HOMEWORK 1 U1
BASIC SEGMENTATION OPERATIONS

Héctor Mauricio Mendoza Xicoténcatl

Abstract—In this report we documented our coding process of basic segmentation operations. Here you will able to see the results of the first topics seen in unit one of Artificial Vision: Thresholds, Color Segmentation, RGB and HSV colors.

Index Terms—Thresholds, Color Segmentation, RGB, HSV, OpenCv2, Bitwise Operation.

1 OPENCV2

It is a library of programming functions mainly aimed at real-time computer vision. Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing

2 PROBLEM TO SOLVE

The main objective to solve in this activity was applying the 5 different types of basic thresholds in OpenCv programming. Those basic thresholds are:

1) Binary Threshold.
2) Binary Inverted Threshold.
3) Threshold to Zero.
4) Threshold to Zero Inverted.
5) Threshold truncated.

In addition, we had to segment an image by 3 different colors in RGB and HSV scales.

3 CODING PROCESS

3.1 BASIC THRESHOLDS

The first exercise consisted of the 5 threshold programming codes. In this section, we first imported the libraries needed to run our code. Then, we used the command **cv2.imread(")** to read our source image which will be considered as our main input image.
One problem regarding OpenCv is that it reads the input image in the BGR scale. So, with the **cv2.cvtColor(")** we convert this image from BGR to grayscale in order to apply the thresholds later. The following step is to apply the five thresholds. Thus, we use the command: **cv2.threshold()** and put the corresponding data to apply the five thresholds.
Finally, we use the command **cv2.imshow(")** to show the final results of the output image.

3.2 IMAGE SEGMENTATION

Similar to the previous exercise, we import our libraries and read the input image.
In contrast with the first exercise, this time we used the command **cv2.cvtColor(")** to change the color of our input image. These results gave us the same image but in RGB and HSV ranges which will later be useful.
Next, we chose our the color ranges we want to work with, the author of this document chose:

1) lower_green = (30,100,50)
2) upper_green = (80,255,255)
3) lower_blue = (100,100,50)
4) upper_blue = (110,255,255)
5) lower_white = (228,236,237)
6) upper_white = (255,255,255)

The next step is to create the masks. Then we use the command **cv2.inRange(src, range1, range2)**. With this command we will be able to take a set of color values of the input image and only show those color values. These values will

be saved in a variable called mask.
In addition, we created two variable called

1) **final_mask**
2) **final_mask2**

Each variable will save and combine three set of color values. The first one, will save the lower values. The second one, the upper ones.
Finally, we used the command:

cv2.bitwise_and(img1, img1, mask = final_mask)

This command will perform a bit-wise AND operation of the color values saved in the **final_mask** variable and apply them into the original input image. Finally, we used the command **cv2.imshow(")** to print the results.

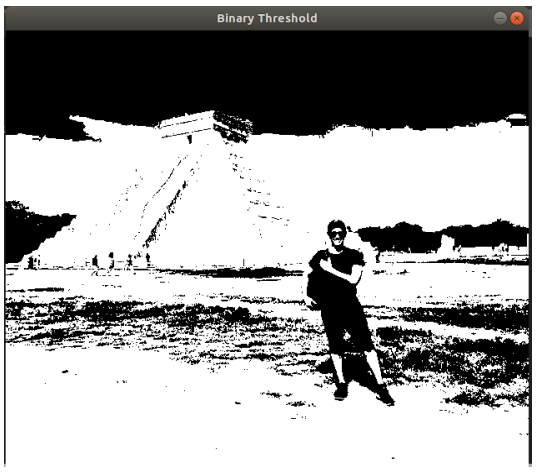


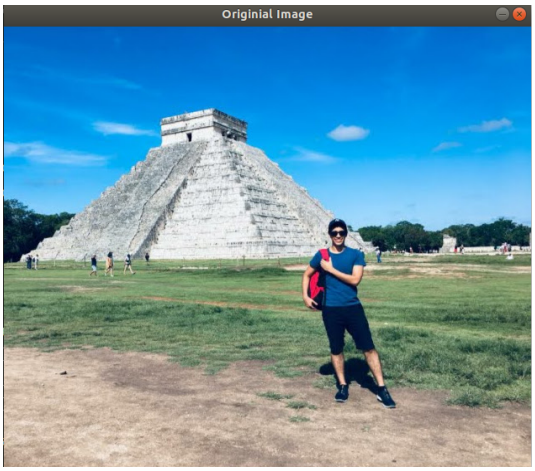
Fig. 1. Binary Threshold

4 CONCLUSIONS

There were many problems for me to understand the logic of the exercise. For example, the output image results of the second exercise. It was difficult to identify if the color segmentation was the correct one; Because you do not have any sort of reference to compare your output results. Another example was the role of the masks variables. At first sight, I did not realize the values were considered arrays, ignoring the fact you can make basic operations to get multiple ranges of the same color.
Fortunately, these doubts were solved thanks to the documentations and help of the teacher.

Fig. 2. Binary Threshold Inverted

5 RESULTS



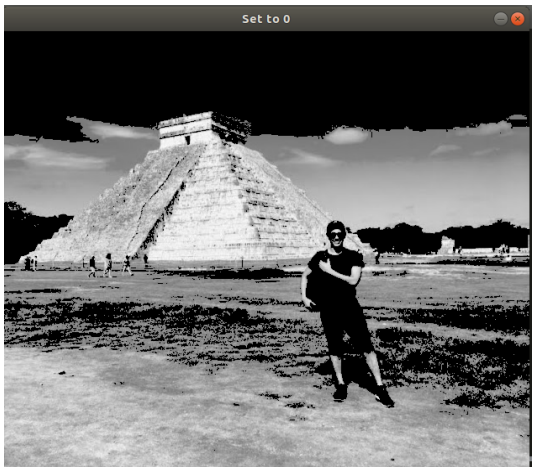


Fig. 3. Set to Zero

5.1 THRESHOLD RESULTS

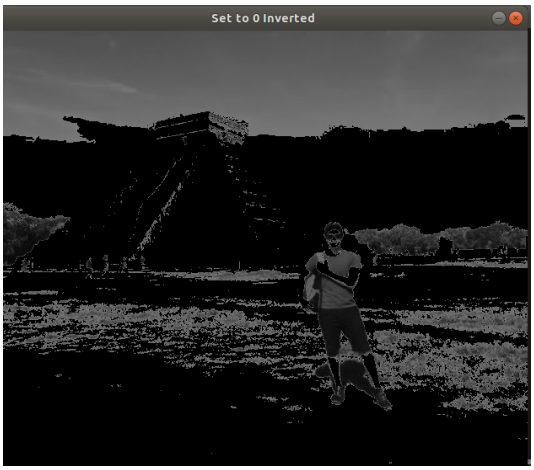


Fig. 4. Set to Zero Inverted

Fig. 5. Truncated Threshold

5.2 RGB - HSV SCALE RESULTS

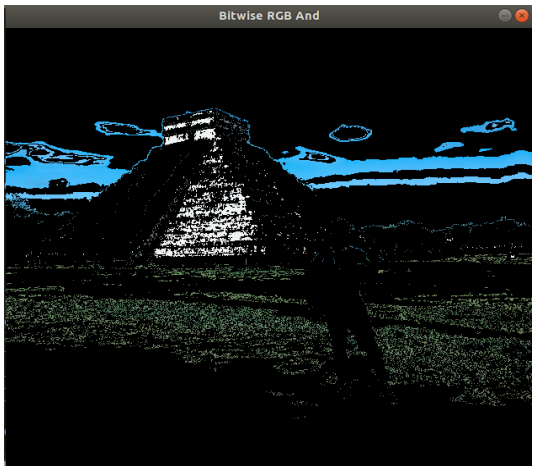


Fig. 6. Bitwise RGB

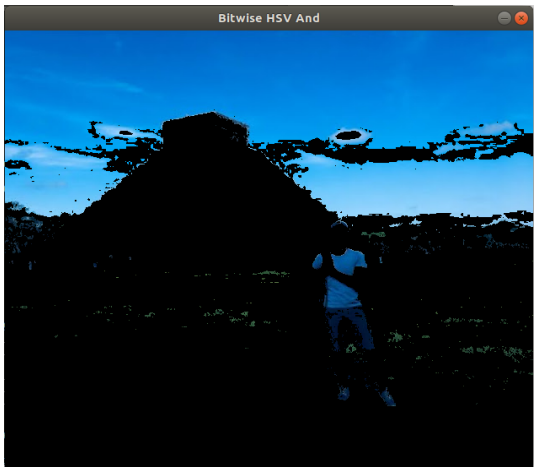


Fig. 7. Bitwise HVS

REFERENCES

[1] "OpenCV", Es.wikipedia.org, 2020. [Online]. Available: https://es.wikipedia.org/wiki/OpenCV. [Accessed: 10-Sep- 2020]

Héctor Mendoza Héctor Mauricio Mendoza Xicoténcatl is a Computational Robotics Junior student, currently stu-diying Universidad Politécnica de Yucatán.