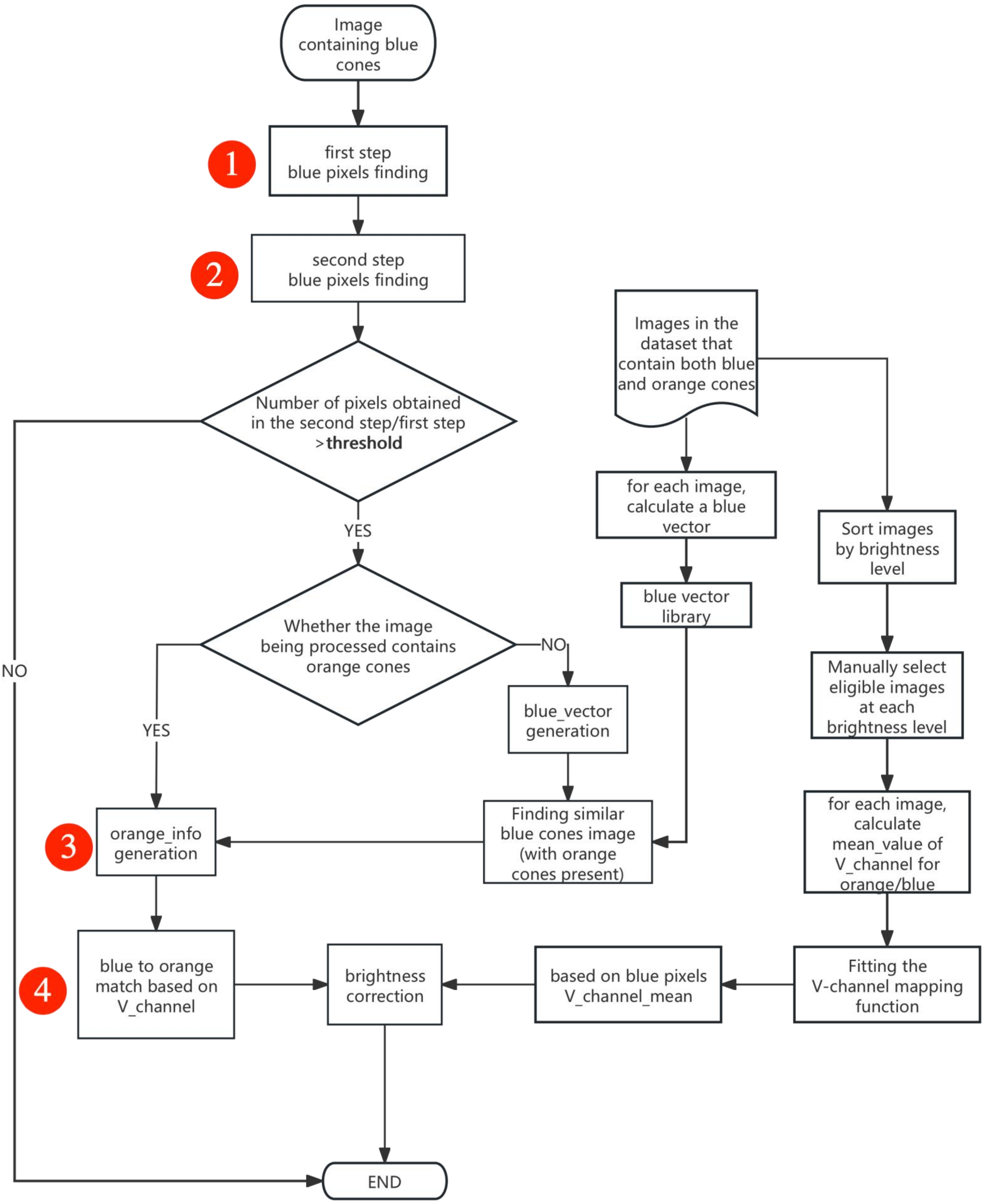
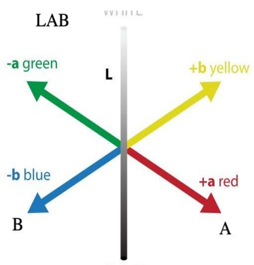
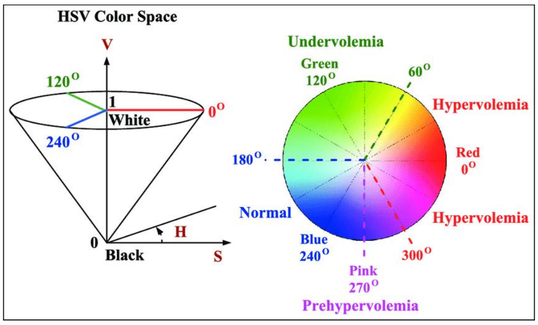


Image containing blue cones
1
first step
blue pixels finding
2
second step
blue pixels finding
Number of pixels obtained in the second step/first step >threshold
NO
YES
Whether the image being processed contains orange cones
NO
YES
Images in the dataset that contain both blue and orange cones
for each image, calculate a blue vector
blue vector library
Sort images by brightness level
Manually select eligible images at each brightness level
for each image, calculate mean_value of V_channel for orange/blue
blue_vector generation
Finding similar blue cones image (with orange cones present)
3
orange_info generation
4
blue to orange match based on V_channel
brightness correction
based on blue pixels V_channel_mean
Fitting the V-channel mapping function
END



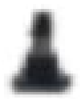


Explanation of key methods

1.**first step blue pixels finding**:

Theoretically, finding blue in LAB color space is very simple, we only need to focus on the negative half-axis of the B-channel. The range for negative half-axis of the B-channel in OpenCV is 0-127, if I set the threshold as 127, in practice, due to a variety of practical errors, it is not possible to obtain the desired results.

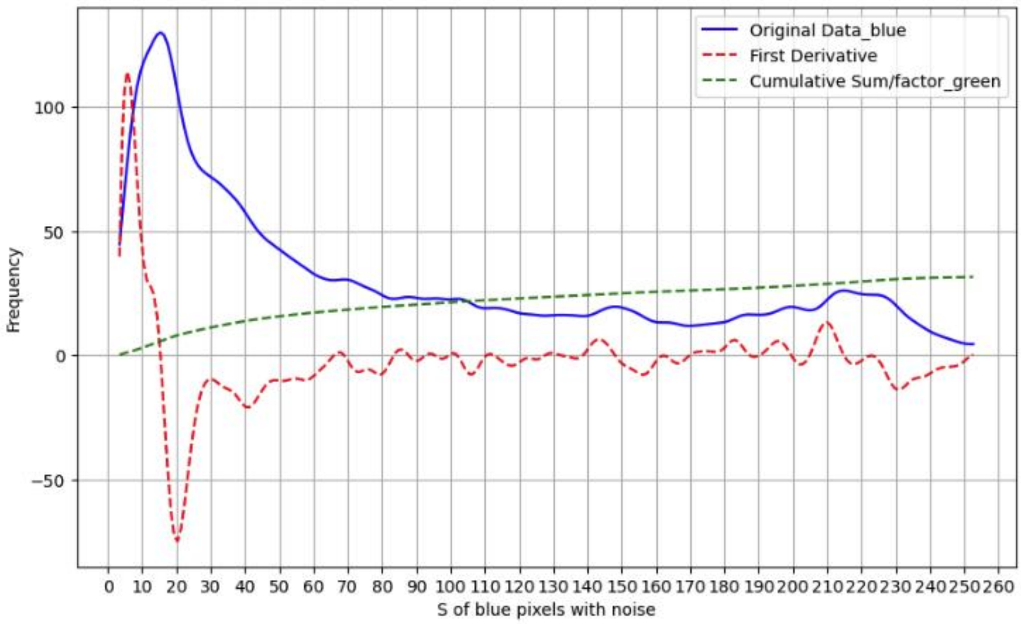
Typically, the result will look like the following image with lots of background noise and the white strip in the middle. We have to find the blue pixels exactly, then we can replace it to produce realistic orange cones.



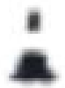
2.**second step blue pixels finding**:

After the previous step, if we transform the found pixels into HSV color space, we find that the S-channel statistics has the following typical pattern.

After much experimentation, I found that at the end of the first peak is an S-channel suitable threshold to separate the blue pixels from the background noise.

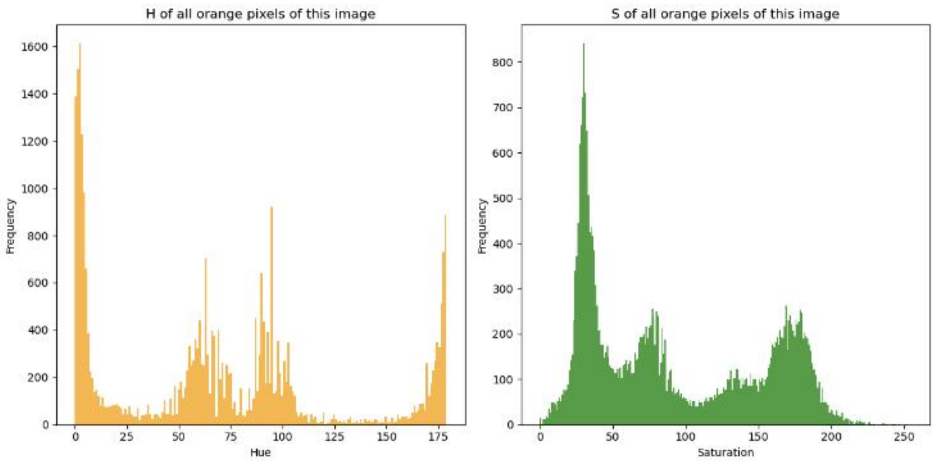


But for different images there exist different thresholds. How to find the threshold automatically? I found a way shown in figure above. We calculate the cumulative sum of original data, then divide it by the current range of the S-channel to lower it. Then the crossing is the proper threshold.



3. **orange_info generation**:

the statistics of orange bounding boxes are shown below.



So in HSV space, we can easily find orange pixels by set H_range(0-20) and S_threshold(100). We don't need to find orange pixels exactly, we can get information from the orange pixels we found.

For example, we found following 6 pixels:

(H,S,V)

(6, 120, 86),(4, 100, 86)-->(5,110,86)

(5, 78, 100),(3, 90, 100)-->(4,84,100)

(7, 130, 127),(5, 128, 127)-->(6,129,127)

Orange info obtained based on V channel, We find all pixels with the same V-channel value and average their H and S channels so that for each V value, there is a unique orange pixel.

4. **blue to orange match based on V_channel**:

In this step I target each blue pixel and use its V-channel to match the V-channel of the orange pixel from the previous step, find the one with the smallest V_difference and then the original blue pixel is replaced.

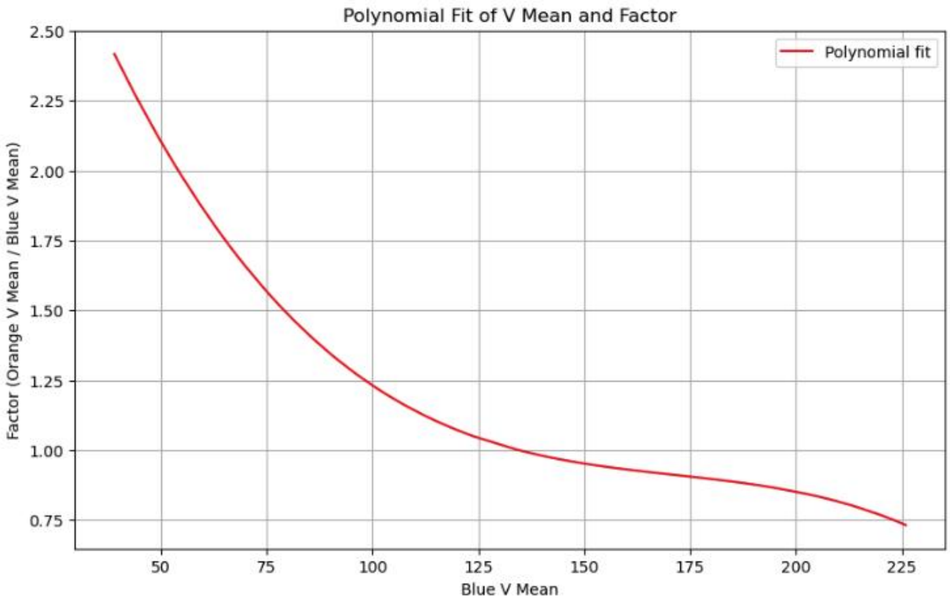
Q&A

-Why we need brightness correction ?

A: Under the same light intensity, the V values of blue and orange pixels are different, with the general rule being that in low light, the V value of orange will be several times that of blue, and in well light, the values are very close. Since we use the match algorithm above, we need brightness correction.

-How to do brightness correction ?

A: See the rightmost branch of the block diagram of the algorithm above. The blue and orange cones are clearly visible in the selected images and are both under the same lighting conditions as perceived visually.
The algorithm for finding blue and orange pixels is first implemented on the image, and for all found blue pixels, its average V_channel value 'blue_V' is calculated, and the same is done for orange, we got also 'orange_V'.
Then we have a point(brightness of image, orange_V/blue_V) for one image.
For many images of varying brightness, we fitted the following function.



-What is blue vector?

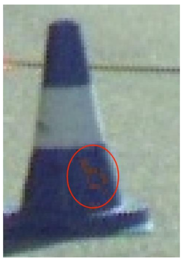
A: For each image containing blue cones (with or without orange cones), we can compute a blue vector with the following composition:
(global_brightness, HSV mean value of found blue pixels, number of blue cones, average size of blue cones)
It somehow quantifies the blue cones in a picture.

-What else can we do for image without orange cones?

A: For whole dataset, We can compute an orange_map using a similar or more advanced method than computing orange_info, and then use the pixel information in the map to replace the original blue pixels.

-Why we need ‘Number of pixels obtained in the second step/first step>threshold’?

A: Sometimes the algorithm for finding blue pixels fails, it is possible that the image does not match the assumptions, e.g. the S-value of the pixel of interest is not very different from the background. Blow is a case of failure。



Some results:

The number of orange cones has increased from about 20,000 to 40,000, alleviating the data imbalance to some extent.