Name: Asadullah Hill Galib

Netid: galibasa

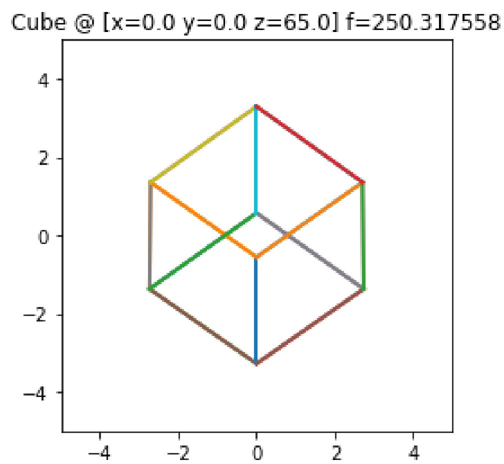
CSE 803 Computer Vision: Homework 1

1. Camera Projection Matrix

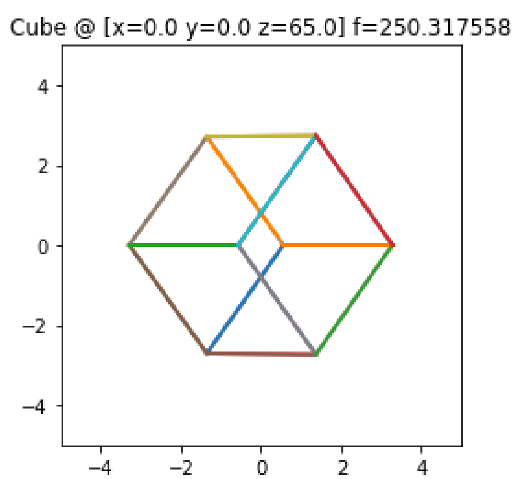
1(a) cube.gif attached

1(b) 3D rotation matrices are not commutative as the resulting transformations yield different views.

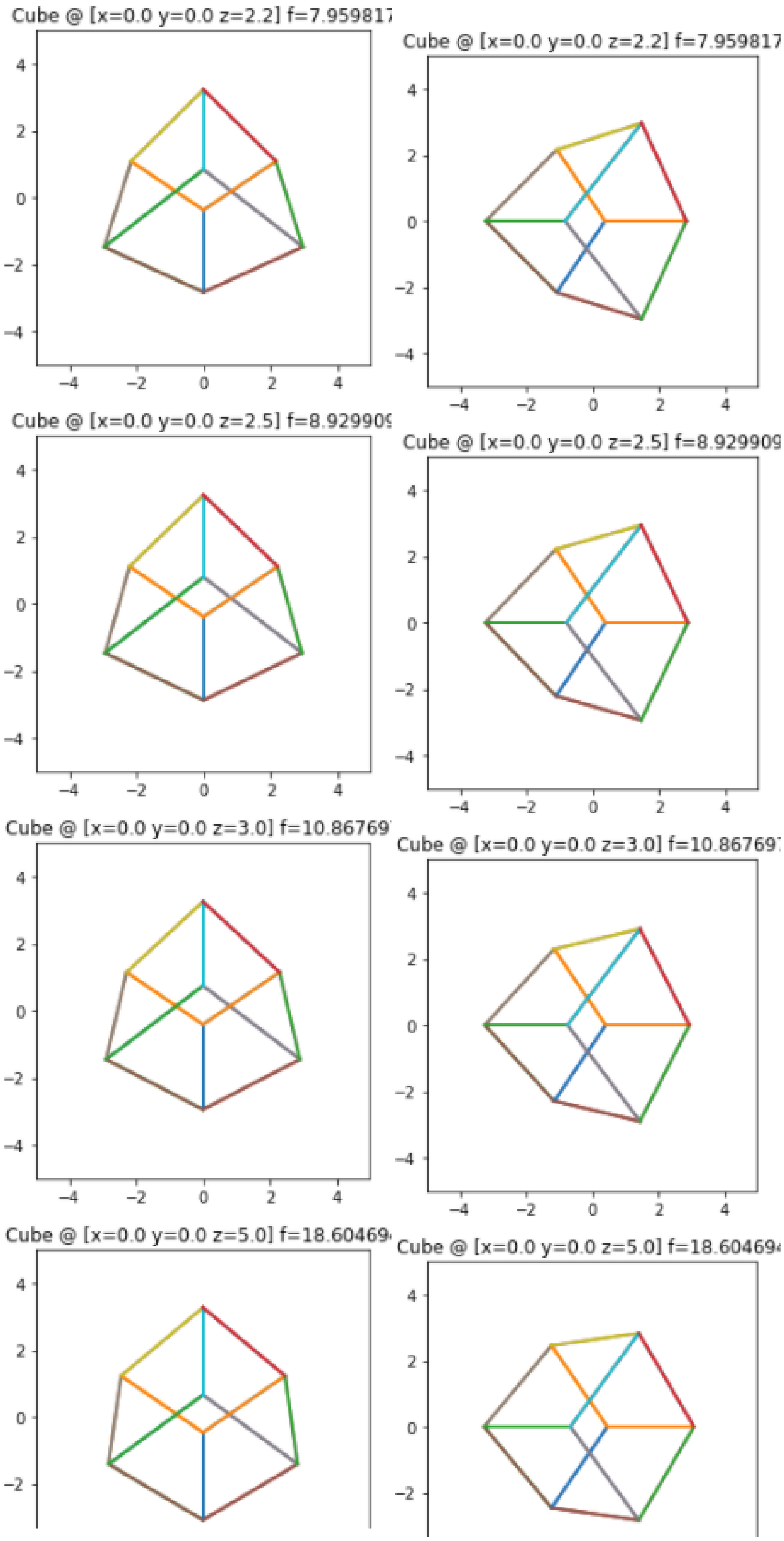
Here, (a) rotX(theta), followed by rotY(theta) with z=65:



Also, (b) rotY(theta), followed by rotX(theta) with z=65:



For different values of z → (1, 1.25, 1.5, 2, 4, 8, 16, 32,64) + , the resulting views are different. The following images depict that (left figures are for (a) and right figures are for (b)):

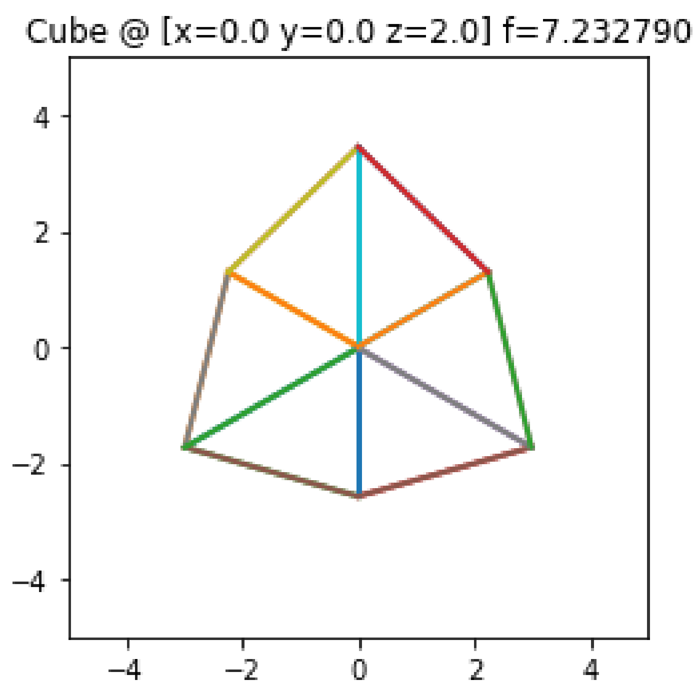


1(c) A projection of the cube where one diagonal of the cube is projected to a single point is depicted in the following figure.

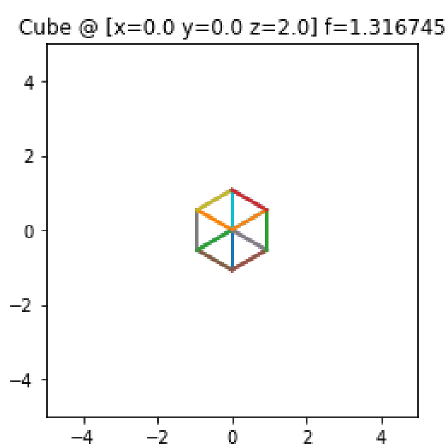
Order: Rotation, R = rotX(theta1), followed by rotY(theta2) using np.dot

Parameters:

- theta1 = 0.195*pi [pi = 3.1416]
- theta2 = 0.25*pi
- z = 1
- scaleFToSize = 6



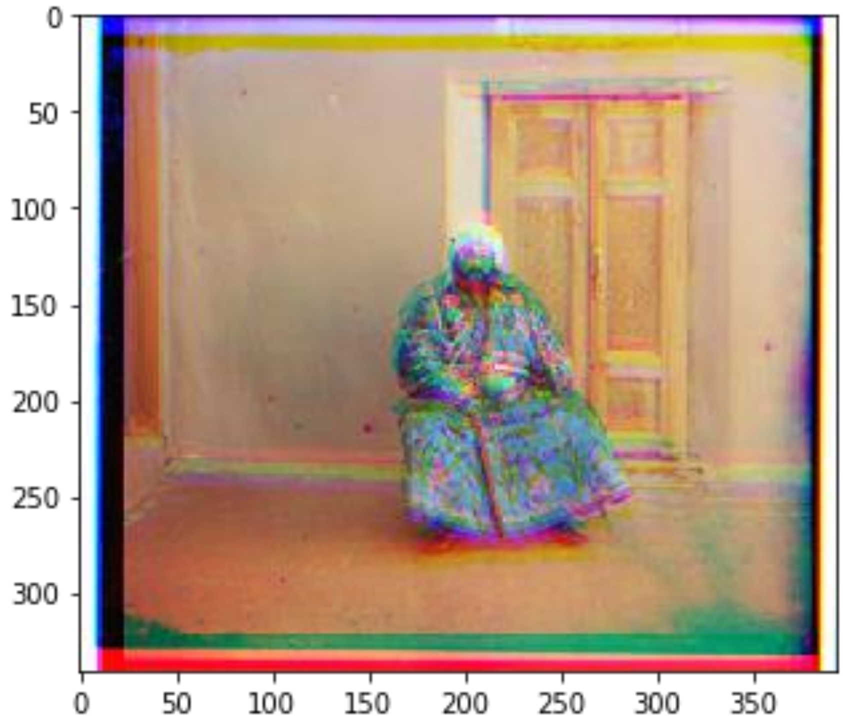
1(d) Implementation is provided in .ipynb file.



2 Prokudin-Gorskii: Color from grayscale photographs

2(a) Task 1

Output colored image:



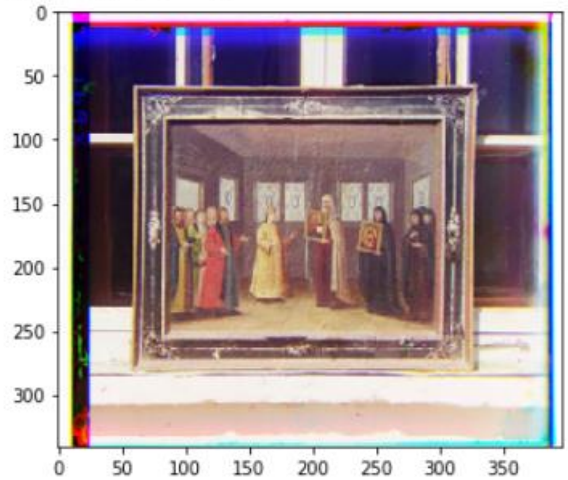
2(b) Alignment

Using the *calculated_offset()* function, possible pixel offsets in the range of [-15,15] are searched and the best alignment for different R, G, B channels. This function returns the offsets required for shifting two of the color channels (R and G) with respect to third. The third channel (B) is fixed: (0,0) offset. This search process is carried way in a brute-force approach based on a metric that maximizes the similarity.

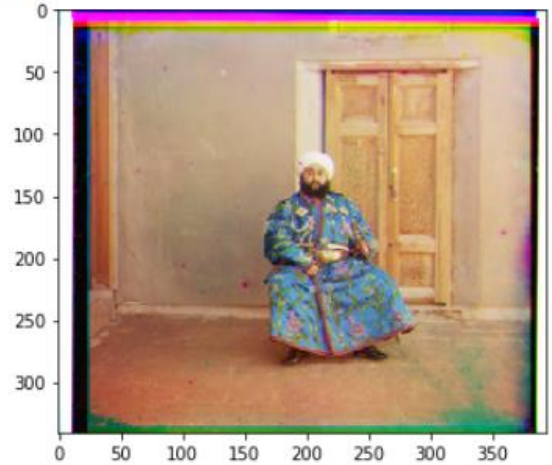
Here, sum of square distances is used to check similarity and the goal is to minimize this distance which denotes higher similarity. It is calculated using the sum of square distances between individual pixels for all channels. This function is implemented under *sum_of_square()* which takes two images and returns the metric result.

Output colored image with associated offsets (**offset shown in the upper left corner**):

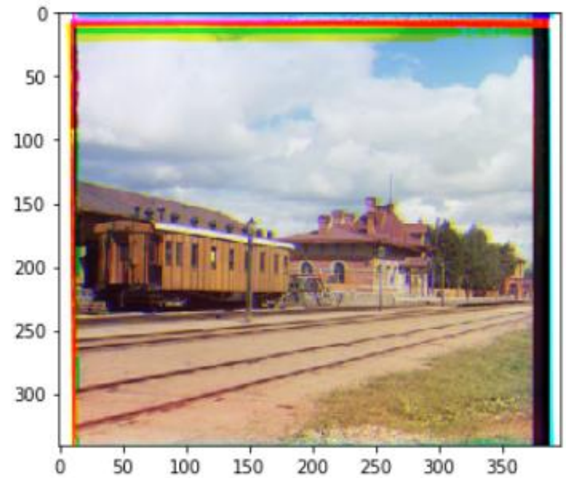
[[2, 4], [2, 9]]



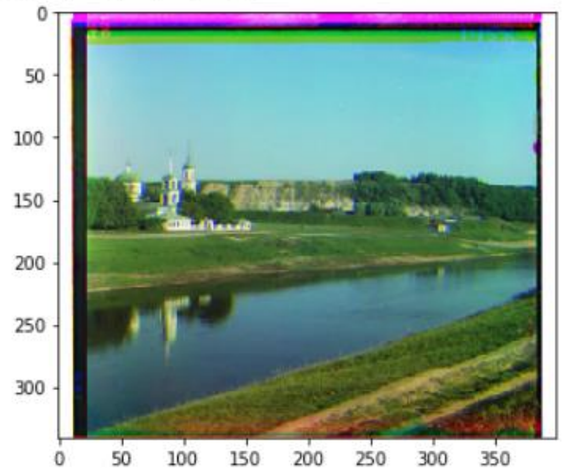
[[1, 7], [3, 14]]



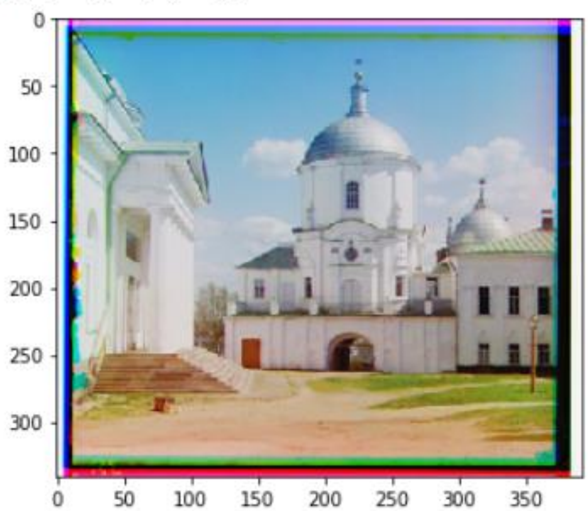
[[-1, 6], [0, 12]]



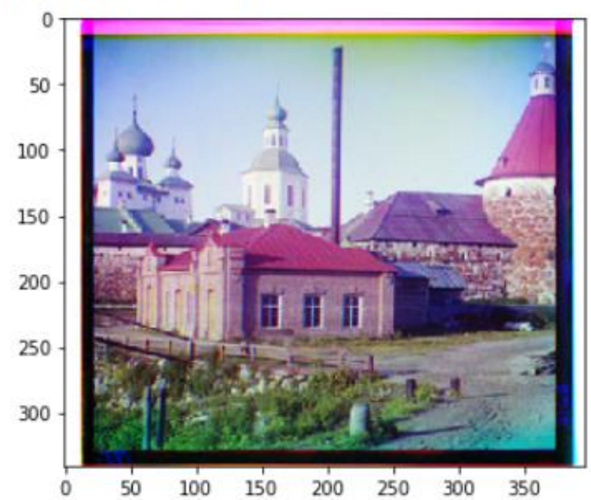
[[-1, 5], [-2, 9]]



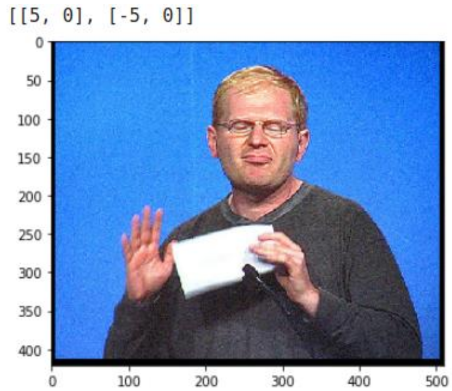
[[0, 0], [1, 5]]



[[0, 4], [0, 13]]



efros tableau.jpg:



2(c) Pyramid

Here, a recursive function – *pyramid ()* is built. It recursively factors up the tripe-frame image by a factor of 2 and then applies the alignment task from the previous part. Offsets are calculated to shift 2 channels (R and G) and another channel is fixed (B). After calculating the offsets using the *calculate_offset()* function, those offsets are multiplied by the current factor to fix alignment.

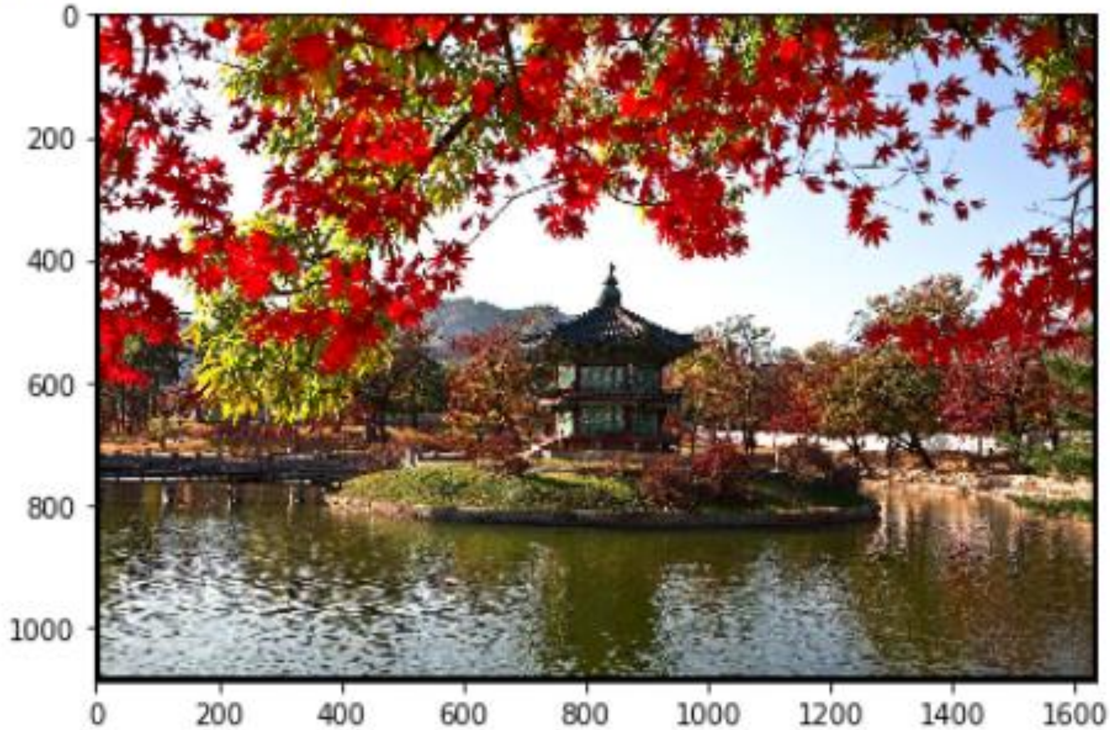
Offsets calculations as well as alignment are applied on individual channel and merged back to full image at the end.

Also, horizontal shift and vertical shift are applied using the *np.roll()* function to apply the actual alignment. *h_shift()* and *v_shift()* functions are implemented for these.

In the end, using the *combine()* function, RGB processing is carried out, like loading and getting individual R, G, and B channels, calling the *pyramid()* function, after returning, re-scaling the images.

Output from this part is shown below:

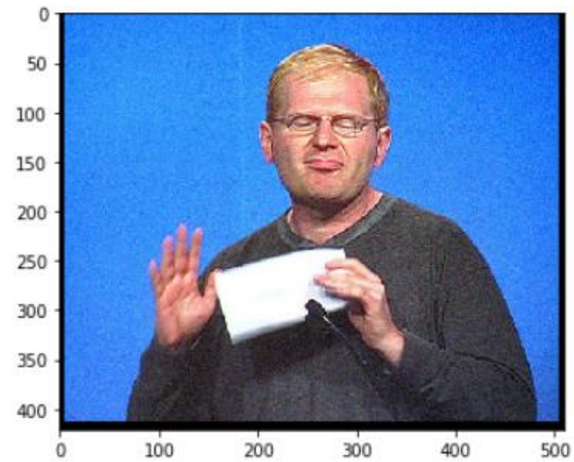
[[5, 2], [-1, 2]]


0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400
1600

[[10, -9], [-12, -8]]


0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400
1600

Offsets (Left to right): 1: ((2,4), (2,9)), 2: ((1,7), (3,14)), 3: ((-1,6), (0,12)), 4: ((-1,5),(-2,9)), 5: ((0,0), (1,5)), 6: ((0,4), (0,13))



Offset: ((5,0), (-5,0))

3 Color Spaces and illuminance

3(1) Output (outdoor):

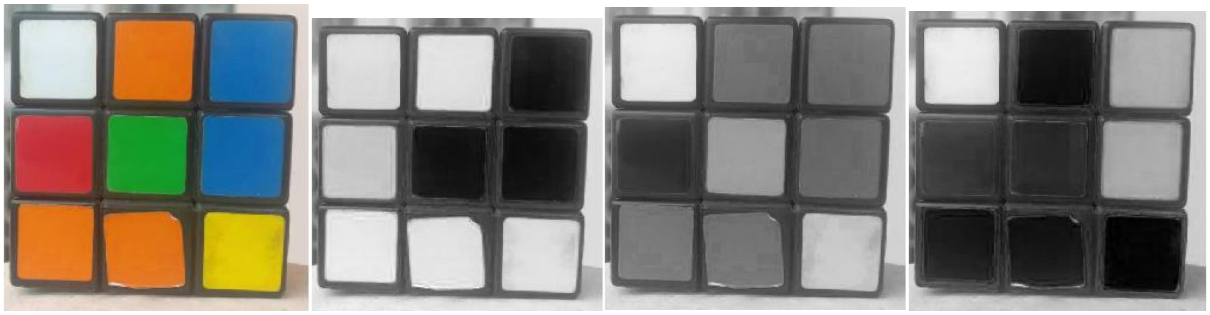


Figure 1: Left to Right: Original outdoor image in RGB, R channel, G channel, B channel

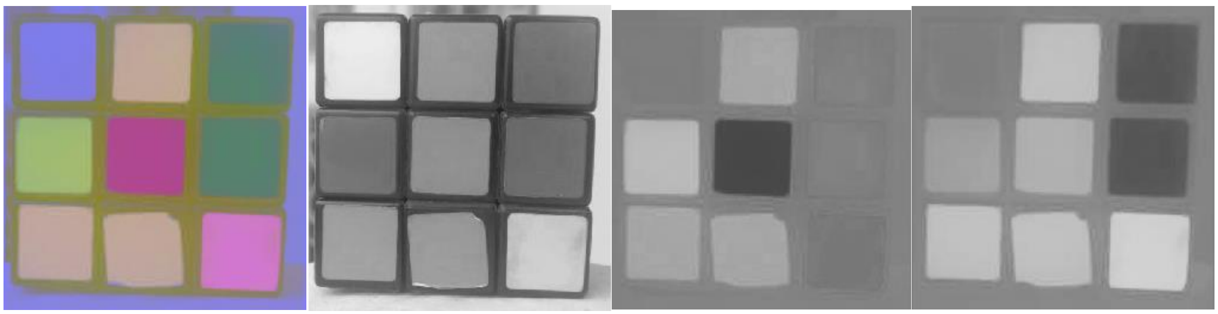


Figure 2: Left to Right: Original outdoor image in LAB, L channel, A channel, B channel

Output (indoor):

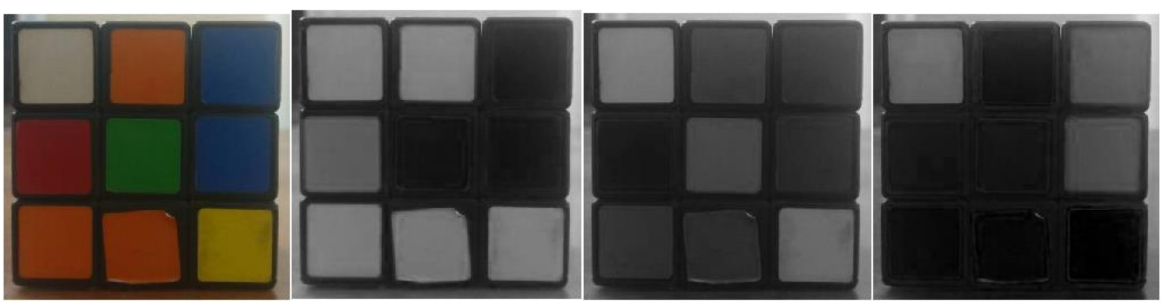


Figure 3: Left to Right: Original indoor image in RGB, R channel, G channel, B channel

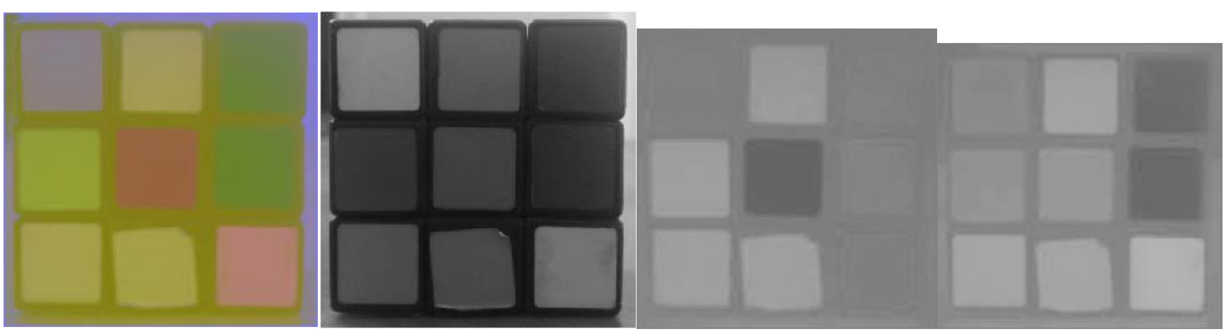


Figure 4: Left to Right: Original indoor image in LAB, L channel, A channel, B channel

3(2)

From figure 1 and 3, the RGB channels hardly represent the illuminance change because RGB doesn't deal with light individually, it blends illuminance into the color channels (R, G, and B). So, it generates some kind of unnormalized representation.

But for LAB, according to figure 2 and 4, particularly the L (light) channel shows noticeable changes with respect to different illuminance level. That L channel closely represents the illuminance level of the actual images (outdoor vs indoor). And, here the other two channels represent the color mostly, don't show any obvious changes with respect to the illuminance levels.

So, the illuminance change is better separated in LAB color space.

3(3) all the corresponding files are attached.