BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2025
http://excel.fit.vutbr.cz

Digitalization and visualization of chess games from video

Filip Roman*

Abstract
The main goal of this thesis is to design a method for automatic digitization of a chess game from a video and its later visualization using computer vision techniques. The proposed system processes a video recording of a chess game by splitting it into frames, detecting the board, and classifying the pieces. It assigns pieces to squares and identifies moves by comparing consecutive frames. Using a chess engine, the system assumes a known initial position and validates candidate moves. As a result of this work, I have developed a system that has been tested on my own videos with high success rates under standard conditions. Detection of individual pieces on the obtained dataset achieved a success rate of 92.6% mAP@0.5 across 12 chess piece classes. This work results in a system that allows players to obtain a reliable transcription and reconstruction of the entire game in digital form.

*xroman16@vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Chess, a game that is growing in popularity daily, is attracting more and more new players who are learning and educating themselves. Tools supporting game analysis, coaching, and sharing as the community expands are becoming very valuable. However, most modern chess analysis systems rely on digital input, overlooking the number of games played on physical boards. Imagine watching two people play a beautiful game: What if we could automatically capture their moves, archive the game, analyze it later with a coach or chess engine, or share impressive combinations with friends?

Electronic chess sets are currently the most popular alternative for digitizing physical chess games at the professional level. However, these sets are prohibitively expensive, making them unavailable to smaller clubs, schools, or casual players. They are also not that easy to set up and move around. Other options, such as physical transcription on paper followed by computer entry, are time-consuming and error-prone, particularly for players under time pressure.

In this work, a system is presented that is capable of detecting and interpreting chess moves directly from standard videos. This method integrates deep learning for piece detection and localization with chess engine validation to ensure accurate move tracking. Using the YOLO [1] object detection algorithm, the system is lightweight and scalable, capable of running on most devices equipped with a standard-quality camera.

This contribution includes a pipeline for real-time move extraction and demonstration of successful performance even under challenging conditions such as occlusions and background noise. This advances the goal of seamlessly digitizing chess games without the need for specialized hardware.

2. Video Processing

The system begins by processing the user's video input. Frames are extracted using Python functions from the Supervision library. A generator function sequentially accesses individual frames, and the system processes only every *n*th frame based on a configurable parameter, which can be adjusted according to the game's tempo. This sampling strategy optimizes performance, as analyzing every frame would be unnecessary.

3. Chessboard and Chess Pieces Detection

After obtaining a frame from the video, the detection phase begins. The first task is to locate the corners of the chessboard. To precisely determine their positions within the image, a model was trained using the YOLOv8 algorithm. This approach was chosen because camera position or external conditions might vary, altering the corners' critical information for mapping chess pieces accurately.

The next step is to detect and classify the type and color of each piece. This was accomplished by training a specialized object detection model to identify all standard chess pieces on the black and white sides. Recognition is performed using the YOLOv8 model, which returns bounding boxes with predicted class labels and confidence scores, as shown in **Figure 1**. The detections are filtered by confirming that the bounding boxes are inside the board area to ensure that only relevant components are considered.

4. Mapping Pieces to Squares on the Board

Once the location of the board corners and chess pieces has been determined, a mapping process assigns each piece to the corresponding square. By applying a perspective transformation [2] using homography, the board is assigned to a fixed coordinate system, ensuring that it appears flat and square. The resulting image aligns with a fixed 8×8 square grid. Each square has fixed pixel boundaries, simplifying the mapping between the pixel coordinates and the board squares.

Each piece is assigned a reference point at the horizontal center of its bounding box and one-quarter down from the top. This point is transformed using the homography matrix into the corrected coordinate system and assigned to the corresponding square.

5. Comparing Boards and Hand Detection

After all chess pieces have been classified and assigned to their respective squares, the system obtains a live digital representation of the board as shown in Figure **Figure 3**. The system uses hand detection as a trigger to monitor activity to determine whether a change in board state represents a move. It is assumed that a move may occur when a player's hand is detected above the chessboard, as shown in Figure **Figure 2**. During such periods, piece detection is temporarily paused to save computational resources and prevent misinterpretations caused by occlusions. The presence of a hand can obscure pieces, leading to potential errors in movement detection. Hand detection uses a dedicated YOLO object detection model, filtered to the projected chessboard area to prevent unimportant activity.

When a move is suspected, the system finds all square changes by comparing the current board state with the previous one. Due to detection noise, there could be more than two changes. All possible move combinations are generated from the set of changed squares. Every candidate move is confirmed against the list of legal moves produced by the Python chess library. If only one candidate matches a legal move, it is accepted. If multiple candidates remain, further filtering is required. Additional logic is implemented to handle special moves such as castling and pawn promotion.

6. Conclusions

The system demonstrates a practical way to digitize chess games from video and performs well under standard conditions, successfully detecting chess moves from real-world scenarios with high accuracy. With more time, I could build a larger and more diverse dataset to improve detection robustness. The system also provides a strong foundation for future work, such as deploying it as a mobile or web application for real-time game analysis and streaming anywhere.

References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[2] Wangbo Zhang, Xiaoyuan Li, and Xingjie Ma. Perspective correction method for chinese document images. In *2008 International Symposium on Intelligent Information Technology Application Workshops*, 2008.