FINAL PROJECT: PAPER PIANO

Computer Vision

Francisco Javier Ríos Montes | José Ridruejo Tuñón

INTRODUCTION

The PaperPiano project is a computer vision system that uses a Raspberry Pi and a camera to recognize and process piano key patterns encoded with ArUco. This system allows users to interact with a virtual piano, displaying a visualization. The whole project can be accessed through this GitHub repository: Paper Piano Repo

Calibration

Calibration is an essential step in our project. We use a cheesboard pattern to calibrate the camera. This process allows us to correct any distortion caused by the camera lens and obtain an accurate representation of the scene. The resulting values of the calibration, including the RMS, are displayed when running calibration.py. The distortion obtained is not significant enough to be considered.

Pattern Detection

Our system can differentiate simple patterns through image processing. It is specifically programmed to recognize triangles, squares, pentagons, and stars. The algorithms used to recognize shapes are: findContours and Harris to count corners found in each contour.

Sequence Decoder

We have implemented a decoder that memorizes up to 4 consecutive patterns. If the patterns are in the correct order, the system allows the passage to the next block, the paperpiano.

Tracker

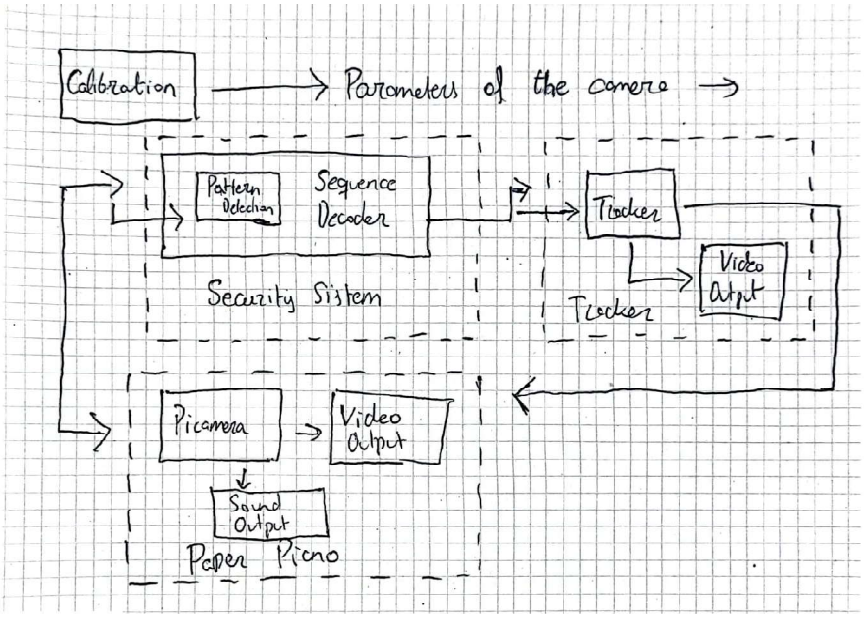
Once the correct pattern sequence is introduced, the tracker is executed. This component displays a bounding box around the geometric shapes and follows them as they move.

PaperPiano

Finally, the PaperPiano works by recognizing the ArUco codes associated with each key, using the threading library to allow two audio tracks to be heard at once. By checking which codes have just been hidden, we make the program play the notes and display the visualization.

METHODOLOGY

This part will be dedicated to a double objective: showing the system diagram and explaining how to run the project. As we can see in the next diagram, the first thing the program does is calibrate the camera. Afterwards, it shows some examples of pattern detection (shapes) and enables the security system. In it, a sequence must be shown in correct order (triangle, square, pentagon and star) to grant access to the tracker. Additionally, the user will have the freedom of executing the paper piano directly.



System Diagram

The first thing that should be done is downloading the src folder, where the codes are included (as well as the documents the modules need). In such folder, 5 folders and a python file (main). The folders have the different modules necessary for each part of the system (Calibration, Pattern detection, Sequence decoder, Tracker and the Paper piano).

The system is structured as a State Machine with 3 states: Blocked, Tracking and Paper Piano, the transitions of which are shown in the diagram. To run the project, then, just run the main.py file, which will create an interactive menu in the command prompt

RESULTS

The system worked well for how basic the techniques behind it are. Two main problems must be highlighted, though: one concerning the tracker and the other with the paper piano. First, the tracker works nicely when only the shape and a white background is in the image, but will detect a lot of different things if the image contains normal things, like the background of a classroom, people, etc. The paper piano, on the other hand, worked by detecting the ARUCO codes, as specified in the introduction, but is sensitive to movement of the camera or the physical piano image. We will talk about this in the next section.

In the test folder of the repository, you will find two different videos, “Decoder & Tracker” and “Paper_Piano”, that act as infographics for our project. They are both brief demonstrations of the results you may expect when executing for yourself, as well as the methods used.

FUTURE DEVELOPMENTS

The main upgrade we have in mind pertains to both the paper piano and the tracker (even though it would affect the whole system). Taking into account that we will have an extension of the subject (Computer Vision) in the last year of the degree, we may be able to apply the knowledge learnt in this project (for example, applying neural networks to detect the hands of the user and make a more smooth piano experience).