Hard Boiled Eggs

EECS 113
Faustino Aguirre (40765644)
Kian Bayati (59973782)
Michael Choi (72797038)
Fabian Garcia (78336321)

Rubik's Cube Solver Final Report

8th June 2017

OVERVIEW

The focus of this project is to create a Rubik's cube solver that analyzes a scrambled Rubik's cube and determines the steps required to return it to a solved state. Several key features of this project include: image-analysis, digital simulation, algorithm creation, and physical sensor implementation.

PROCESS

The program begins by accepting a picture of each face of the cube using the Logitech webcam. The pictures are taken by holding the camera on top of the cube and pressing any of the four pynq onboard buttons. The left onboard rgb light indicates the color of the next face to take a picture of. Magenta is used instead of orange, since the color orange is not available. The right cyan light indicates that the program is ready to accept the next picture. Once the six pictures are taken, the program runs a series of algorithms to determine the steps required to solve the cube. The user then uses the user interface to follow the instructions for solving the cube.

USER INTERFACE

The pynq board is the user interface:

RGB LED 4 (LD4): Direction to rotate the face of the cube 90 degrees. Green for clockwise. Red for counterclockwise.

RGB LED 5 (LD5): Color of face to rotate. These colors are based on the middle square of each face. Since the color orange is not supported, magenta is used for orange. This LED flashing indicates that the face should be rotated 180 degrees.

Switch 0 (SW0): Toggle this switch to move through the steps backward from current move.

Switch 1 (SW1): Toggle this switch to move through the steps automatically.

Buttons 0-3 (BTN0-3): Press on a button to move to next step. If moving automatically, pressing a button changes the speed at which the moves are shown. Button 3 is the slowest at 3 seconds per move, followed by 2 seconds, followed by 1 second, followed by half a second at button 0.

LEDs 0-3 (LD0-3): These LEDs show the current progress of the move list. As the user advances through the steps the LEDs light up from 3 to 0. When on the final step, these LEDs flash.

GOALS

In order to expand the feasibility of this project, the goals set to be accomplished will be broken down into smaller milestones. The first milestone is our primary object we believe we can achieve within five weeks, and the following are goals we hope to achieve if time allows.

1. Register a Rubik's cube onto the PYNQ board and print out the steps required to solve the Rubik's cube based on the F2L method.
2. Use a camera linked up to a PYNQ board in order to recognize and automatically register a Rubik's cube based on either a physical setup to detect color locations, or use a more advanced image recognition technique to extract the colors on the Rubik's cube.
3. Create a physical setup with acrylic components to solve the Rubik's cube using servos controlled by the PYNQ board.

GENERATING A SOLUTION

The solution algorithm uses the Fridrich method of solving the Rubik's cube. The four steps in this solution are solving a white cross on the bottom layer, solving the bottom two layers completely, correctly orienting the pieces of the last layer, and correctly permuting the pieces of the last layer. The average number of face rotations required to solve the Rubik's cube with our implementation of the algorithm is sixty-nine. To accomplish this, the program is comprised of several modules.

A python class called 'Cube' was designed that contains a 6x3x3 matrix to store the colors of the cube. It contains a method to rotate an individual face of the cube, mapping the colors of the cube to different locations accordingly. In addition, there are methods that return whether or not the cube is solved and convert an integer value to the corresponding color. On top of the 'Cube' class, there is the 'CubeSolver' class that holds algorithms from the Fridrich method. It ultimately takes in a 'Cube' object and returns the move list required to solve it. The steps are generated by looking at specific locations on this cube that are relevant to each step and manipulating them with simulated face rotations, applying a combination of the several stored algorithms based on the current positions.

IMAGE DETECTION

One feature of the cube solver is that it is able to scramble a solved Rubik's cube and generate a solution to that cube state. It can also take manual input of an unsolved cube as the starting point to the solution generator. This, however, can be tedious because the user would have to enter fifty-four colors; instead we decided to include a feature that streamlines this process by attaching a webcam to the PYNQ so that it can scan each face of the cube. A majority of the image processing was done with the help of the OpenCV Python libraries which are included in the PYNQ board.

The image detection algorithm works by taking an image of one of the cube's faces and converts it to a binary image, which can be done using a function provided in the OpenCV library: it must use a binary image to detect the corners of the cube so that it can detect each cell. Each of the corners are detected by diagonally searching through each pixel until the first black pixel is found, which is one of the corner pixels. This process is done for all four corners. One limitation of this technique is that it the cube cannot be tilted more than a certain angle from the direction of the webcam. Also, the binary image must not contain any dark pixels other than those contained

in the cube. Once the corners are located, the coordinates of each colored sticker in the image can be extrapolated. From here, the rgb values at each coordinate are compared to threshold values and converted to their corresponding colors.

RESULTS

The solving algorithm is guaranteed to generate a solution for any valid state of a Rubik's cube. Generating a solution is fast; from testing there has not been a case where it took longer than five seconds. The webcam is able to take images of a Rubik's cube under well-lit conditions and a completely white background. The pynq board correctly outputs the steps to solve the cube with the LEDs, and correctly handles input to traverse the solution move list.

FUTURE IMPROVEMENTS

With the current implementation of the program, if the chosen input is the camera and there is a problem reading the colors on the cube, the program crashes. This problem can be caused by lighting issues such as glare and low light, obstructions between the camera and the cube, and foreign objects being present in the picture. It can be fixed with more image processing that removes glare and accounts for obstructions or objects in the picture besides the cube. However, because RGB values do not simply scale uniformly under different conditions, we would have to consider how to properly compensate for them under unexpected circumstances. Given more time, this is something we could add to the program without great difficulty.

Another improvement would follow with physical implementation. The main issue with motor implementation is both cost and complexity. Because simple servos cannot be used due to angular limitations, stepper motors would be required. Unfortunately this would also require darlington arrays to control which would require four input pins each. The primary issue we had with physical implementation for both servo, continuous-rotation, and stepper motors were that the granularity of PWM signals did not satisfy our ability to control them properly, though it is likely due to software sided issues. Ultimately with enough experience, motor control should be achievable with our currently existing program.

One way that could improve the usability would be to correctly implement an led cube. This would be useful both to visually understand the moves taken, and also to easily check the correctness of the camera read. Some important notes to take would be to be mindful of current supply and also purchase easily solderable led strips. An important factor of success, at least from this experience, is ease of use. If led strips are cheap but small, they are not worth using.

SPECIFICATIONS & REQUIREMENTS

Item	Description	Vendor	Qty	Unit	Total
Xilinx PYNQ Board	Microcontroller to control all peripherals	Diligent	1	65.00	65.00
PYNQ Cables	Includes Ethernet Adapter, USB Cable	Diligent	1	40.00	40.00
Rubik's Cube	Self-Explanatory	Amazon	1	6.99	6.99
Logitech HD C270 Webcam	Object detection vision sensor	Amazon	1	21.50	21.50
Total					133.49