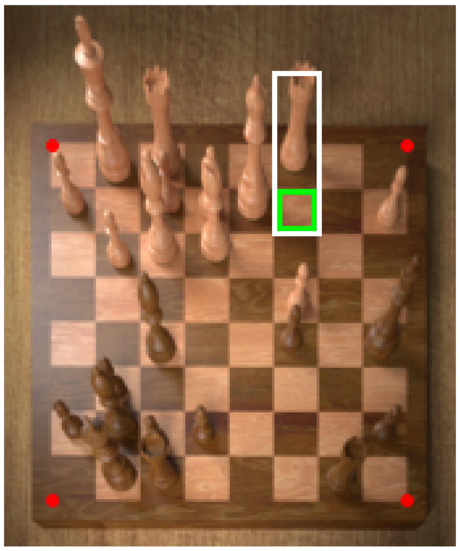

Figure 4. An example illustrating why an immediate piece classification approach is prone to reporting false positives. Consider the square marked in green. Its bounding box for piece classification (marked in white) must be quite tall to accomodate tall pieces like a queen or king (the box must be at least as tall as the queen in the adjacent square on the left). The resulting sample contains almost the entire rook of the square behind, leading to a false positive

errors and improving the overall accuracy of the system.

It has also used Ensemble Learning to reduce misclassification, an ensemble approach is used, where multiple models vote on the classification results, improving accuracy and reliability.

4. Transfer Learning for Adaptability

Leveraging transfer learning, the system can adapt to new chess sets with minimal additional data. This is achieved through fine-tuning the model on a small dataset comprising just a few images of new chess sets.
The system includes a one-shot learning model that requires only a single image of a new chess set to adjust its parameters for accurate piece recognition.

5. Performance Evaluation and Optimization

5.1. Inference over chesscog own dataset

The model is optimized for real-time applications, with rapid inference times that support on-the-fly analysis during chess games.
It employs various evaluation matrices which provides the relevant scenario of a model made for chess.

The following is the actual inferences over the chesscog dataset:

Table 1. Performance metrics of the chesscog

Metric	Train	Validation	Test
Mean number of incorrect squares per board	0.27	0.03	0.15
Percentage of boards predicted with no mistakes	94.77%	97.95%	93.86%
Percentage of boards predicted with $\leq$ 1 mistake	99.14%	99.32%	99.71%
Per-board corner detection accuracy	99.59%	100.00%	99.71%
Per-square occupancy classification accuracy	99.81%	99.97%	99.92%
Per-square piece classification accuracy	99.99%	99.99%	99.99%

Table 2. Performance metrics of our own Dataset

Metric	Test
Mean number of incorrect squares per board	0.95
Percentage of boards predicted with no mistakes	6.34%
Percentage of boards predicted with $\leq$ 1 mistake	74.62%
Per-board corner detection accuracy	96.72%
Per-square occupancy classification accuracy	98.68%
Per-square piece classification accuracy	96.68%

5.2. Chesscog model over our Video Data Set

This dataset has been meticulously curated by our team. We recorded each move of the game we played, adhering to standard game practices, and timestamped each move. These timestamps were utilized to sample images from the gameplay, enabling predictions to be made over these images.

6. Evaluation Analysis and Comparison

6.1. Aditya:

Based on the performance metrics obtained for the digital projection of our chess game dataset, we can make the following observations:

- Fraction of boards with no or <=1 mistake were lesser than chesscog as:
  - The predictions were affected by the angles of images and the board area relative to the overall image area.
  - Higher piece-to-square ratio led to the creation of dense overlaps between the network of adjacent pieces that led to more mispredictions.
  - The structure, quality, and orientation of pieces were different leading to mispredictions for similar and overlapped pieces
  - Per-board error in corner and square detection is slightly more than 1 indicating poor performance of the model after adapting to our non-synthetic dataset in contrast to the chesscog data.

- Average per-square piece and occupancy classification accuracy was similar to that of chesscog due to:
  - Standardized image preprocessing (resize and grayscale)
  - Edge detection focuses on salient features, including chessboard square boundaries.

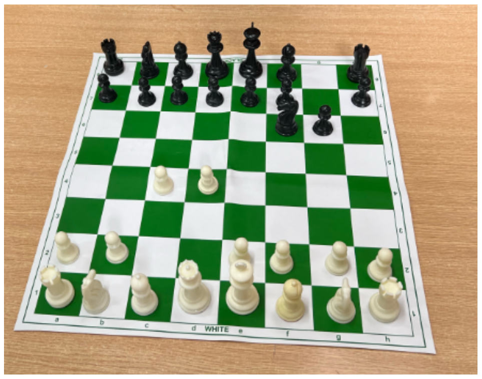

Figure 5. A frame sampled from our video dataset

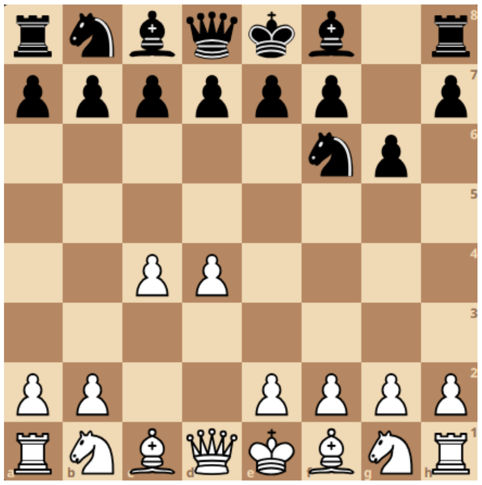

Figure 6. Digital Projection of the above frame

– Robust homography estimation corrects for perspective distortions and camera angle variations.

– Image warping and border refinement enhance accuracy by removing unwanted regions.

6.2. Chirag:

The dataset that has been used by chesscog has been made synthetically even though it has made under data augmentation and various spot light scenario to pose a real world situation, it still does not performs excellently good over real world dataset that we prepared on our own.
The various forms of dataset that we produced includes one with static camera so that we can get a still position, one with slightly moving camera position so that we can get a more realistic view of a chess game kind of situation.
While we were able to reproduce the chesscog results over its test data, we did not get the same result for our own made dataset. ON an average it was having around 1 error per chess board position which looking at real world application is very poor. While it was still able to perform good for metric like average occupancy and classification, the other metric like percentage of correct prediction of board with no and 1 mistake gives a better picture that it still needs to go through betterment.
One more thing to note here is that in chesscog data the sizes of the model pieces used was small and it was posed such that it only occupied one square from the angle images were taken. That is, it seems like it mostly covered only one square with covering less than 10 percent of any other squre in case of large pieces such as king and queen. But in our dataset the pieces were a bit larger and king and queen size with the angle we clicked photographs from covered around 25 percent areas in some photos. This has played some significant role in the error that chesscog generated. When black queen is on a white square it upper part seemingly rests on the black square and classify queen as something like a bishop while in other case it mis-classifies both the square or just adds an extra piece on the black square.

6.3. Himanshu:

The performance evaluation unveils notable disparities between our model's performance on the Chesscog dataset and our own dataset. The Chesscog dataset showcases exemplary results, boasting minimal errors across various metrics. Notably, we achieve impressively high accuracies in per-square occupancy and piece classification, indicating our robust understanding and recognition of chessboard configurations. Moreover, our corner detection accuracy consistently registers at a high level, affirming our ability to accurately localize key board features. In contrast, our performance on our dataset, derived from recorded games, exhibits a slight decline in accuracy metrics. The mean number of incorrect squares per board is noticeably higher, and the percentage of boards predicted with no mistakes is substantially lower. These disparities suggest potential challenges in generalizing our performance from the curated Chesscog dataset to our unique dataset. Factors such as variations in lighting conditions, camera angles, and board states might contribute to the observed differences. Additionally, the synthetic nature of the Chesscog dataset, compared to the real-world nature of our dataset, could also play a role. While the Chesscog dataset consists of synthetically generated images designed to represent ideal chessboard scenarios, our dataset comprises real-world images captured during actual gameplay, introducing complexities such as varying lighting conditions and occlusions. As a result, it is reasonable to expect some disparity in our performance between the two datasets, with us potentially exhibiting slightly poorer performance on real-world data due to these inherent differences. Addressing this divergence may necessitate strategies such as data augmentation techniques or domain adaptation methods to better align our capabilities with the complexities present in real-world scenarios.

References

- Wölflein, G.; Arandjelović, O. Determining Chess Game State from an Image. J. Imaging 2021, 7, 94. https://doi.org/10.3390/jimaging7060094
- Wölflein, G.; Arandjelović, O. Dataset of Rendered Chess Game State Images; OSF, 2021.
- Urting, D.; Berbers, Y. MarineBlue: A Low-Cost Chess Robot. In International Conference Robotics and Applications; IASTED/ACTA Press: Salzburg, Austria, 2003.
- Sokic, E.; Ahic-Djokic, M. Simple Computer Vision System for Chess Playing Robot Manipulator as a Project-Based Learning Example. In Proceedings of the IEEE International Symposium on Signal Processing and Information Technology, Sarajevo, Bosnia and Herzegovina, 16–19 December 2008.
- Hack, J.; Ramakrishnan, P. CVChess: Computer Vision Chess Analytics. 2014
- Ding, J. ChessVision: Chess Board and Piece Recognition. 2016.
- Xie, Y.; Tang, G.; Hoff, W. Chess Piece Recognition Using Oriented Chamfer Matching with a Comparison to CNN.
- Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database.
- Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database.
- https://medium.com/@daylenyang/building-chess-id-99afa57326cd
- https://github.com/georg-wolflein/chesscog