Rubik’s Cube Documentation

Useful functions:

1. single\_face\_scanner:
   1. Arguments:
      1. Input image array
   2. Output:
      1. Single cube mapping
   3. Purpose: This function is made to isolate the colors of the Rubik’s cube and

place the colors into a 3x3 matrix.

1. single\_face\_rotation
   1. Arguments:
      1. face: [front, back, left, right]
      2. angle: [1: 90 clockwise, 2: 180 clockwise, -1: 90 clockwise]
      3. pin\_dict: dictionary of device pins (prepared for you in the main loop)
   2. Purpose: Performs a simple face rotation i.e., a rotation of a face that

has a designated hand.

1. back\_off:
   1. Arguments:
      1. face: [front, back, left, right]
      2. angle: [1: 90 clockwise, 2: 180 clockwise, -1: 90 clockwise]
      3. pin\_dict: dictionary of device pins (prepared for you in the main loop)
   2. Purpose: Routine that reverses a simple rotation without rotating the cube. The reason for the backoff is that we do not have 360 rotation. So, we need the arms to return to their original position after each move.
2. special\_face\_rotation
   1. Arguments:
      1. face: [up, down]
      2. angle: [1: 90 clockwise, 2: 180 clockwise, -1: 90 clockwise]
      3. pin\_dict: dictionary of device pins (prepared for you in the main loop)
   2. Purpose: Performs a special face rotation i.e., a rotation of a face that

does not have a designated hand.

1. full\_cube\_rotation
   1. Arguments:
      1. face: [x, y]
      2. angle: [1: 90 (one way), -1: 90 (the opposite way)] (not sure about directions, test to see)
      3. pin\_dict: dictionary of device pins (prepared for you in the main loop)
   2. Purpose: Performs a full cube rotation. I am not sure of the directions so test and see. I’ll just call it back and forth. This is made specifically for scanning and performing special face rotations.

Main Pipeline:

1. Motor pin setup

Text

Description automatically generated

These are the gpio pins used for each motor. The comments describe what each pin is. I expect Adi to be used to this and if any pins need to be changed, he is in charge of that. They are set up in the following format:

Diagram

Description automatically generated

A picture containing text, indoor

Description automatically generated

Each of the pin variables are then stored in a dictionary for easy access in the functions. This dictionary is known as pin\_dict. **It should not be touched**. Pin changes should happen at the individual variables in the previous image. The indexing for pin\_dict is made to be the same as the naming convention of the pins. That is:

pin\_dict[‘front’][1][0] => front10

1. Important variables

Timeline

Description automatically generated with medium confidence

scanning\_loop: This is an array of full cube moves made to take all the pictures of the cube. Ideally, it should take all the pictures necessary and in order, and then return the cube back to its original position. Each move is an array of the following format:

* + - * Index[0] => axis (x or y)
      * Index[1] => direction(1: one way, -1: the other way)
      * Index[2] => take picture(0: don’t take, 1: take)]

You guys can mess with this but with the way I set it up the red middle piece should be at the top and the white middle piece should be towards the back hand. If y’all have issues with this scanning loop pattern, call me.

colors: The output of the single\_face\_scanner function is a matrix of numbers. This dictionary converts those numbers into proper inputs for the Rubik’s cube.

cube: This variable will hold the cube information in the form that the cube solver is expecting.

cam: This is the cv2 webcam interface object. It is used to take pictures of the cube.

1. Scan loop

Text

Description automatically generated

So, for each scan move, three things happen:

1. A full cube move occur to prepare for the photograph.
2. A picture is taken (it sometimes is not since some moves are strictly to return the cube to its original position
3. A back-off is done to return the arm to its original position.

This happens for each move in the scanning loop array.

1. Implementing Solution:

Text

Description automatically generated

Please delete the highlighted line of code. It was for testing purposes.

After the scanning loop you should get a cube variable with the required cube info for the solver. Plug that into the solver function (second line) and you get a list of moves in the variable solution.

Looping through each move

1. If statement 1: turns every move of form “F” to “F1”
2. If statement 2 does not do anything so, take it out
3. If statement 3: checks to see if move is a simple face move by checking the first index of the string. It then performs the move accordingly (if move is of form “ F’ “, it performs an anti-clockwise move)
4. If statement 4: checks to see if move is a special face move by checking the first index. It them performs the move in similar fashion.