IRL #6: Progress Review

Man-Ning Chen (Mandy)

Team G: EXCALIBR

Teammates: Yiqing Cai, Huan-Yang Chang,

Siddharth Raina, Sambuddha Sarkar

Individual Progress

Overview

After our team meeting with Oculus, we set up several tasks for this semester. First, we are continuing the sensor noise calibration. An integrating sphere will be added into our system for high accuracy performance. Second, after proving the RGB photometric responses are linear, we are going to conduct color calibration and generate the mapping function of color sensors. Third, we will keep doing geometric calibration. Forth, in addition to the real-world experiment, a simulation is required. We will use Blender to generate simulated images of virtual targets from virtual cameras.

Color Calibration

Color calibration is to measure and adjust the color response of a device (input or output) to a known state. We use an X-Rite ColorChecker Classic Card [Fig. 1] as our ground truths (The manufacture gives us the true color values) and aim at mapping the colors recorded by the cameras these ground truths.

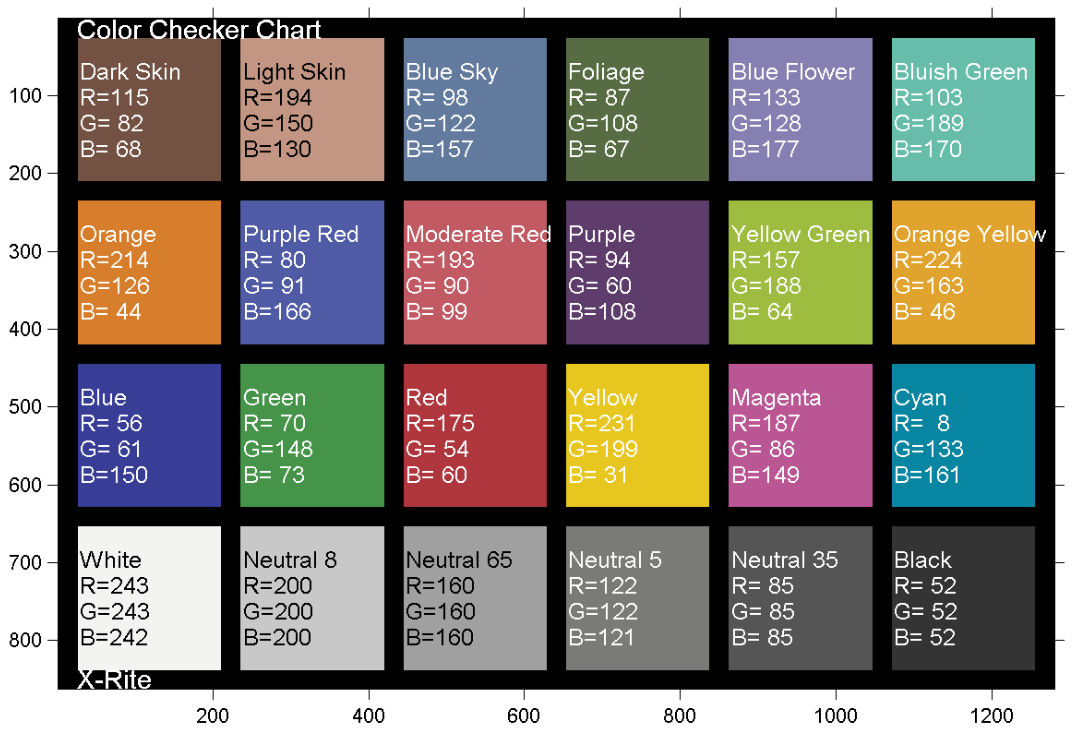

Figure 1. An X-Rite ColorChecker Classic Card

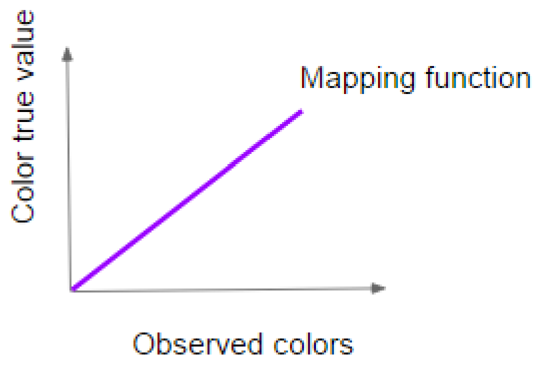

Figure 2. Mapping function (unreal): This graph is only for illustrate the concepts

Color Calibration Pipeline

I plan to divide and conquer the problem by doing color patch segmentation first and address the color board detection part later. I will combine these two parts to get the algorithm which can each color on the colorchecker in the images. In the end, I can use the recorded colors and the ground truth to generate the mapping function.

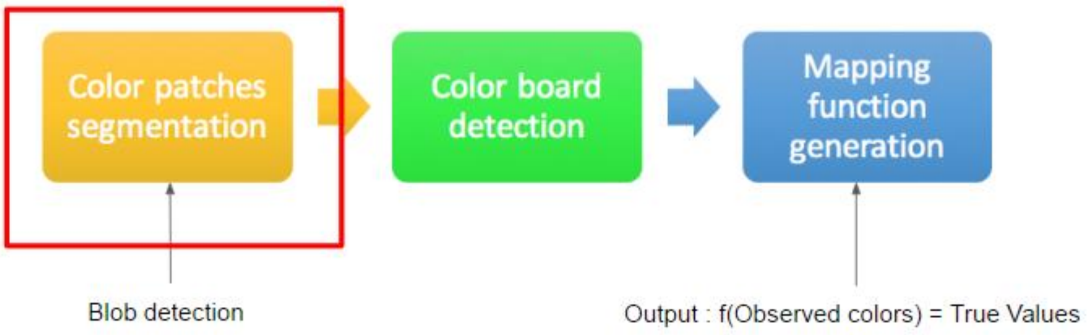
Color patches segmentation
Color board detection
Mapping function generation
Blob detection
Output : f(Observed colors) = True Values

Figure 3. Color Calibration Pipeline

Current Status

Note: I am currently using my personal smart phone for this algorithm development. After the algorithm is completed, the testing will be done on EVT cameras.

Since the colorchecker detection problem is ignored in current stage, I crop the image, leaving only the interested part in the image. [Fig. 4]

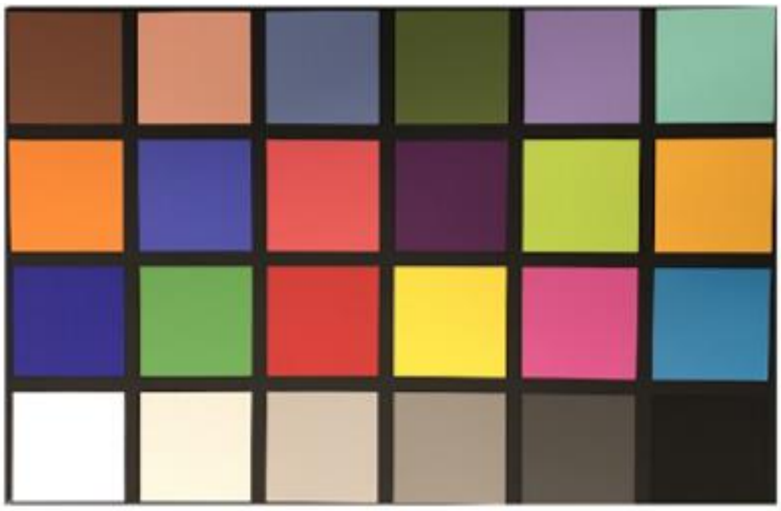

Figure 4. The cropped image

I use a threshold to cut off make the image separate into background (zero) and foreground (one). [Fig. 5] Then, the image becomes a binary image. [Fig. 6]

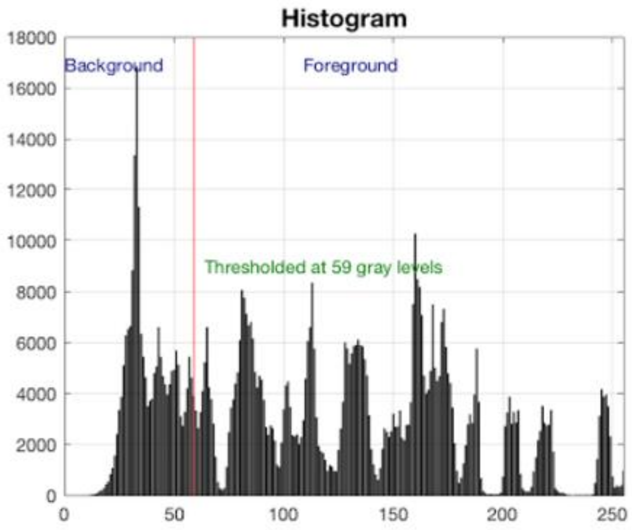

Figure 5. Intensity histogram of the image

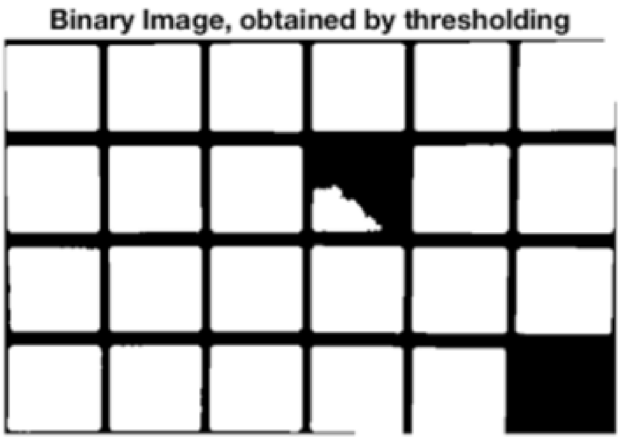

Figure 6. Binary image

We can see that the background splits the foreground into 23 patches. If we use flood-fill algorithm to find out and label these patches. [Fig. 7] Then, we can get the centers of each patches. [Fig. 8] As you may see in the image, the center of label 16 is not really in the center of the patch because we lost most parts in label 16. However, it doesn't hinder our goal which is getting the recorded value of each color patch and compare it with its correspondent ground truth.

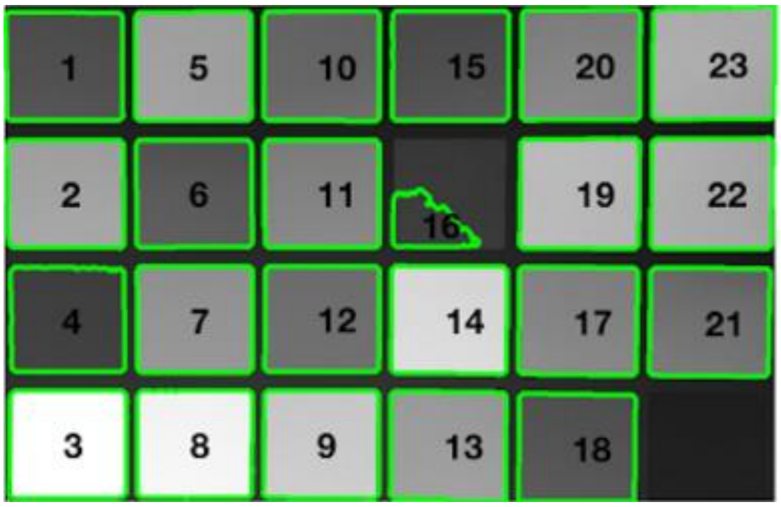

Figure 7. Labeled Image

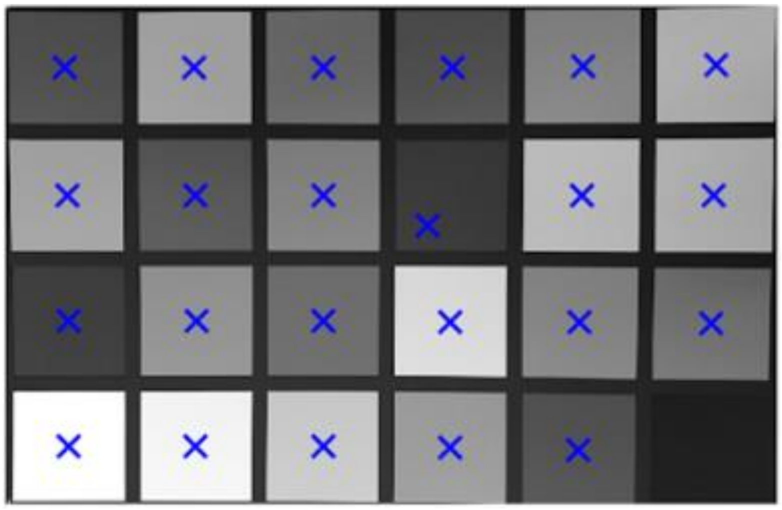

Figure 8. Centers of each color patches

Challenge

1. While the Intensity threshold used in our image cuts off the background, it also cuts off the black patch. We need other ways to find the center of the black patch. I am considering using the intersection of lines of other patch centers or extrapolating other patch center lines to get the position of the black patch.
2. The algorithm is easily affected by the lighting condition of the experiment condition. I'll have to conquer this problem.

Teamwork

Yiqing Cai	Generate one of the best path for robot arm (go through as least positions as possible to cover FOVs of all cameras)
Huan-Yang Chang	Simulation system setup, Geometry calibration
Siddharth Raina	Sensor noise calibration
Sambuddha Sarkar	Works on Blender. Generate virtual calibration target and render the virtual images of the target.

References

[1] https://en.wikipedia.org/wiki/Color_calibration

[2] https://en.wikipedia.org/wiki/Flood_fill