Automated Rubik's Cube Recognition

Justin Ng
Department of Electrical Engineering
Stanford University
Email: justinn@stanford.edu

***Abstract*—In this paper, we aim to design and implement an algorithm that will quickly and accurately identify the current state of a Rubik's cube. Techniques used include lighting compensation, feature detection, and pixel classification.**

I. Introduction

The Rubik's Cube is a 3-D mechanical puzzle invented by Erno Rubik in 1974. In a classic Rubik's Cube, each of the size faces is covered by nine stickers, each of one of six solid colors: blue, green, orange, red, white and yellow. A pivot mechanism enables each face to turn independently, thus mixing up the colors. [1] There are approximately forty-three quintillion possible permutations of the Rubik's Cube, so solving the Rubik's Cube can be a daunting task. Therefore, it would be helpful to have a program that can read in the current state of the Rubik's Cube and suggest an algorithm for solving it.

For the scope of this project, we will focus on producing an algorithm that can automatically identify the current state of the Rubik's Cube. Our objective is to correctly identify the Rubik's cube using only two images, each showing three different sides of the Rubik's Cube. We will then display the resulting identified state of the Rubik's cube.

II. Overview of the Algorithm

Our algorithm to identify the state of Rubik's Cube consists of three steps. First, we have to detect the three faces of the Rubik's Cube from each of the two input image. Then, we will identify the colors for each of the faces on the Rubik's Cube. Finally, we combine the results from the two images determine the current state of the Rubik's Cube.

A. Dectecting the Cube

This step is to create a mask for the locations of the faces on the Rubik's cube from an input RGB image. We apply a high pass filter to each of the three color channels to help enhance the black edges separating the faces on the cube. Then we convert the filtered RGB image into grayscale and apply a threshold to isolate the faces of the cube.

We check the regions that were created from the thresholding, discarding the regions that are not an acceptable size, do not have have an acceptable shape (eccentricity), or are not solid. At this point, we should be left with only faces from the cube. However, some of the faces on the edge may have the black border visible and thus were not detected. We will need to recover these faces by estimating their location from the detected faces.

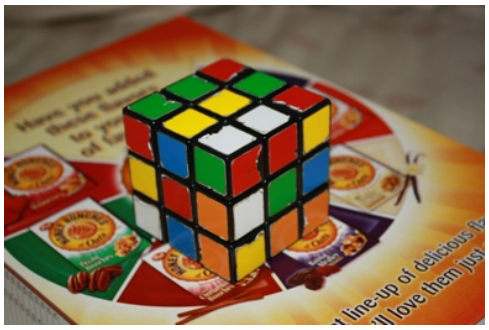
Fig. 1. The Original RGB Image

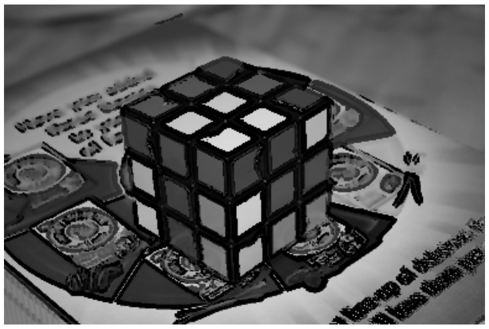
Fig. 2. Grayscale Image with High Pass Filtering in Color Channels

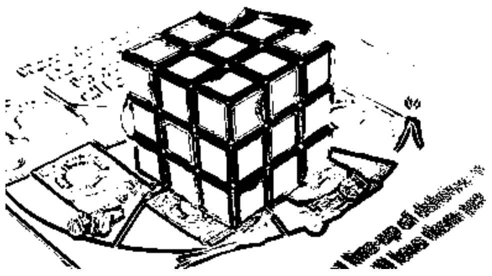
Fig. 3. Grayscale Image After Thresholding

We divide the detected faces into three groups based on orientation. This will help us help us identify which side each of the faces belong to. Finally, for each of the sides, we fill in any potential missing sides to create the mask for the image.

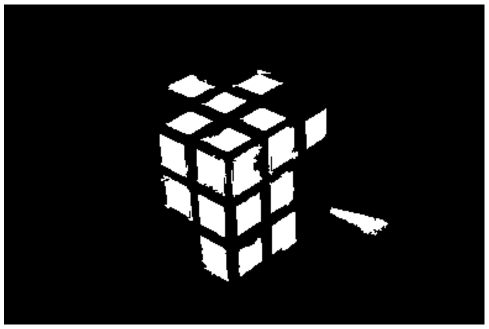
Fig. 4. Image After Size, Eccentricity and Solidity Filtering

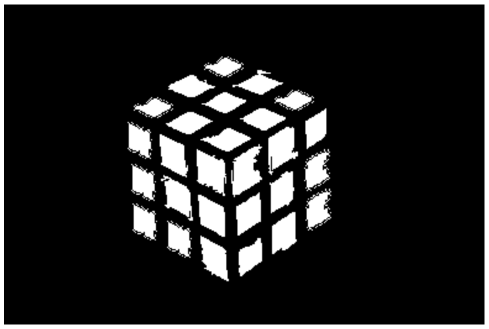
Fig. 5. Image With Missing Faces Filled In

B. Identifying the Colors of the Faces

Once we have the mask for the image, we transform the image from RBG space into YUV space. The pixel color components are normalized with respect to the pixel's intensity according to the following equations. [2]

$$\begin{aligned} Y'_p &= 128 \\ U'_p &= (U_p - 128) * k_p + 128 \\ V'_p &= (V_p - 128) * k_p + 128 \end{aligned}$$

where

$$k_p = \begin{cases} \frac{Y'_p}{Y_p} & \text{for } \mathrm{Y}_p > 128 \\ \frac{256 - Y_p}{Y'_p} & \text{for } \mathrm{Y}_p < 128 \\ 1 & \text{for } \mathrm{Y}_p = 128 \end{cases}$$

Prior to running the algorithm on our test image, we took 44 images of the Rubik's under varying lighting conditions to train a multidimensional MAP detector to differentiate between the six classes of pixes we expect to see on the faces of the Rubik's Cube. We used 9x9x9 uniformly spaced bins for the normalized YUV values.

Then for each of the faces of the Rubik's Cube in our test image, we use our MAP detector to classify the colors of the faces. Because there was an overlap in the bins in our MAP detector for some of the colors, we classified the colors with a likelihood percentage. This allows some optimization to be performed in the next step of our algorithm.

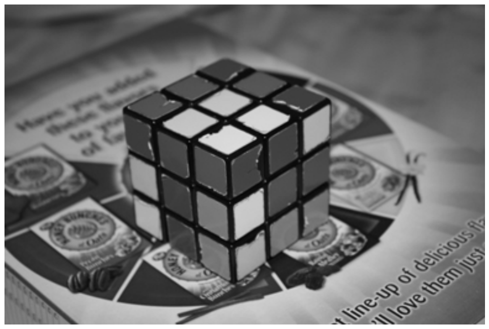
Fig. 6. Y Channel

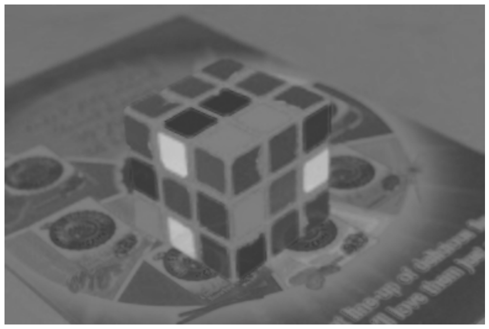
Fig. 7. Normalized U Channel

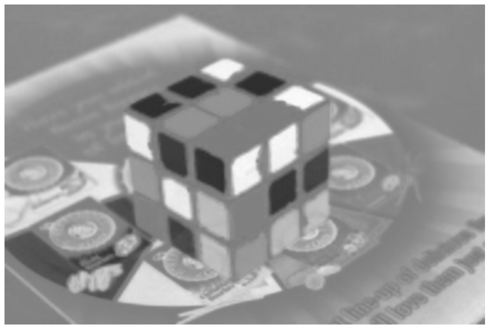
Fig. 8. Normalized V Channel

C. Determining the State of the Cube

In this step, we determine the state of the cube by maximizing the probabilities determined in the previous section. We also take into consideration restrictions on the colors of the pieces of the cube. Examples of the restrictions are that the blue center piece must be on opposite as the green center piece, there is only one blue-yellow side piece, a red-orange side piece does not exist, etc.

III. Experiments

To test our algorithm, we took 20 sets of test images (each set consisted of two images capturing 3 different faces of the cube in each). The test images were taken in varying lighting conditions: sunlight, under a desk lamp and using the flash from a camera. We also took our test images with different backgrounds, and with different states for the cube: both scrambled and unscrambled.

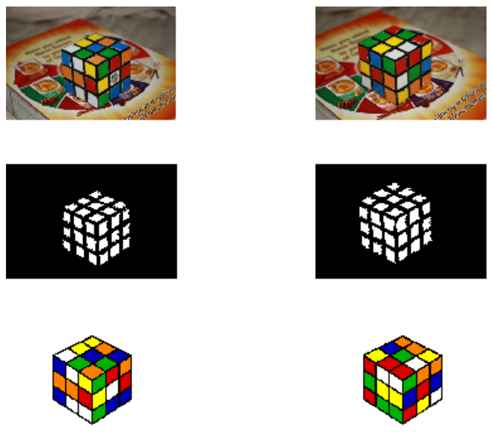
Fig. 9. The Original Image, the Mask, and Detected State

In our set of test images, we were able to successfully locate the faces of the Rubik's cube in 80% of the images. In the cases where the algorithm was unsuccessful, it was usually caused by a noisy background that was in focus and had blocks of color very similar to the Rubik's cube. Another cause was the reflection of the background in the cube for images taken in the sun. Our results here suggest that if our algorithm was extended to a mobile application where we have access to the frames in a video feed of the Rubik's cube, the success rate of locating the Rubik's cube would be much higher.

Once the Rubik's cube was detected, we had a 94 % accuracy in detecting the pixels. Errors existed in bunches because of our assertion on the number of pieces that that the Rubik's Cube can have for each color. Therefore, we didn't have any errors in 69% of our test images but in the cubes with errors in the detection, we had accuracy percentages of as low as 52%. In the cases of errors, we usually mixed up similarly colored pieces such as the blue and green pieces or the red and orange pieces. This was usually the case with pieces on the edges that were as well illuminated as the rest of the image. Once again, if our algorithm was extended to a mobile application, we may be able to increase our accuracy by averaging the predictions across various frames from a video feed.

Results

	n	Detection %	Accuracy %
Sunlight	6	50%	96%
Desk Lamp	6	83%	87%
Flash	8	100%	97%
Scrambled	16	75%	94%
Unscrambled	4	100%	95%
Noisy Background	7	71%	88%
No Noisy Background	13	85%	97%
Total	20	80%	94%

IV. Conclusion

The results of our algorithm were very promising. We were able to detect the Rubik's Cube in 80% of our images and from just two images of the Rubik's Cube, we were able to attain a 94% accuracy in identifying the colors on the faces of the Rubik's Cube. This algorithm can be extended to a mobile application to improve the detection and accuracy rate by using frames from a video feed of the Rubik's Cube. Furthermore, this algorithm can be used in conjunction with a Rubik's Cube solver algorithm to determine an method for unscrambling any Rubik's Cube without having to input the current state of the Rubik's Cube manually.

References

[1] "Rubik's Cube." Wikipedia. Wikimedia Foundation, 06 Feb. 2012. Web. 06 June 2012. <http://en.wikipedia.org/wiki/Rubik's_Cube>.

[2] Kasprzak, Wlodzimierz, Wojciech Szynkiewicz, and Lukasz Czajka. "Rubiks Cube Reconstruction from Single View for Service Robot."