**UAS PROJECT**

**-Made by Aaryaman Bisht(24/A11/006)**

**Links:**

[**https://github.com/ScriptedLines**](https://github.com/ScriptedLines)

[**https://www.linkedin.com/in/aaryaman-bisht-543816324/**](https://www.linkedin.com/in/aaryaman-bisht-543816324/)

**Documentation**

I thought of 2 ways to proceed with the solution of the problem. First didn’t worked an second one is the final, on which I built the code.

My first thoughts to begin with the solution was to find all the contours in the image, using cv.RETR\_CCOMP retrieval method and get the info about the hierarchy. I would have checked each contour if it had a children or not, if the contour had children that means it was the contour of the larger land (i.e. the brown or green part), this would have segregated the contours of the triangles and the main land. Then I would have checked for the mean colour of the parent contour and get the position of contours in the hierarchy array I got from the findContour fxn. Then I would have checked for the children contours to check the parent of those and find out which part of region, the contour lie in (in the burnt part of green part). Then I would have taken the mean colour of the triangles and got all the info. But I was not able to detect the contours using findContours. I also tried canny but to no avail. And thresholding in binary was useless in this case, because here we had to segregate the shapes having 4 different colours.

Now the second method, with the help of which I built my code.

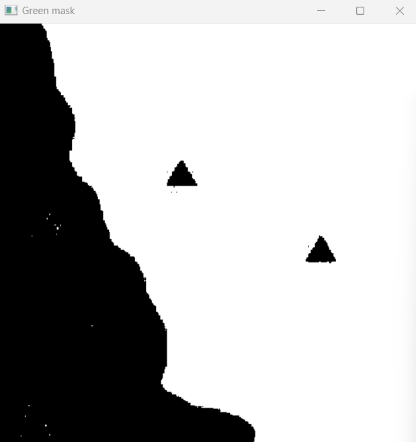
The basic principle of tis approach is masking using inRange() functions, which masks the area of the image ghat lies in the upper and lower limit of the color (in my case HSV) passed in the inRange function.

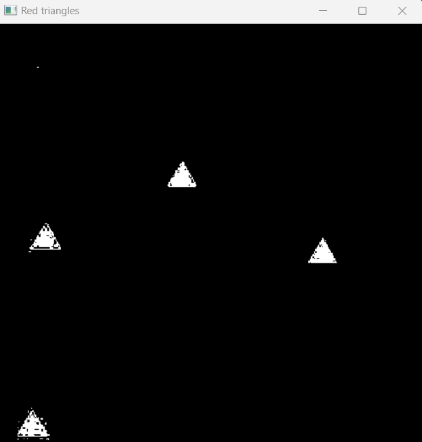
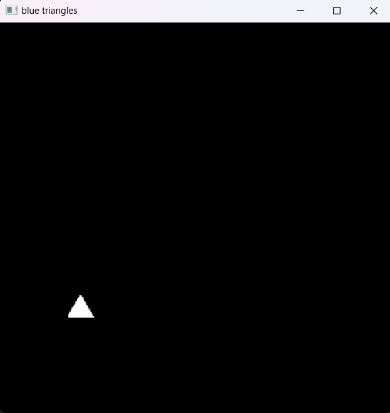
I masked the green part, the burnt part, the red triangles and the blue triangles using the inRange fxn which gave me 4 different binary images.



Original Image

Binary Images of:

   
Burnt Part Green Part

Red Triangles Blue Traingles

After this, I thought of 2 approaches:

First, That I would somehow paint the binary images of green and burnt part in separate solid colours to clearly find contours and also make the triangle a red and blue respectively. Then in than combined image I would proceed with finding the hierarchy and finding which triangles lie in which parent contour. But this was a longer procedure as I already had everything separated. So I did it with second approach.

In this I first took the binary image(mask) of the green part, found contours in that, which was only one(the green region) and drew the contour on the binary image with color value(255,255,255)[white colour], this filled the binary image with the triangle cut out of it and made it solid white. Same I did it with the brown part as well. This is done so we can later take cv.bitwise.and() to merge the triangles with the green and brown part so we can identify which triangles lie in which part

Green Part filled Brown Part filled

Now I took bitwise and with the binaries

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **&** |  | **=** |  |

**Green Part Red Triangle Bitwise and of green part and**

**Red triangle (red triangles in green part)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **&** |  | **=** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Green Part** |  | **Blue Triangle** |  | **Bitwise and of green part and blue triangle(blue triangle in green part)** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **&** |  | **=** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Brown Part** |  | **Red triangles** |  | **Bitwise and of brown part and red triangles (red triangles in brown part)** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **&** |  | **=** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Brown Part** |  | **Blue triangle** |  | **Bitwise and of brown part and blue triangles (Blue triangles in burnt region)** |

Now we have the number of triangles of each colour in each region. This soves half the problem. Now by basic list manipulation, we can find, number of houses in each region, number of priority of houses in each region and ratio of priorities of brown region and green region. We store this in a variable inside the for loop which we are using to iterate through images, Then append the data in the global list.

Now the next step of solving this problem, we need to form a final image, highlighting the green and brown region in a different solid colours.

Do this, I first created a solid yellow window the same size of the window of green region mask, and with bitwise\_and we colour the white region of the mask in yellow colour. Same we did for brown region.

For green region:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **or** |  | **=** |  |

For brown region:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **or** |  | **=** |  |

Now for making a land image, images of green and brown region , we use bitwise\_or again

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **or** |  | **=** |  |

Now for the houses, first converted the masks from gray2BGR, and then found contours in those masks, there was some disturbance in the mask, so I only considered contours having perimeter in range of 80 to 300, and drew contours over them by passing -1 in thickness so it fill the contour in a solid colour.

Now I took bitwise\_or of the blue and red triangles and got this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **or** |  | **=** |  |

Now for the final image I took bitwise\_or of the triangles and the colored region

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **or** |  | **=** |  |

These procedures were for a single image, for multiple images in directory, we get the list of names of images, using os.listdir(path\_to\_image\_folder), and iterate through each image.

And append the data to the global list. But one problem I aced was, when I printed the list I got from os.listdir(). It was not in order, so I had to sort it using list.sort().

Then the last step, in which we have to sort the names of the images in descending order.

For this I first copied the list that had the priority ratios. And sorted it in descending order. Then I iterated through each elements and found the index of the element in the original priority order list. Then I appended the image corresponding to that index in a new sorted\_img list. Some of the ratio values were same, so I had to copy the priority\_ratio to a new list and also the img\_name list to a new list. And with each iteration I deleted the element from the copied list of ratios and images. So even the images with same ratio can be appended into the sorted\_img list.

Thes were all the steps I took to complete the program.

I also used the hsv range detector program I got from stackoverflow to determine hsv range of green, brown, red and blue colour.

Link: <https://stackoverflow.com/questions/10948589/choosing-the-correct-upper-and-lower-hsv-boundaries-for-color-detection-withcv>

Name: Aaryaman Bisht

24/A11/006

Contact:

9873210256

aaryaman.bisht2006@gmail.com