

**Chess game registration**

Vision Systems in Robotics

Autors: Jan Gallina  
 Jakub Górski

Field of study: Automatic Control and Robotics

Specialization: Komputerowe Systemy Sterowania

Semester: I

Cracow, 2025 r.

**Table of contents**

**1. Introduction and preliminaries 5**

1.1. O 5

**2. I 7**

2.1. M 7

2.2.1. I 10

**3. Conclusions 36**

**References 38**

1. **Introduction**
   1. **Objective**

The goal of this project was to record a game of chess using a webcam. We decided to use a convolutional neural network (CNN) to detect and classify chess pieces on a real chessboard in real time.

* 1. **Overview of used technologies**

During this project, we made use of various technologies from different areas. To accomplish the project goals, the following tools and libraries were used:

* Python language – main programming language used for implementation
* OpenCV – library for image processing and computer vision tasks
* TensorFlow – to build, train and run the convolutional neural network (CNN)
* NumPy – for numerical operations and matrix manipulation

1. **Method Overview**
   1. Edge detection

The first step in developing the project was to detect the edges of the chessboard using the Canny Edge Detection algorithm, which we had learned during lectures. Based on an image of the empty chessboard (without pieces), edges were detected, and then, using the Hough Transform, the detected edges were drawn as straight lines. This approach made it possible to extract all the chessboard squares. However, the method was highly sensitive to parameter tuning—small movements of the chessboard caused the algorithm to stop detecting the intersection lines correctly.

* 1. Perspective Transformation

The problem described above made it necessary to add perspective transformation to the code. We decided to implement this step manually. During program startup, the user selects the four corners of the chessboard using the mouse, based on the live camera feed. This allows the calculation of a perspective transformation matrix. Applying this transformation places the chessboard in the center of the frame and enables consistent edge detection regardless of slight shifts or camera angles.

* 1. Chessboard Square Extraction

Using the lines detected with the Hough Transform, the image was divided into 64 individual squares. The first step in extracting the squares was to find the intersection points of the detected lines. Then, a dictionary was created that mapped each square’s name (e.g., C1, B8) to its coordinates in the transformed image. These square regions were later used by the neural network to detect chess pieces.

* 1. Piece Detection

The final step of the program is detecting whether a chess piece is present on a given square. Iterating through all the squares identified in the previous step, the convolutional neural network predicts the probability of each square containing a specific piece. Based on this, the program creates an interpretation of the board state, which is printed to the console. After each move, the user must press the “a” key to trigger a new scan of the board. The current board layout is then displayed in the console.

1. **CNN Model**
2. **Results**

a

**References**

**Figure 1.** a

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

**Table 1.** y

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

* w