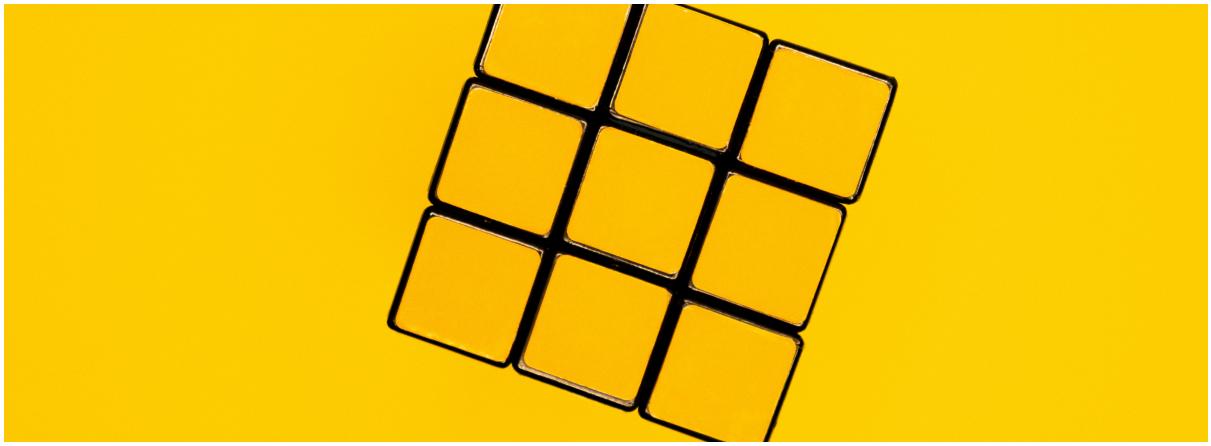


Paper

Abstract

Rubik's Cube, a 3-D combination puzzle invented by Ernő Rubik in 1974, is the World’s top-selling puzzle game and best-selling toy. Rubik's Cube is surprisingly complex, there are 43,252,003,274,489,856,000 permutations of a standard 3×3×3 Rubik's Cube. Although humans are impossible to memorize all permutations of a Rubik’s cube, humans can solve the puzzle as fast as 3.47 seconds. In recent years, a robot developed by MIT students can solve it in 0.38 seconds using an optimal algorithm.

In this project, I developed a Rubik’s cube-solving robot base on MCU from scratch. The robot has 4 stepper motors and 4 Servo motors which are the actuators of the robot, also a camera is included to observe each color arrangement of the face of the cube in a scrambled state sequentially. The image that the camera observed will be sent through a serial interface to the computer. I had implemented a solving algorithm that will take the cube state as an input, and the sequence of moves that solve the cube as an output. The sequence of moves is then translated into a series of instructions and send to the MCU. The stepper motors and the Servo motors then cooperate and twist the cube according to the instructions, fully solve the cube.

Contents:

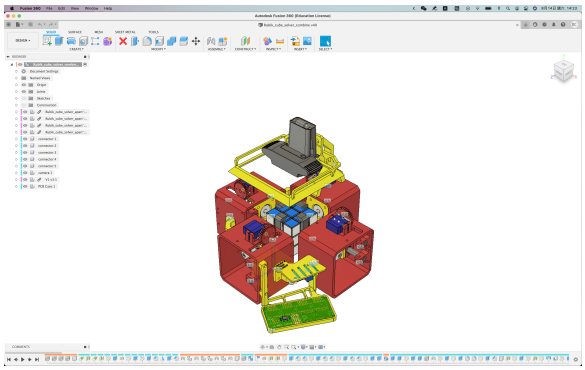
Motivation

In recent years, computer vision becomes more popular. Computer vision has a lot of applications, such as object detection, event detection, video tracking, and color recognition. To gain more experience, I decide to develop a robot that solves a Rubik’s Cube using Computer Vision.

The goal is to fully solve a Rubik’s cube by twisting it using clamps attached to stepper and Servo motors. We will use a camera to capture each face of the cube. A computer will then be used, using the computer vision algorithm, to process these images and find out the color of the 9 stickers in each face. Once the color states of the cube are determined, the computer can then compute the sequence of moves that fully solve the cube.

Introduction

I design the robot using Fusion 360 and Eagle. This robot consists of 4 stepper and Servo motors, 4 pairs of 3d-printed clamp and frame, a camera, and a PCB board included a microcontroller and 4 stepper motor drivers



The design of the robot ^

To know the initial state of the cube, I use a camera to capture the color of the cube. But the camera is only allowed to capture a single face of the cube each time. So, by applying a sequence of movements to the cube, the camera is now able to capture all the faces of the cube. After capturing, the camera is going to send those images to the computer through a USB port.

The computer will then detect the position and orientation of each face from all 6 images. By using these information, the computer can find out the position and area of each sticker of the cube in those images. The computer can

now extract the color of each sticker, and determine the color of each sticker using computer vision.

Once the initial state of the cube is found, the computer computes the sequence of moves that solves the cube using Thistlethwaite's algorithm. The sequence of moves will be converted into a series of Rubik's Cube Notation.

After the solution to a scrambled cube is found, the computer will send the entire notation that solved the cube to the microcontroller. Then, the microcontroller is going to translate these notations into a series of stepper and Servo motors movements. Eventually, these Servo motors will cooperate with the stepper motors to twist the cube according to the series of notations. Fully solve the cube.

Materials

- 4 stepper and Servo motors
- 4 3d-printed clamps attached to the stepper motors
- An Arduino nano and 4 stepper motor drivers
- A DC transformer
- A fill light

Designs

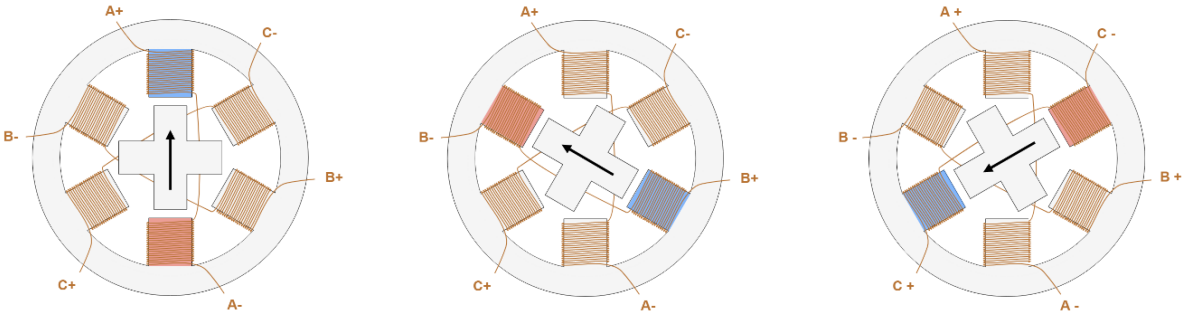
Hardware

Actuators

To physically twist and rotate the cube, I use 4 NEMA 17 stepper motors that use a clamp to clip and twist 4 individual faces of the cube. I had considered using DC motors and brushless motors, but both of them can't know their physical rotated-angle without an encoder, so they can't twist the cube precisely. Although Servo motors are able to know their rotated-angle, most Servo motors can only rotate from 0 degrees to 180 degrees. Finally, I decided to use stepper motors, because stepper motors are continuous(they can rotate in one direction infinitely). They also know their relative rotated-angle. Therefore, a stepper motor can perfectly finish its tasks.

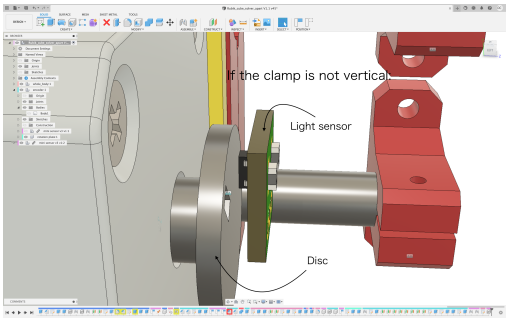
Stepper motors operate using a coil to attract the permanent magnets that encircle the center axle. By energizing the coils, the axle of the motor can rotate precisely by 1.8 degrees. (Stepper motors take 200 steps to make a complete rotation, so its resolution is 1.8 degree = (360° /200 steps))

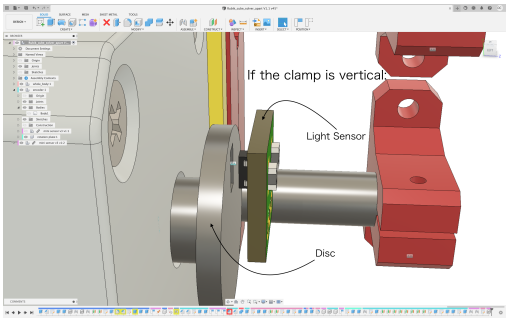
Stepper motors are move in steps. For example, a stepper motor with 200 steps resolution will rotate 1.8 degrees angle for one step (360° / 200 steps).

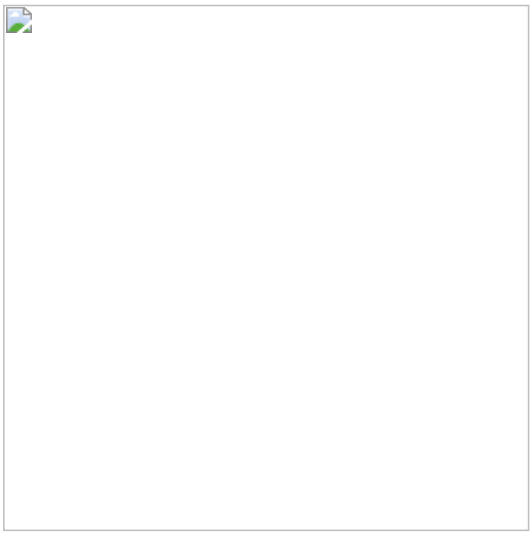


Simplified stepper motor rotates 3 steps.

Stepper motor is an Open Loop Control System, we can't know its position when start-up. And you can't always make sure the clamp is vertical every time the robot start-up. To make sure the motor knows if the clamp is vertical, I design a device that has a disc with a hole and a light sensor. The disc is connected to the axle of the stepper motor and the light sensor is facing the disc. When the clamp is vertical, the hole of the disc is right at the top and facing the light sensor. By reading the value of the light sensor, we can know that the light sensor is facing the hole or not. Therefore, we can also know whether the clamp is vertical or not.







I found a problem when the clamp twisting the cube. If one of the clamps rotates to 180 degrees, the clamps on its left and right can't twist the cube because the clamp that is horizontal blocks their rotation. So there is also a Servo motor that is responsible to separate the clamp from the cube so that all clamps can rotate freely.

Other than the stepper motor, Servo motor is a Closed Loop control system. When the Servo motor is moving, a feedback of position given by the potentiometer(Fig 1) of the Servo motor is sent to the Servo motor control board to ensure if it rotates to the expected position.

I use Rack and pinion mechanics to convert the rotational motion of the Servo motor

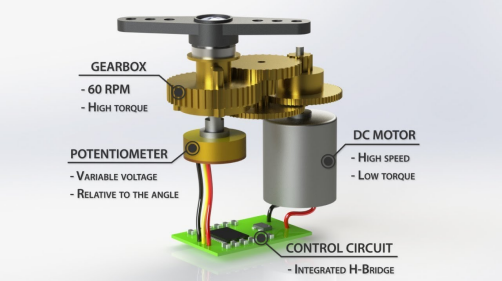
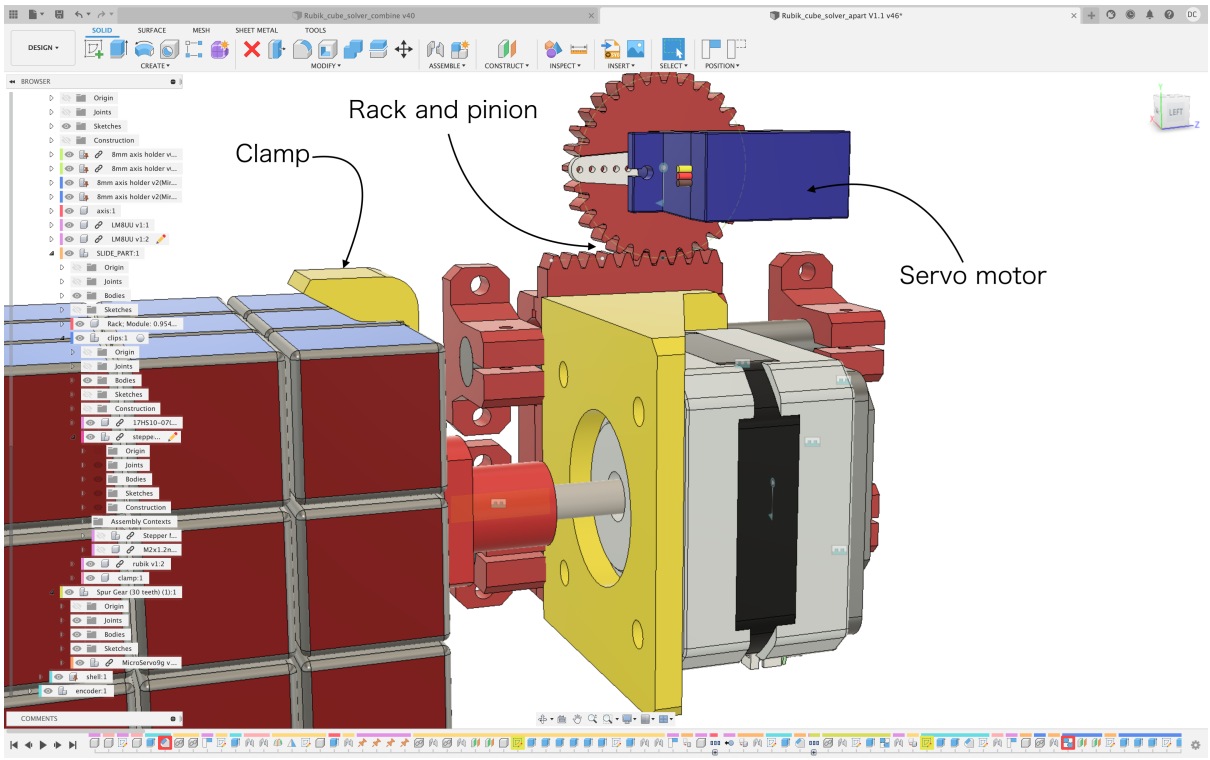


Fig 1

into linear motion to separate the clamp from the cube.



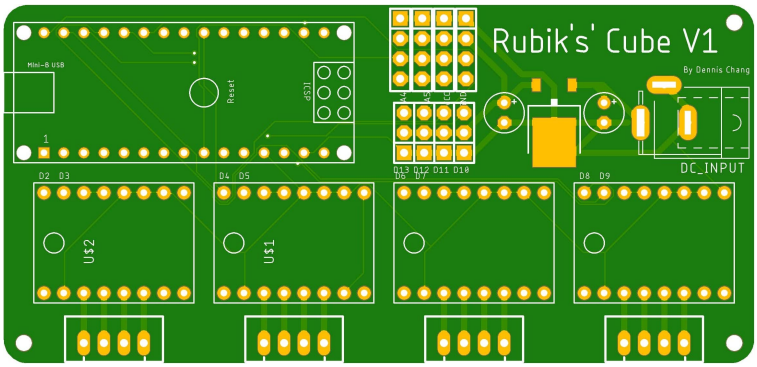
Camera

To know the initial state of a Rubik's cube, I use a camera to capture the color of each face of the Rubik's cube. In this project, I chose Logitech HD Pro Webcam C920 as the camera because it has an HD resolution and an autofocus function. Also, the camera allows you to adjust the brightness, sharpness, contrast, and saturation of the image. By adjusting these values, the images will become sharper and easier for the computer to distinguish colors.

When I try to distinguish color from the image that the camera sends to the computer, I find that if the ambient light projected to the cube is too weak, the color recognition program will be extremely inaccurate. To make sure the light that projects to the cube is stable, I decide to add an artificial light source to maintain the brightness of the image. Therefore, I use 4 LED strings to supply light in 4 directions, so the brightness anywhere on the face of the cube is the same.

Electronics

Instead of using a commercial CNC control board that can control stepper motors easily, I am going to design my robot controller PCB that can control 4 stepper motors and 4 Servo motors of my robot because most commercial CNC control boards can't control 2 or more Servo motors. Also, there is a lot of limitations of programming these commercial PCB board, so I decided to make my own control board.



Robot control board

- Microcontroller

In order to control the whole robot, I decided to use Arduino nano to control the whole system because it is smaller than Arduino UNO and easy to program. Arduino Nano has a built-in Mini USB jack, so it can communicate with the computer through a Serial port with the Mini USB jack.

- Stepper motor Control

Although stepper motors have a lot of advantages, unfortunately, controlling stepper motors often require sufficient and controlled energy for phases in a precise sequence, so a driver is often needed. Due to this, I plan to use 4 A4988 stepper motor drivers to control all of my stepper motors of the robot.

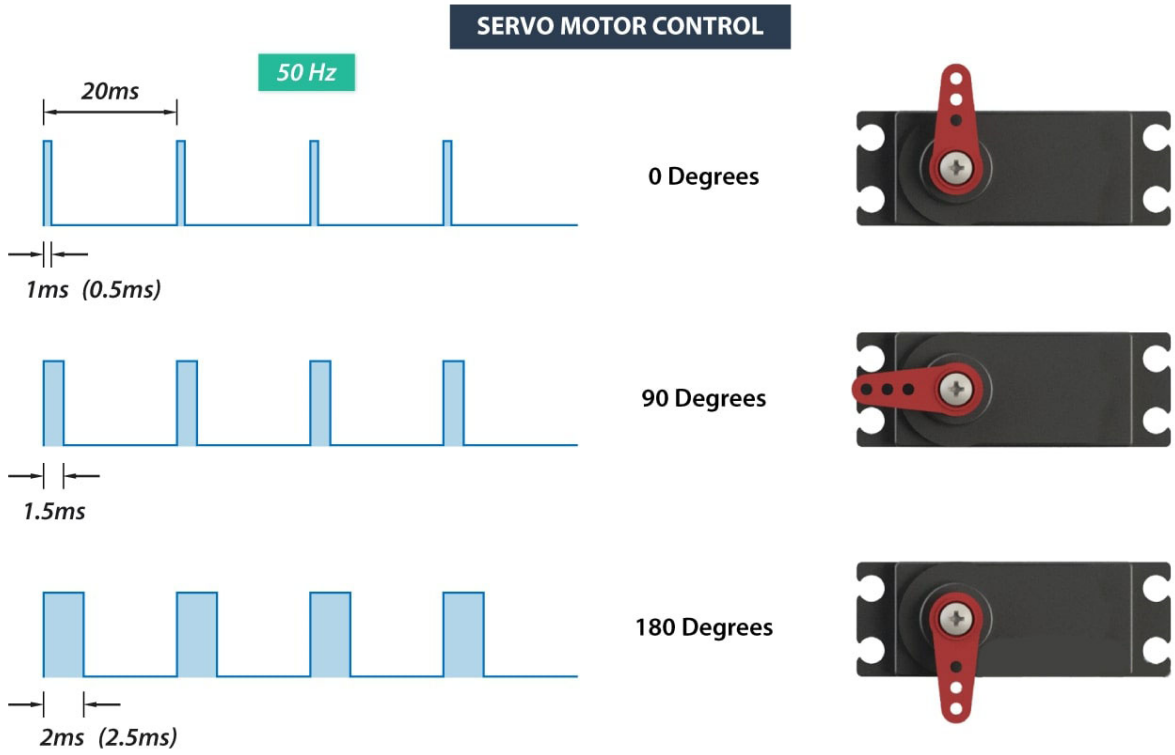
Each A4988 motor driver required 2 control signal pins (step, direction) and 3 setup signal pins(MS1, MS2, MS3). The step and direction pins are used to control the movement of the stepper motors. The direction pin controls the rotate direction of the motor, and we can control the movement of a step of the motor by sending a pulse to the step pin. The MS1, MS2, MS3 pins are used to control the step of the stepper motor. By changing the input state of these pins to logic low or logic high, you can select one of the five steps resolution according to the table(Fig 2).

MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Halft Step
LOW	HIGH	LOW	Quarter Step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth Step

Fig 2

- Servo motors Control

A Servo motor can be controlled by sending a series of pulses through the signal pin. By changing the width of these pulses, we can control the Servo motor rotates to different angles.



How pulses with different width control the Servo motor

Programming

Computer visions

In order to determine the color arrangement of each of the image that captured by the camera precisely, I implemented a method that can found out the difference of two colors.

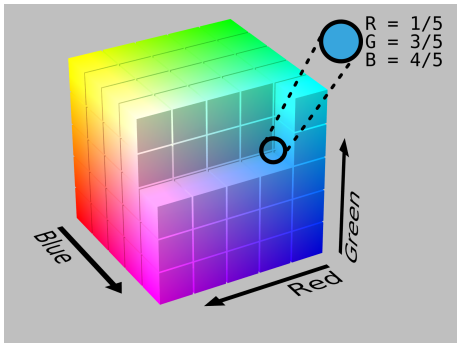
A color model is a visualization that depicts the color spectrum as a multidimensional model. There are some commonly used color models in the world. Most modern color models have 3 dimensions (like RGB and HSV), while other models have more dimensions (like CMYK), and I am going to compare these different color models and found out which of them are more reliable in this project.

- RGB

  RGB is a 3-dimensional color model. RGB color model represents a color by mixing three different colors (Red, Green, and Blue). Although RGB color model is commonly used in our daily life, RGB color model is a non-uniform color model, which means the Euclidian distance of two colors represented by the RGB color model does not correspond to the color difference perceived by humans.

  Euclidian distance:

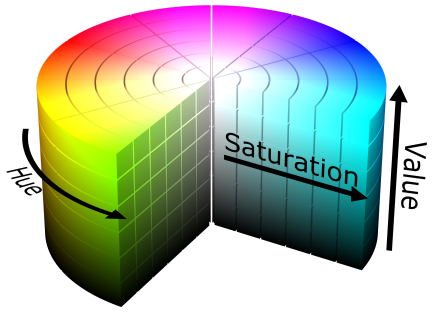
  $$distance = \sqrt{((R2-R1)^2 + (G2-G1)^2 + (B2-B1)^2)}$$



- HSV

  HSV is a cylindrical color model that remaps the RGB primary colors into dimensions that are easier for humans to understand.

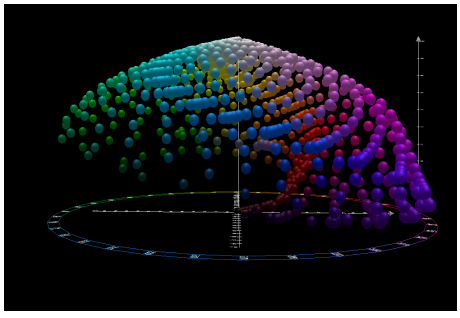
  In HSV color model, we can represent a color with 3 variables (Hue, Saturation, Value). Hue controls the angle of the color on the RGB color circle; Saturation controls the amount of color used; Value controls the brightness of the color.



HSV (cylindrical color model)

- CIELAB

  CIELAB color model is three-dimensional, you can represent a color with 3 variables in CIELAB color model while L stands for perceptual lightness, and A and B stand for the four unique colors of human vision: red, green, blue, and yellow. In short, CIELAB is designed to approximate human vision, so it can describe a color more accurate than RGB and HSV color model can.



CIELAB color space top view

According to the comparison, I think using CIELAB color model in the Computer Vision method has a lot of advantages, so I plan to use CIELAB color model to represent a color in the Computer Vision method.

When the computer wants to distinguish which color (orange, white, blue, red, yellow, green) of a sticker is, the most important thing is to find out the distance between the sticker pixel color and the reference colors, so the computer can then compare the distance between these reference colors, and distinguish the color. In order to get the color difference between two colors reliably, I use a CIEDE2000 ΔE method instead of the Euclidean distance method because the CIEDE2000 ΔE method is more reliable for human vision.

Given a reference color (L*1, a*1, b*1) and another color (L*2, a*2, b*2), the CIEDE2000 difference of these 2 colors is:

$$\Delta E_{94}^* = \sqrt{\left(\frac{\Delta L^*}{k_L S_L}\right)^2 + \left(\frac{\Delta C_{ab}^*}{k_C S_C}\right)^2 + \left(\frac{\Delta H_{ab}^*}{k_H S_H}\right)^2}$$

CIEDE2000 ΔE formula
https://en.wikipedia.org/wiki/Color_difference

Where:

$$\begin{aligned}
\Delta L^* &= L_1^* - L_2^* \\
C_1^* &= \sqrt{{a_1^*}^2 + {b_1^*}^2} \\
C_2^* &= \sqrt{{a_2^*}^2 + {b_2^*}^2} \\
\Delta C_{ab}^* &= C_1^* - C_2^* \\
\Delta H_{ab}^* &= \sqrt{{\Delta E_{ab}^*}^2 - \Delta {L^*}^2 - {\Delta C_{ab}^*}^2} \qquad = \sqrt{\Delta {a^*}^2 + \Delta {b^*}^2 - {\Delta C_{ab}^*}^2} \\
\Delta a^* &= a_1^* - a_2^* \\
\Delta b^* &= b_1^* - b_2^* \\
S_L &= 1 \\
S_C &= 1 + K_1 C_1^* \\
S_H &= 1 + K_2 C_1^*
\end{aligned}$$

CIEDE2000 ΔE formula
https://en.wikipedia.org/wiki/Color_difference

Because of the rotation of the cube, the cube position may always shift when scanning the cube, so I have to frequently adjust the cube detect area of the camera manually before scanning it, and I think that adjusting the cube manually is annoying, especially when I debugging the program.

Then, I try to implement a cube position tracking program in the solver, so that the solver can adjust the detect area automatically.

The tracking program is actually dum. First of all, the program is going to make the dimmer place of the image more dimmer; the brighter place brighter.

The program can then find out the position of the centerpiece of the cube by comparing the pixel of the filtered image. The program can then find out the position of other pieces.

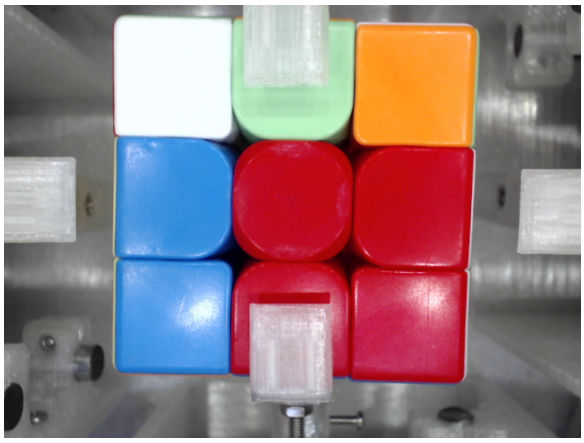
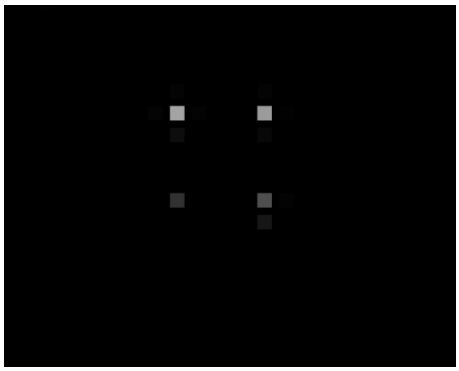


Image before filter



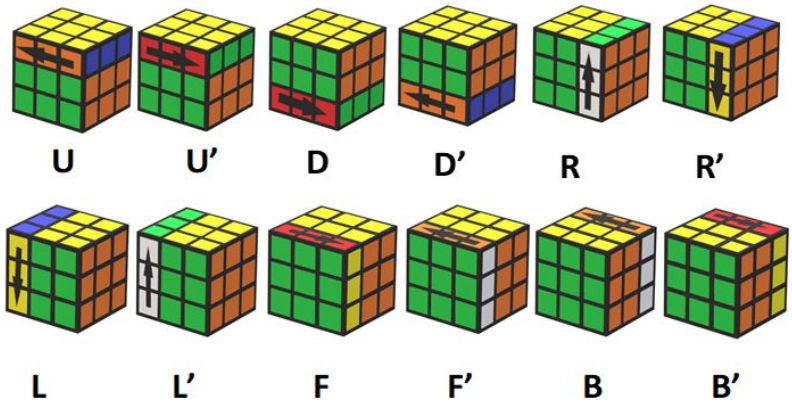
Filtered image

Thistlethwaite's algorithm

After finding the state of the cube, the computer will then solve the cube using Thistlethwaite's algorithm. Thistlethwaite's algorithm is a famous algorithm devised by Morwen B. Thistlethwaite.

Before I explain how the algorithm works, we first have to know how do people represent a twist of a cube. People usually use a notation called Rubik's Cube Notation to represent a twist of a cube. The twist of a Rubik's Cube is classified into three categories of Notations (clockwise twist, counterclockwise twist, double clockwise twist).

- The notation with only one letter refers to a clockwise face twist in 90 degrees (quarter turn). "F" refers to a clockwise turn of the Front face in 90 degrees.
- The notation with one letter followed by an apostrophe (') refers to a counterclockwise face twist in 90 degrees (quarter turn). "D'" refers to a counterclockwise turn of the Down face in 90 degrees.
- The notation with one letter followed by the number 2 refers to a clockwise face twist in 180 degrees (double turn). "U2" refers to a clockwise turn of the Up face in 180 degrees.



Thistlethwaite algorithm solves a cube by twisting the cube by a certain set of moves. When the cube reaches a specific state, the cube will "move" from one "group" to another. As the cube reaches another "group", the set of moves of the cube will change and the cube will be easier to solve computationally.

The groups are:

- Group 0:

$$G_0 = \langle L, R, F, B, U, D \rangle$$

Required Cube state: This group contains all possible states of the Rubik's Cube.

- Group 1:

$$G_1 = \langle L, R, F, B, U^2, D^2 \rangle$$

Required Cube state: All 12 edge pieces are correctly oriented.

After reaching Group 1, The cube can then be solved without using U and D turns.

- Group 2:

$$G_2 = \langle L, R, F^2, B^2, U^2, D^2 \rangle$$

Required Cube state: All corner pieces are correctly oriented. Also, four of the edge pieces are moved to the correct slice: the front-up, front-down, back-up, and back-down edge pieces are placed in the M slice.

After reaching Group 2, The cube can then be solved without using F, B, U, and D moves.

- Group 3:

$$G_3 = \langle L^2, R^2, F^2, B^2, U^2, D^2 \rangle$$

Required Cube state: All corner pieces are moved to the correct orbit. (A corner is in the correct orbit if it can be moved to its home position with only 180-degree twists.) Also, each edge piece is moved to its home slice.

At that point, the cube can be solved with only 180 degrees turn.

- Group 4:

$$G_4 = \{1\}$$

The final group requires only one state, which is a solved cube.

Firmware

In this project, I use Arduino nano board as my microcontroller to control the stepper motors and servo motors. I implement an Arduino stepper motor driver library in the Arduino board to control the movement of the Stepper motors. For the Servo motors, I use an Arduino build-in Servo library to control them.

There is a lot of ways to communicate between the computer and the Arduino board, such as I2C, Serial communication, and SPI. I decided to use the Serial communication way because it is widely used in the Arduino board (Every Arduino board has its own Serial port). For an Arduino Nano board, it has a build-in mini-USB jack

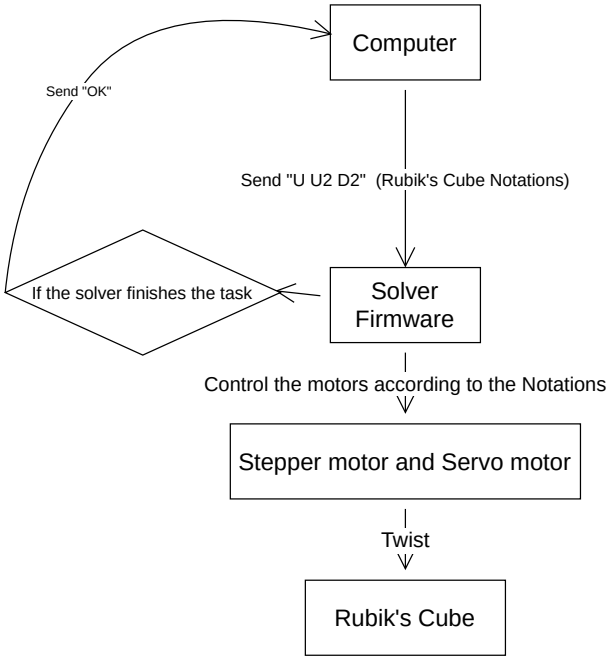
Here is a case of communication:

The Computer aims to twist the Cube in U U2 D2:

Step 1: The computer sends a command("U U2 D2") to the Solver through the Serial interface.

Step 2: The Solver receives the command and controls the Stepper motors and the Servo motors to twist the Cube according to the command.

Step 3: After twisting the cube, the Solver will send a signal to the computer. Let the computer knows that it has done the task.



Challenges and Conclusion

After assembled the Solver, there are some problems that I have experienced. When designing this robot, I aim to use a magnetic encoder to monitor the position of the robot. After I fully assembled the Solver, I try to read the value of the magnetic encoders by connecting them in parallel using I2C, but then I found that these magnetic encoders have the same I2C addresses, and I can't change the addresses. Therefore, the microcontroller can't read the values of them using I2C, so I have redesigned the robot, and I use a light sensor as the position monitoring sensor instead of a magnetic sensor.

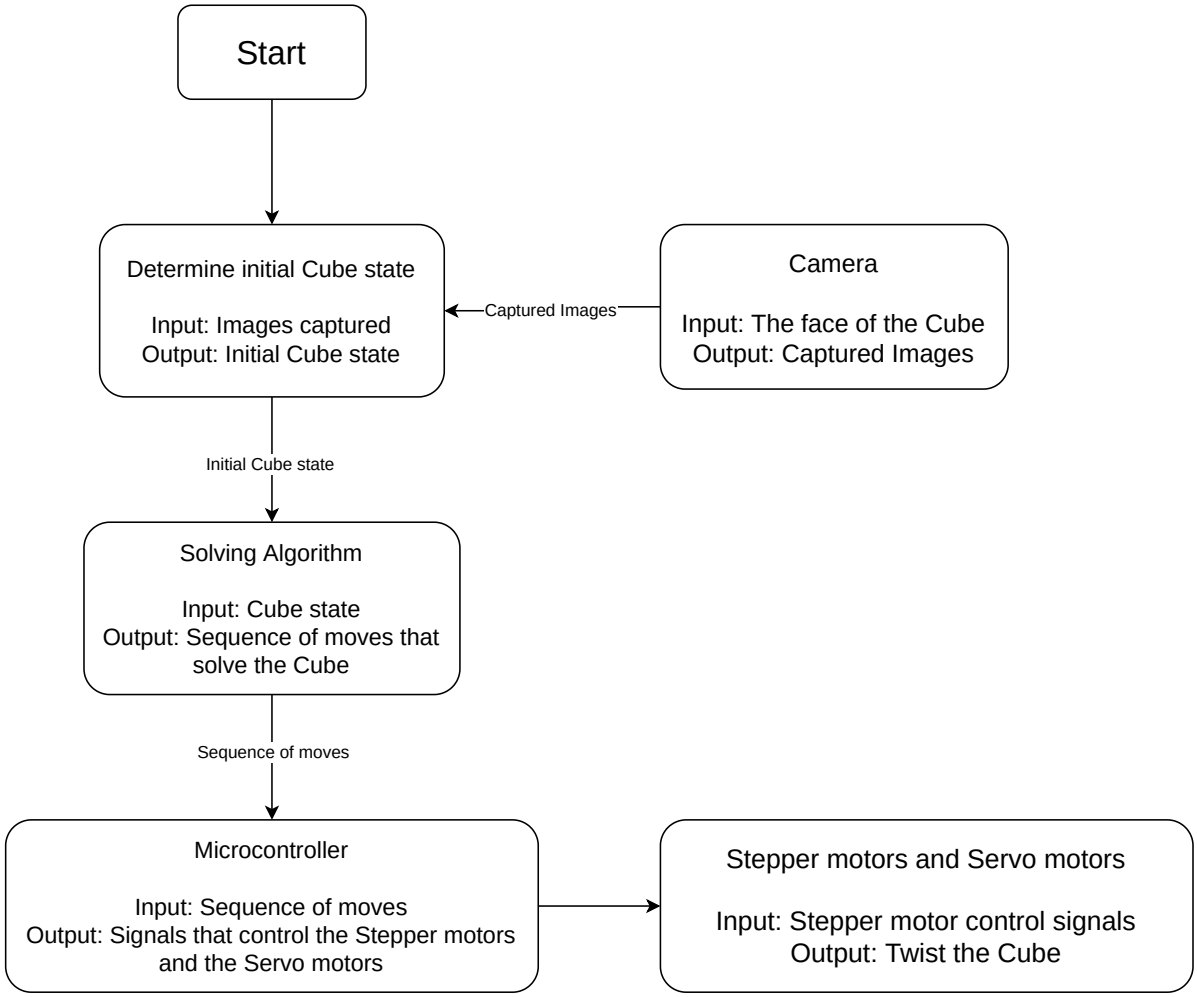
There are also some challenges that I have gone through when I programming the robot, such as the inaccuracy of the color detection. To fix the issue, I add an artificial light source to stabilize the light condition of the cube.

The robot also experienced some issues when the stepper motor rotate. Because the stepper motor provides no feedback to the microcontroller, the stepper motor can't affirm a perfect 90 degrees twist is executed. Also, there are gaps between the clamp and the Cube, so there will be some degrees of error when twisting the cube. As a result, slight errors accumulated in the alignment of the Cube after a twist. Because of the alignment of the Cube, the rotation of the stepper motor may be stuck, but the stepper motor can't realize that it is stuck, and it will keep executing the following twists.

Were I to continue work on this project, the following attributes would be changed:

- A Feedback would be added to affirm a perfect 90 degrees twist is executed
- Reduce the size of the Robot.
- Design an adjustable clamp, so that the solver can fit any size of 3*3 Rubik's Cube.

Appendix – Workflow of the Rubik's Cube solving robot:



Rubik's Cube Solving Robot test:
https://youtu.be/ZOZHCysa5HY

Reference

- Color models and color spaces https://programmingdesignsystems.com/color/color-models-and-color-spaces/
- Color difference - Wikipedia https://en.wikipedia.org/wiki/Color_difference
- How to control stepper motor with A4988 driver and Arduino https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/
- jeswiezy_Project_Final_Report http://web.mit.edu/6.111/www/f2017/projects/jeswiezy_Project_Final_Report.pdf
- Thistlethwaite's 52-move algorithm https://www.jaapsch.net/puzzles/thistle.htm
- Rubik’s Cube Reconstruction from Single View for Service Robots https://www.ia.pw.edu.pl/~wkasprza/PAP/ICCVG06.pdf

- All About Stepper Motors
  https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor