Abstract

The project is implementation of simulation of Rubik's Cube puzzle solver using only hand gestures. We have used Leap Motion device to capture hand gestures. This projects enables the user to solve a virtually simulated Rubik's Cube through Human Computer Interaction using Leap Motion Technology with which we can track hand gestures and map them to some computer programs.

Contents

1. Introduction

1.1 Leap Motion

The Leap Motion controller is a small USB peripheral device which is designed to be placed on a physical desktop, facing upward. It can also be mounted onto a virtual reality headset. Using two monochromatic IR cameras and three infrared LEDs, the device observes a roughly hemispherical area, to a distance of about 1 meter. The LEDs generate pattern-less IR light and the cameras generate almost 200 frames per second of reflected data. This is then sent through a USB cable to the host computer, where it is analyzed by the Leap Motion software using "complex maths" in a way that has not been disclosed by the company, in some way synthesizing 3D position data by comparing the 2D frames generated by the two cameras. In a 2013 study, the overall average accuracy of the controller was shown to be 0.7 millimeters.

The smaller observation area and higher resolution of the device differentiates the product from the Kinect, which is more suitable for whole-body tracking in a space the size of a living room. In a demonstration to CNET, the controller was shown to perform tasks such as navigating a website, using pinch-to-zoom gestures on maps, high-precision drawing, and manipulating complex 3D data visualizations.

Leap Motion initially distributed thousands of units to developers who are interested in creating applications for the device. The Leap Motion controller was first shipped in July 2013. In February 2016, Leap Motion released a major beta update to its core software. Dubbed Orion, the software is designed for hand tracking in virtual reality.

1.2 Rubik’s Cube

Rubik's Cube is a 3-D combination puzzle. In a classic Rubik's Cube, each of the six faces is covered by nine stickers, each of one of six solid colours: white, red, blue, orange, green, and yellow. In currently sold models, white is opposite yellow, blue is opposite green, and orange is opposite red, and the red, white and blue are arranged in that order in a clockwise arrangement. An internal pivot mechanism enables each face to turn independently, thus mixing up the colours. For the puzzle to be solved, each face must be returned to have only one colour. Similar puzzles have now been produced with various numbers of sides, dimensions, and stickers, not all of them by Rubik.

2.Merits and Limitations

2.1 Merits

- The application enables user to use gestures to solve Rubik's Cube which is simulated in his/her computer giving a sense of virtual reality.
- Controlling and Solving of Cube will be much easier using gesture control than solving it in computer by giving inputs through GUI or command line etc.

2.2 Limitations

- Since gesture detention is not accurate in leap motion sensor, mapping every gesture to an action becomes difficult. There is always an error involved.
- Since leap motion sensor looks an object like an eye, it is difficult to figure out number of fingers or accurate movement of fingers/hand in orthogonal plane.
- The cube simulation involves a lot of mathematics along with visualization. Redrawing of cube after every position change can sometimes lag a little or may not respond.

3. Model and Detailed Approach

3.1 Cube Simulation

Cube simulation is done using **python** programming.

Matplotlib along with **threading** has been used for continuous cube plotting on the canvas.

For creating the cube structure and implementing **Quaternion** cube rotation mathematics we have used **numpy** module of python.

3.1.1 Cube Specifications

Sticker representation

- Each face is represented by a length [5, 3] array: [v1, v2, v3, v4, v1]
- Each sticker is represented by a length [9, 3] array:
    - [v1a, v1b, v2a, v2b, v3a, v3b, v4a, v4b, v1a]

In both cases, the first point is repeated to close the polygon.

- Each face also has a centroid, with the face number appended at the end in order to sort correctly using lexsort.

The centroid is equal to sum_i[vi].

Colors are accounted for using color indices and a look-up table.

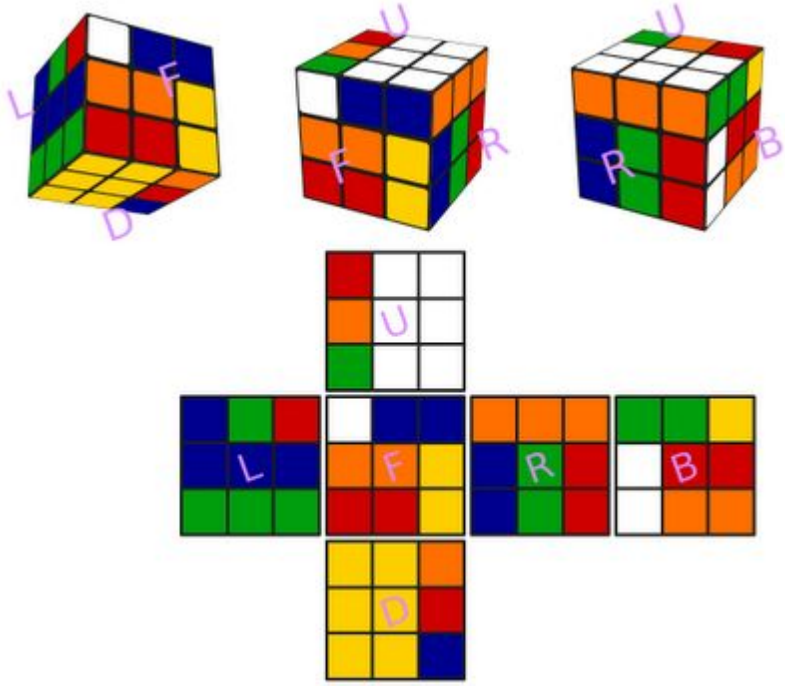
- With all faces in an NxNxN cube, then, we have three arrays:
    - centroids.shape = (6 * N * N, 4)
    - faces.shape = (6 * N * N, 5, 3)
    - stickers.shape = (6 * N * N, 9, 3)
    - colors.shape = (6 * N * N,)

The canonical order is found by doing ind = np.lexsort(centroids.T)

After any rotation, this can be used to quickly restore the cube to

canonical position. (Reset Cube)

Cube Diagram



The game will involve the following gestures/controls:

- Rotate the cube along any axis to change the viewing angle
- Rotate each visible face of the cube in the given direction with the required step size

3.2 Leap Interface

We have used the leap motion API for python in order to map gestures onto specific actions on the cube. The gestures we have used and the objects we have tracked for the purposes of our project are :

- Circle Gesture - Clockwise and Anticlockwise rotations to rotate faces of the cube in those directions
- Swipe Gesture - Rotate and reorient the cube
- Palm and palm face - To reposition and reorient the cube

Reorientation of Cube - A face up palm can be used to reorient the cube . We track the normal to the face of the palm in order to have the cube mimic the actions of the palm. For example , moving your palm right/left would rotate the cube along Y-axis and moving the palm up/down will rotate the cube along X-axis.

Turning to face the palm downwards will stop the cube from rotating/reorienting. This feature provides a large degree of control on the cube and helps to avoid accidental actions from moving the cube.

Rotating the faces - We use the circle gesture in order to rotate the various faces of the cube. Performing a circle gesture with different hand and under different conditions has different results on the cube. The following is a detailed list of the various actions to rotate the different faces -

- Right hand only - Clockwise and anticlockwise circle gestures to rotate the right face of the cube in those respective directions.
- Left hand only - Clockwise and anticlockwise circle gestures to rotate the left face of the cube in those respective directions.
- Both hand - Gestures on right hand to rotate the top face and gestures on the left hand rotate the bottom face of the cube.

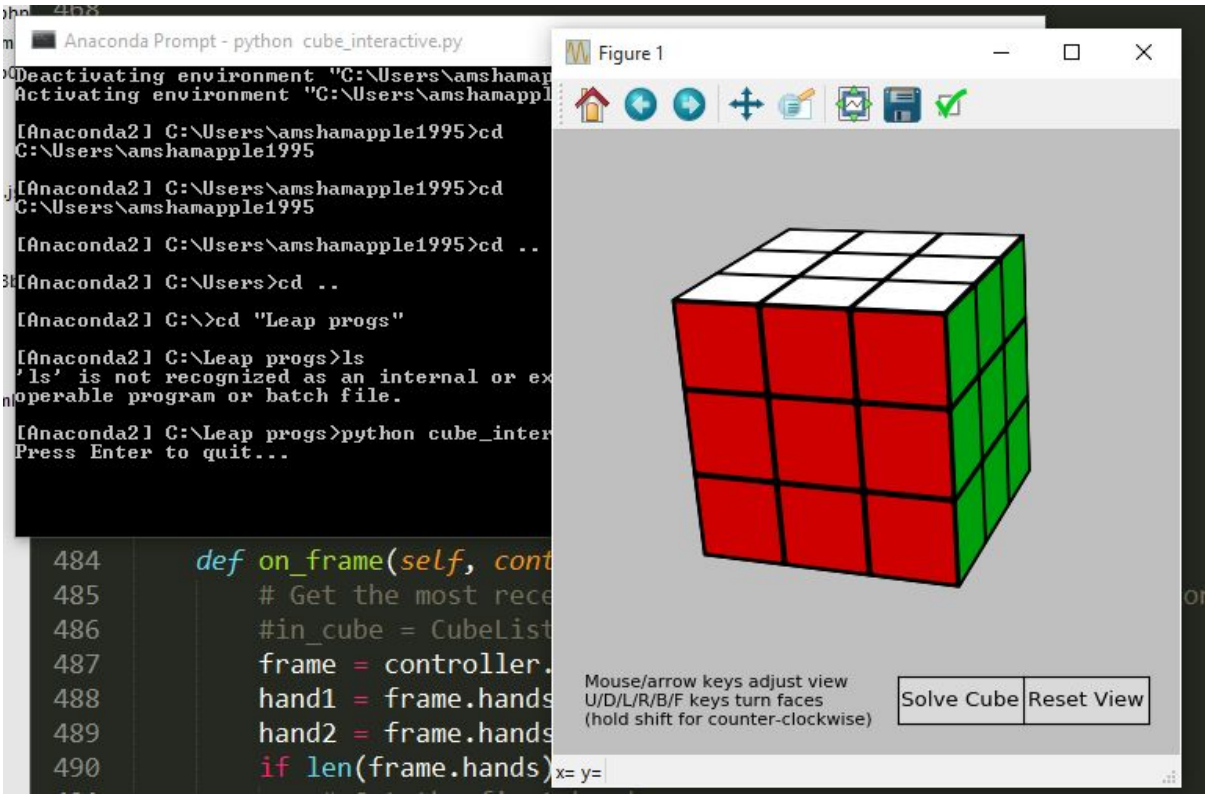
4. Results

The application is implemented using python and Leap API (Python). The code ran successfully on a laptop with core i5 processor and 8GB ram. There were no errors encountered and Leap Device was properly integrated with the application.

A little bit lag or unresponsiveness, due to either Leap or Simulation overhead, might occur sometimes.

The purpose of the project was to implement Leap Motion technology for Rubik's Cube game and give user a sense of gesture control.

A screenshot of the working application:



5. Future Work

There definitely is scope for improving this project. We envision the following things as possible expansions/improvements to this project -

- Implementing the project in VR - Using the leap VR it would be possible to provide the user with a life like and possibly a much better and more accurate cube solving experience.
- We can also use a database and some machine learning to make leap interpret the gestures more accurately.

Bibliography

- https://en.wikipedia.org/wiki/Leap_Motion
- https://en.wikipedia.org/wiki/Rubik%27s_Cube
- https://developer.leapmotion.com/