Chessboard Piece Detection and Move Prediction

Problem Statement

Given an image of a chessboard game, the task is to detect and classify the chessboard pieces and figure out the board’s configuration. Once figured out, we also suggest the next best move based on the player’s turn.

Chess Dataset

The dataset consists of chessboard images with annotated bounding boxes for each piece on the board as well as the piece type (King, Bishop, etc.).

All the images are taken in a controlled environment of fixed orientation and lighting setting, and all the pieces belong to the same set (design).

The board configuration is different in all the images and not necessarily a valid game of chess.

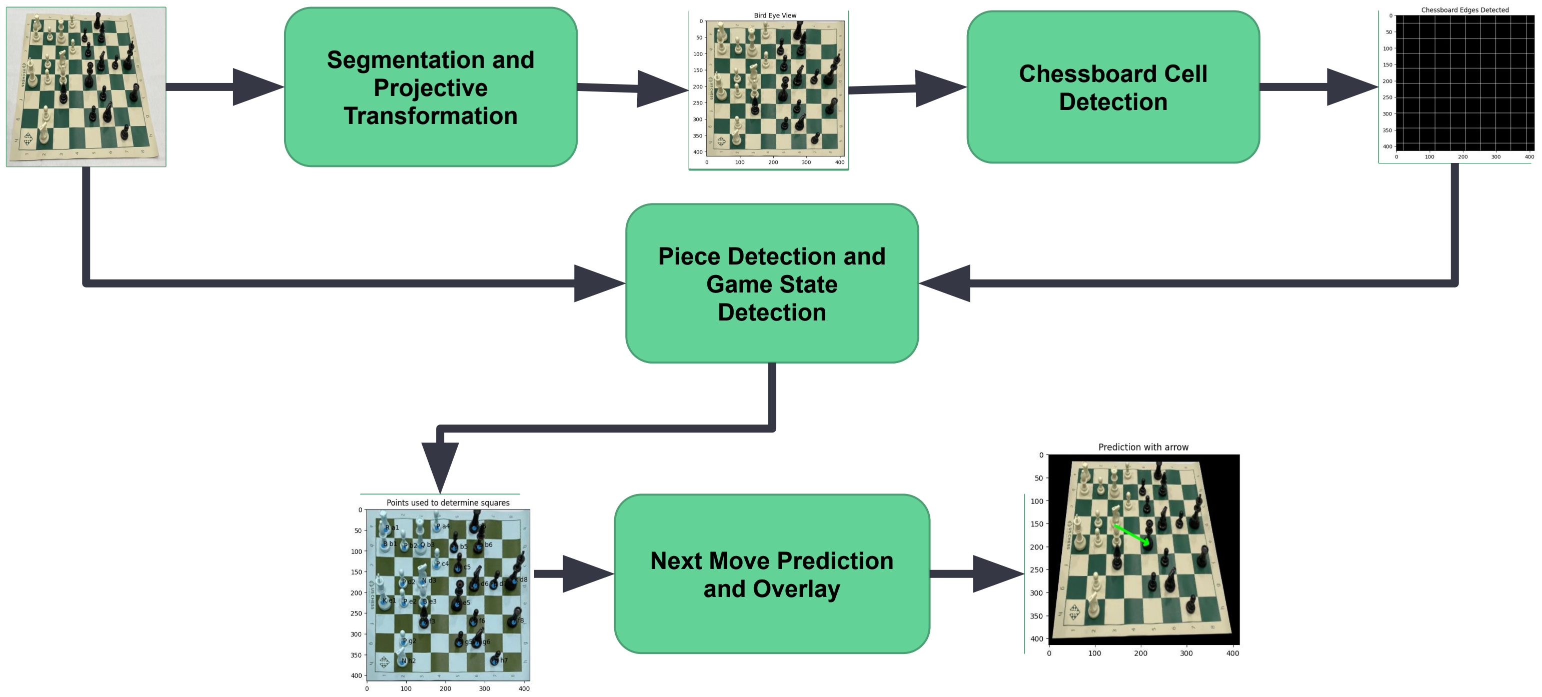
Examples of Dataset

Only one piece – not a valid game state

Valid game states from real games

Methodology: Adopted Pipeline

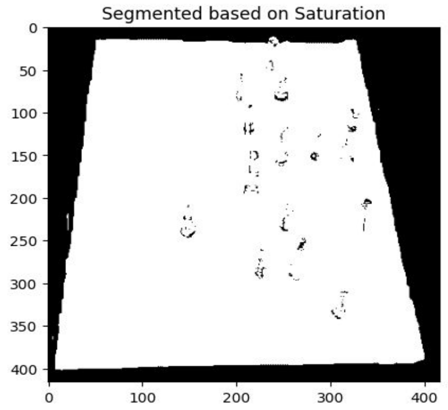
Our adopted pipeline involves the following steps:

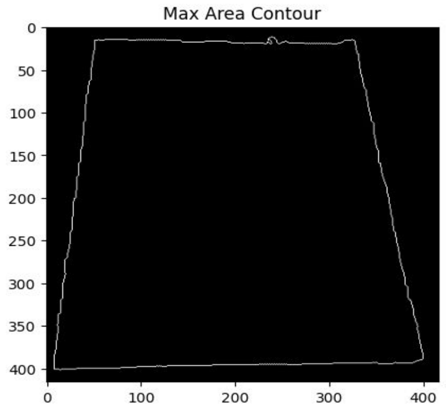

Segmentation and Projective Transformation
Bird Eye View
Chessboard Cell Detection
Chessboard Edges Detected
Piece Detection and Game State Detection
Points used to determine squares
Next Move Prediction and Overlay
Prediction with arrow

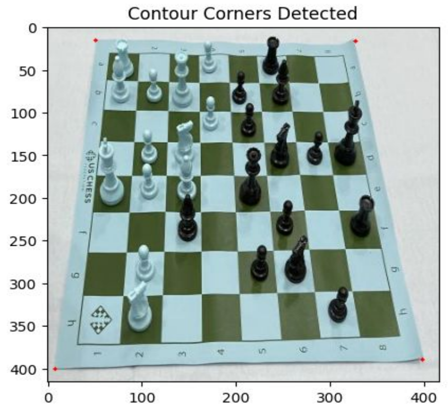
Chessboard Segmentation and Projective Transformation

First, we segment out the chessboard from the background by thresholding the saturation value.

To estimate the projective transformation that brings the image to a top-down view, we first detect the 4 corners of the chessboard. This is done by detecting the max area contours over segmented image.

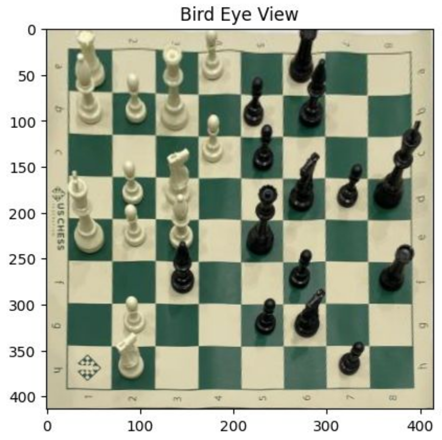


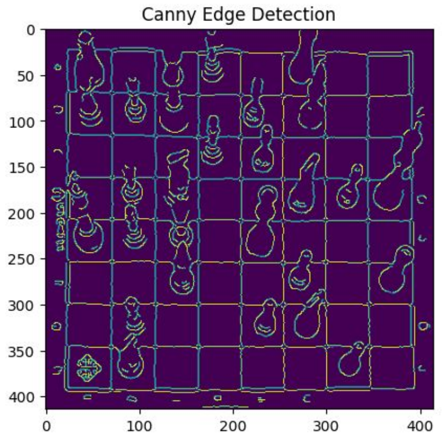


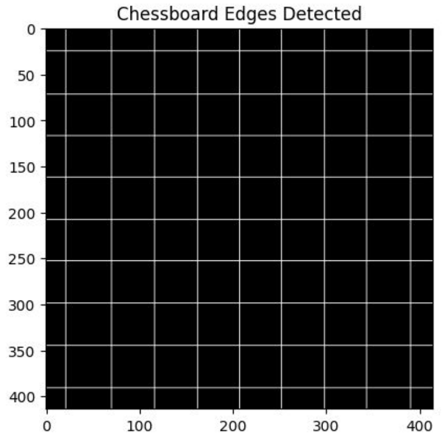


Edge Detection and Chessboard Cell Detection

Once the image is in top-down view, we apply canny-edge detection. After testing various approaches, we developed our algorithm which finds the chessboard cells by multiple horizontal and vertical scan-lines over the detected edges.



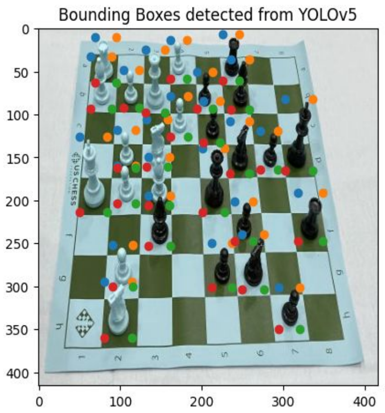


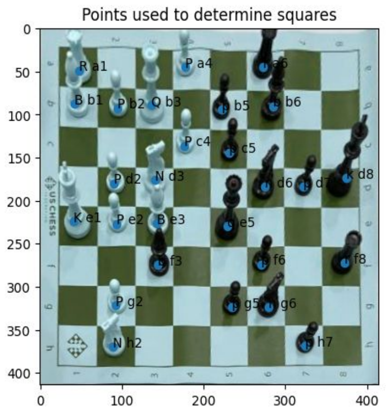


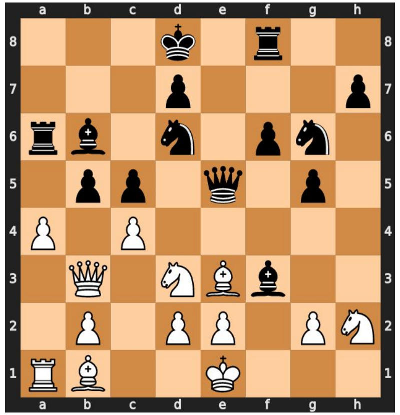
Chess Pieces Detection and Game State Detection

The chess pieces are detected using a pre-trained YOLOv5 object detector.

The bounding box of the detected chess pieces is used to determine the cell on which the piece lies. This is then fed to the Chess library to configure the state of the game.

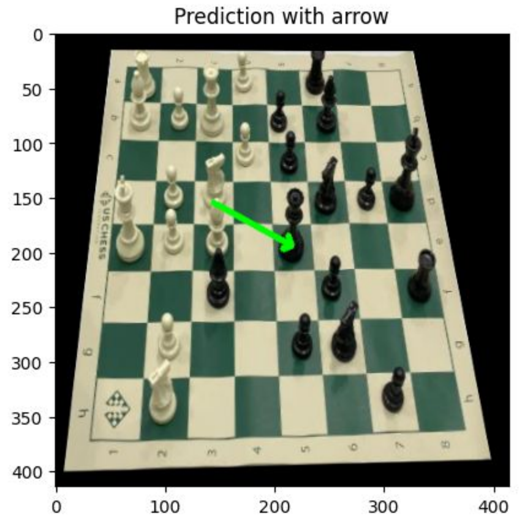






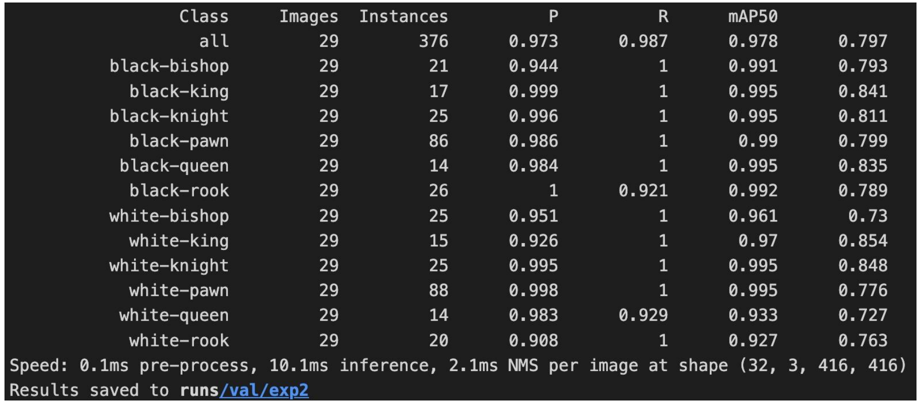
Next Move Prediction and Visualization

Finally, a chess engine (Stockfish) is used to generate the next best possible move based on the board's configuration and next player's turn. The resultant move is then denoted by an arrow reprojected on the original image. This can be then used as an interactive guide.



Results

YOLO Detection Metrics:

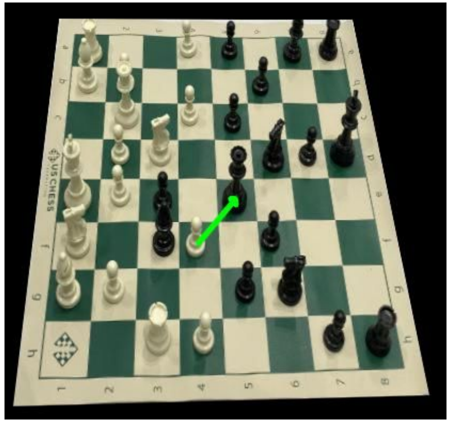


```
                 Class     Images  Instances          P          R      mAP50
                   all         29        376      0.973      0.987      0.978      0.797
          black-bishop         29         21      0.944          1      0.991      0.793
            black-king         29         17      0.999          1      0.995      0.841
          black-knight         29         25      0.996          1      0.995      0.811
            black-pawn         29         86      0.986          1       0.99      0.799
           black-queen         29         14      0.984          1      0.995      0.835
            black-rook         29         26          1      0.921      0.992      0.789
          white-bishop         29         25      0.951          1      0.961       0.73
            white-king         29         15      0.926          1       0.97      0.854
          white-knight         29         25      0.995          1      0.995      0.848
            white-pawn         29         88      0.998          1      0.995      0.776
           white-queen         29         14      0.983      0.929      0.933      0.727
            white-rook         29         20      0.908          1      0.927      0.763
Speed: 0.1ms pre-process, 10.1ms inference, 2.1ms NMS per image at shape (32, 3, 416, 416)
Results saved to runs/val/exp2
```

Qualitative – There is no metric for determining whether the predictions made by our classical CV model for chessboard parameters were accurate. However, qualitatively, the results are good for move prediction, as can be seen from the outputs.

Relevance of proposed solution

Two chessboards,
Harder to track



Our overlaid chess movements,
Easier to track and analyse

Room for Improvement

The current setup is very sensitive to the orientation of the chessboard which can be made more robust by including deep learning approaches for chessboard segmentation and cell detection.

The YOLOv5 detector fails to classify pieces correctly when applied over a different set of pieces with different features. The model can be made more robust by introducing various data from different orientations and chessboard pieces sets.

This pipeline can be further adopted to a real-time detection and suggestion model working on videos.

Individual Contributions

Niranjan: Corner detection and looked into YOLOv5 for piece detection, Presentation

Aadit: Cell detection and integration of top-down view with chessboard engine, Presentation

Vibhu: Image processing, segmentation and edge detection, Presentation

Note: The contributions only describes who led the particular task. All three of us worked together and understand each aspect of the task in detail.