Rubik's Cube with Hand Gesture Controls

Tian Grumerec

Univerza v Ljubljani, Ljubljana 1000, Slovenia `tg8827@student.uni-lj.si`
`https://fri.uni-lj.si/sl`

Abstract. The Rubik's Cube has long fascinated individuals across various domains, from education to research. This project introduces an innovative approach to interacting with a virtual Rubik's Cube using hand gesture controls. By leveraging modern computer vision technologies, specifically MediaPipe's Hands module, and combining it with 3D rendering in *p5.js*, we developed an intuitive and interactive system. Users can manipulate the cube in real-time by performing natural gestures, such as swiping or pinching, to rotate layers or select axes. The system integrates gesture normalization, smooth animations, and dynamic visual feedback to enhance usability and accessibility. Inspired by Daniel Shiffman's Rubik's Cube implementation on The Coding Train , this work advances user interaction by enabling real-time, gesture-based control. The resulting system demonstrates high accuracy in gesture recognition, smooth and responsive cube rotations, and a user-friendly interface. This novel integration of gesture recognition with Rubik's Cube manipulation paves the way for future developments in interactive 3D puzzles and gesture-based interfaces.

Keywords: Rubik's Cube · Hand Gesture Controls · Computer Vision · Interactive 3D Visualization · MediaPipe

1 Introduction

In this report we will represent our project, which we made for our second seminar for class Interaction and Information design. The Rubik's Cube[5] is an iconic 3D combination puzzle that has inspired enthusiasts, educators, and researchers alike. In this project, our goal was to implement a fully functional Rubik's Cube that could be controlled through hand gestures, providing an innovative and intuitive interaction method. By leveraging modern computer vision techniques and 3D rendering, we created a system that allows users to rotate the cube by performing specific gestures. This work draws inspiration from a coding challenge by Daniel Shiffman on The Coding Train[6], where a Rubik's Cube was implemented programmatically. Extending this idea, we incorporated gesture recognition to explore new dimensions in interaction and visualization. In this report, we will describe which libraries were used and how the program works.

2 Similar Work

There are numerous online Rubik's Cube implementations, but none of them support hand gesture controls. Some notable examples include rubikscu.be and Grubiks Rubik's Cube which are widely available on the internet. After making some deep research I have also found a more in depth solutions such as Web Application for Competitive Rubik's Cube Solving[1], which had everything I needed in regards of Rubik's Cube itself. The most similar project, which heavily inspired my work, is Daniel Shiffman's Rubik's Cube implementation on The Coding Train, as mentioned in the *Introduction*. However, his project does not implement user interaction for solving the cube; it focuses on automatic solving. Daniel's version was created using Processing.

Additionally, there are several Rubik's Cube solver projects created by individuals, many of which are available on GitHub.

3 Implementation

The system for this project consists of several integrated components that work together to achieve gesture-based Rubik's Cube manipulation. Two biggest parts are Rubik's Cube mechanics and of course gesture recognition. Application's design is very simple - dark background, in the center of which you can see 3D Rubik's Cube. In the top left corner you can also see a preview of camera, on which, landmarks of hands are drawn, if hand is in the picture.

3.1 Rubik's Cube

A 3D-rendered Rubik's Cube was developed using the *p5.js*[4] library to provide a visually engaging and interactive interface.

The Rubik's Cube mechanics were designed to mimic the functionality of a physical cube. It was represented as a 3x3x3 grid of smaller cubes, known as cubies[1]. Each layer of the cube could be rotated along the x, y, or z axis, allowing complete manipulation. The mapping of gestures to specific actions was an essential aspect of the project. Swiping gestures were used to rotate layers, while pinching gestures enabled the selection of rotation axes. Animations were implemented to ensure that these transformations were not only functional but also visually smooth and intuitive, enhancing the overall user experience. This was achieved by interpolation and manipulation of before mentioned cubies.

We have also implemented visual feedback mechanism. It consists of highlighting the currently selected face as seen in image1 and image2, which would be rotated in case we make a swipe gesture. This selected face obviously changes, depending on which layer and/or axis we choose.

3.2 Gestures

Gesture recognition was implemented through MediaPipe's Hands module[3], which tracks 21 key landmarks on the user's hand to interpret movements. Use

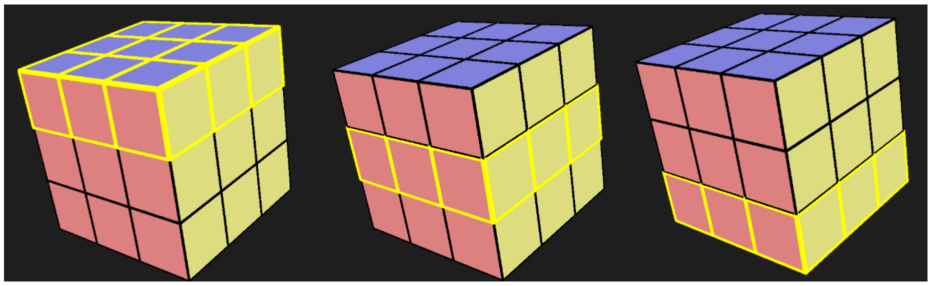

Fig. 1. Screenshot of the Rubik’s Cube, each time with different layer selected.

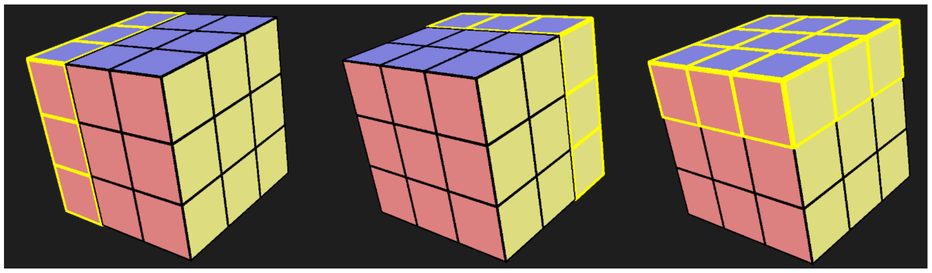

Fig. 2. Screenshot of the Rubik’s Cube, each time with different axis selected.

of webcam was implemented with the help of library EasyCam[2]. Together they provided us with working hand detection.

Gesture detection was a critical component of the project and involved several challenges. There were two major challenges, the first of which is how to ensure a consistent recognition of gestures regardless of hand size, distance from the camera, hand’s orientation. For example if you only calculated distances between between the landmarks and you moved your hand away from the camera, distances would get smaller and detect a fist, even though your hand is extended. Or another example, if you rotate your palm sideways, landmarks also have smaller distances, leading to the same result. We fixed this by using the diagonal of the bounding box as a stable scale. Then we calculated thumb to pinky distance and distance between wrist and middle finger base. We normalized these distances with the diagonal value (hand scale). The first distance helped with camera distance, while the second one helped with palm rotations.

The principles of interaction design played an important role in guiding the development of the gesture control system. The goal was to minimize cognitive load by using natural gestures that users could perform instinctively, such as swiping to rotate a layer. But there were a lot of actions needed to be implemented, so we had to resort to some not so intuitive gestures for example pinch to toggle between axis and "rock n’ roll" sign to switch between layers.

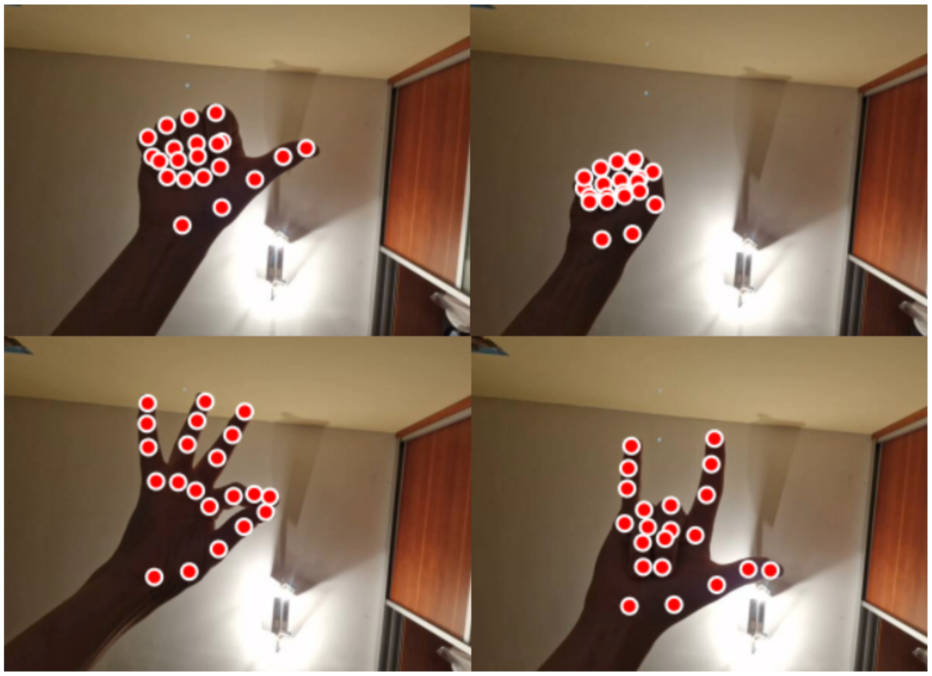

Fig. 3. Examples of gestures implemented.

4 Results

We successfully implemented a Rubik's Cube that could be controlled in real-time through hand gestures. The gesture recognition system achieved a high degree of accuracy, reliably interpreting user input even in dynamic scenarios. The cube rotations were smooth and visually engaging, contributing to an intuitive user interface that required minimal instruction.
List of possible gestures:

- fist - rotate the entire cube
- pinch - toggle axis
- rock n' roll - toggle layers
- swipe - rotate selected face
- thumbs up - scramble cube

To illustrate the project, figures and video demonstrations were included. A screenshot of the interface, shown in Figure 4, highlights the Rubik's Cube with its layers dynamically highlighted during a rotation. In addition, we have provided a video demonstration (available at my github repository) of how to use the application alongside written instructions.

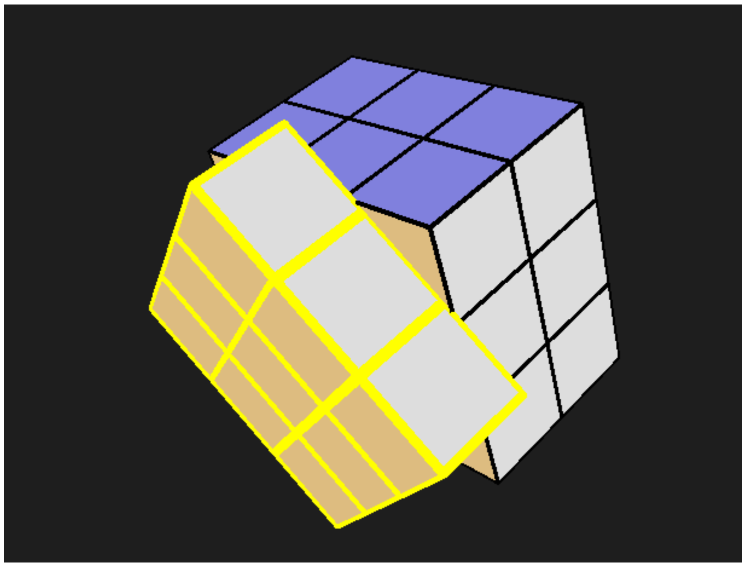

Fig. 4. Screenshot of the Rubik's Cube interface with highlighted layers during a rotation.

5 Discussion

This project underscores the potential of combining gesture recognition with interactive 3D visualization. In the context of interaction design, the system demonstrates how real-time hand gestures can enhance user engagement by providing a novel way to interact with a virtual object. The integration of computer vision and 3D graphics required careful consideration of several factors, including the consistency of gesture mappings and the clarity of visual feedback. Robust detection algorithms were critical to minimizing errors, especially in scenarios involving complex or rapid hand movements.

The Rubik's Cube can also be analyzed through the lens of shape grammars. Each move of the cube corresponds to a transformation in 3D space, and the addition of gesture controls introduces an interactive grammar that maps hand movements to these transformations. This framework highlights the structured, yet flexible nature of the system, where predefined rules govern interactions, but user inputs introduce variability and creativity.

Despite its success, the project faced several challenges and limitations. The accuracy of gesture detection was susceptible to fast movements and similar gestures. Distinguishing between similar gestures, such as a fist and a swipe, required extensive fine-tuning of the detection algorithms.

We also wanted to implement an automatic solver for our application, but the scope of implementing such a thing turned out to be too big to add to the current project. Implementing such a function would require integrating Rubik's cube solving algorithms such as Thistlethwaite's, Kociemba's, or Korf's algorithm[5].

Another addition to that was generation of music, which would depend on how close we were to solving the cube. This proved to be too ambitious functionality as it required implementation of music generation itself and figuring out how to calculate how close we are to the solved cube.

6 Mind Map

A comprehensive mind map of the project is included in Figure 5. This visualization captures the various aspects of the project, including technical components, interaction design principles, and potential applications.

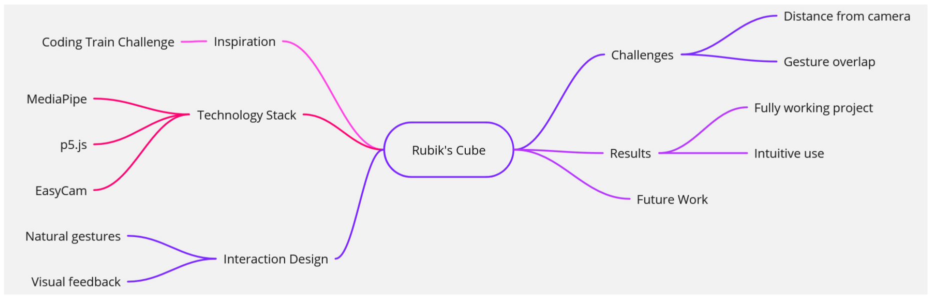

Fig. 5. Mind map illustrating the key components and ideas of the project.

7 Conclusion

This project demonstrates the feasibility and effectiveness of using hand gesture recognition for interactive 3D visualization. The Rubik's Cube served as a compelling case study, showcasing the potential of this approach for exploring interaction design, real-time computer vision, and visualization techniques. The intuitive controls and engaging interface of the system highlight the value of combining natural user input with advanced computational methods. Future work could extend this approach to other applications, such as educational tools, virtual reality experiences, or even more complex puzzles. These extensions could further explore the intersection of technology and user interaction, paving the way for innovative solutions in human-computer interaction. We could also implement two handed controls, which would allow us to create even more intuitive controls.

References

1. PREMYSL BEDNAREK. Web application for competitive rubik's cube solving. `https://is.muni.cz/th/cffcl/thesis_Archive.pdf`. Last accessed 2025/01/09.

2. EasyCam documentation. Easycam for p5.js. `https://github.com/freshfork/p5.EasyCam/blob/master/documentation/p5.easycam.docs.md`. Last accessed 2025/01/07.
3. MediaPipe documentation. Mediapipe hands documentation. `https://mediapipe.readthedocs.io/en/latest/solutions/hands.html`. Last accessed 2025/01/07.
4. p5.js Tutorials. p5.js tutorials. `https://p5js.org/tutorials/`. Last accessed 2025/01/07.
5. Ekta S. Toshniwal and Yogesh Golhar. Rubik's cube solver: A review. In *2019 9th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-19)*, pages 1–5, 2019.
6. Rubik's Cube / The Coding Train. Rubik's cube challenge. `https://thecodingtrain.com/challenges/142-rubiks-cube`. Last accessed 2025/01/07.