# Computer Vision Project - Group 09

Bouchy Elliot Martins Adrien Rion Léa

## 1 Problem Statement

The objective of this project is to develop a computer vision solution for recognizing and annotating chess pieces on a chessboard in real-time, replacing them with their respective images in an augmented reality setting. The problem involves challenges such as video data acquisition, camera calibration, motion-based piece recognition, and optional augmented reality integration. This requires robust methods for detecting, tracking, and mapping chessboard states accurately.

## 2 Methodology

### 2.1 Data Acquisition

Each team member recorded two chess games: one with a fixed camera and another with a moving camera, ensuring diversity in captured perspectives. Annotated videos included consistent markers for white and black pieces. The captured data was structured as sequences of frames for further analysis.

### 2.2 Camera Calibration

Using the video frames, the calibration process involved identifying the key internal corners of the chessboard, specifically the positions of *a1, a8, h8,* and *h1*, which define the rectangular bounds of the internal chess grid. These corners were detected using computer vision techniques, such as contour detection and corner refinement, to ensure sub-pixel accuracy.

Once these real-world corners were detected, their pixel coordinates in the video frame were mapped to their corresponding logical positions on an ideal chessboard grid, represented in a normalized coordinate space (e.g., (0,0) for a1,(7,0) for a8, etc.). This mapping was used to calculate a homography matrix, which describes the geometric transformation needed to project the ideal grid onto the observed image plane.

The homography matrix was then applied to transform each square of the ideal chessboard into its image representation. This transformation ensured that the chessboard grid remained accurately aligned with the video frame, even if the camera was at an angle or experienced minor shifts. The calibrated grid served as the foundation for subsequent detection and recognition steps, enabling precise identification of chess pieces within their respective squares across all frames of the video.

### 2.3 Piece Detection

**Corner Detection and Grid Alignment:** The system begins by detecting the key internal corners of the chessboard (e.g., *a1, a8, h8, h1*) in the video frame using computer vision techniques like edge detection and corner refinement (e.g., Harris corner detection or Shi-Tomasi). These detected corners are used to compute a homography matrix, which maps the real-world chessboard grid to the image plane. Using this matrix, the ideal chessboard grid is transformed into the observed frame, creating a precise alignment. Each square of the chessboard is then mapped to its Region of Interest (ROI), defined by the pixel coordinates corresponding to the square's boundaries in the image. This ensures that every chessboard square is localized for individual analysis.

**Piece Detection:** Within each ROI, the system uses HSV (Hue, Saturation, Value) color segmenta-

tion to isolate objects (i.e., pieces) from the background. This is done by applying color masks to filter for specific ranges of saturated colors, which helps distinguish pieces from the chessboard itself. After segmentation, the system applies contour detection (using algorithms like OpenCV's `findContours`) to identify the boundaries of potential pieces within the ROI.

The detected contours are analyzed based on their area and shape to eliminate noise or irrelevant artifacts. For example:

Contours that are too small or too large relative to the expected piece size are discarded. Additional shape constraints, such as circularity or rectangularity, are applied to refine detection further. This process is optimized to handle common challenges like minor occlusions (e.g., partial overlaps) and variations in lighting by preprocessing the ROI (e.g., applying Gaussian blur or adaptive thresholding). The result is a reliable detection of pieces in each square, forming the basis for motion analysis and game state updates.

### 2.4 Piece Recognition and Game State Updates

The code implemented in `piece_recognition.py` dynamically updates the game state:

1. Initial Game State: An initialization step assigns placeholder values (-7 and 7) to unknown black and white pieces, respectively.

2. Motion Analysis: By comparing consecutive frames, the code computes row and column differences to infer piece movements. Functions such as `ascending_recognition()` and `decreasing_recognition()` refine the possible piece identities based on movement patterns.

3. Context Updates: A recursive function `context_update()` ensures consistency with chess rules. For example, once all knights are identified, subsequent detections exclude this piece type.

## 3 Results

Each part of the code has been tested on its own and works. Indeed, the `calibration` came from the part 2 of the project but was better to ensure there was no mistake made. Then the `detect_pieces` was first tried on frames alone at different moment of the video to adapt parameters to luminosity and movement. It also allowed us to chose the best method possible for the detection. However, we still have some mistakes on some square detection, especially ones that are *far* from the camera therefore with a weak coloration. Then the detection of moves and the category of piece deduced were tested thanks to game state and matches supplied for the task 1. Finally, the creation of a .json file was also tested based on previous task. Nevertheless, the link between each algorithm is not yet fully working, in particular the .json output.

## 4 Contributions

Bouchy Elliot contributed to: script global organization, basic calibration and the detection/determination of the movement.
Martins Adrien contributed to: script global organization, basic calibration script and the whole logic for the piece recognition.
Rion Léa contributed to: global organization, calibration and transformation of frames, detecting pieces.