BILKENT UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

CS 484

TAKE-HOME QUIZ 1

BERAT BİÇER
21503050

08.10.2018
ANKARA, TURKEY

INTRODUCTION

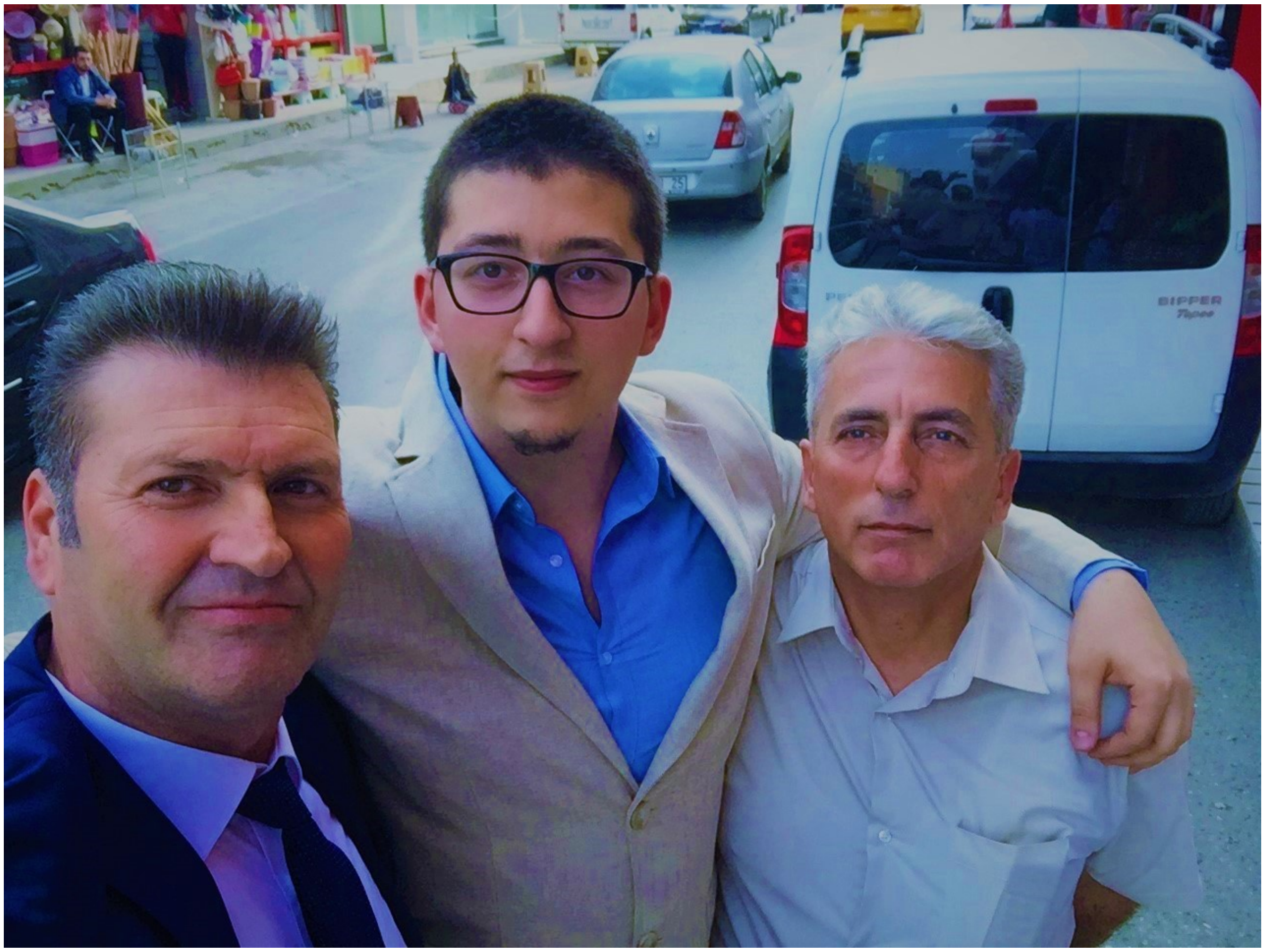

Figure 1: Original image

This image fits the criteria since it contains
- 3 people who are easily noticeable,
- 2 cars behind the people which are less obscured compared to the black car on the far left,
- 2 accessories (glasses and a tie) that have darker colors compared to their surroundings.

However, please note that this image was applied a filter that not only increased its brightness and contrast, but also the blue tone of every pixel. As a result, thresholds used for detecting the objects enumerated above have high values of blue as well.

In this report, I attempt to detect the objects enumerated above using thresholds. There are 2 major factors complicating this process as below, resulting in less accurate object detection:
- The lighting: Sunlight reaching the camera from behind the people causes shadows that reduce the color tones of the pixels. Also, pose of individuals engenders unequal light distribution among different sides of their faces, as in the one on the left where left side of his face is lighter compared to the right side. Moreover, the top side of the image has high brightness levels and less hue. The brightness in this area increases all red-green-black values of the pixels in that region, and increasing the noise in the resulting binary image for detecting 2 cars mentioned before.

- Color tones: Because there is a filter that increases the blue tone of every pixel, it is difficult to separate pixels having similar red-green-blue values because colors are blending in. For example, the car on the right-end side of the image is originally white, however in this picture the shirt of the one on the right has a very similar color to the car mentioned.
- Low contrast: Because binary thresholding requires high contrast, and the settings this photo was taken were poor, binary thresholding is expected to perform poorly compared to a high quality version of the same image.

In this part of the report, I will discuss my method, resulting binary images, and difficulties I experienced during this quiz. Corresponding MATLAB script can be found at the end of this report.

PART 1: FACES

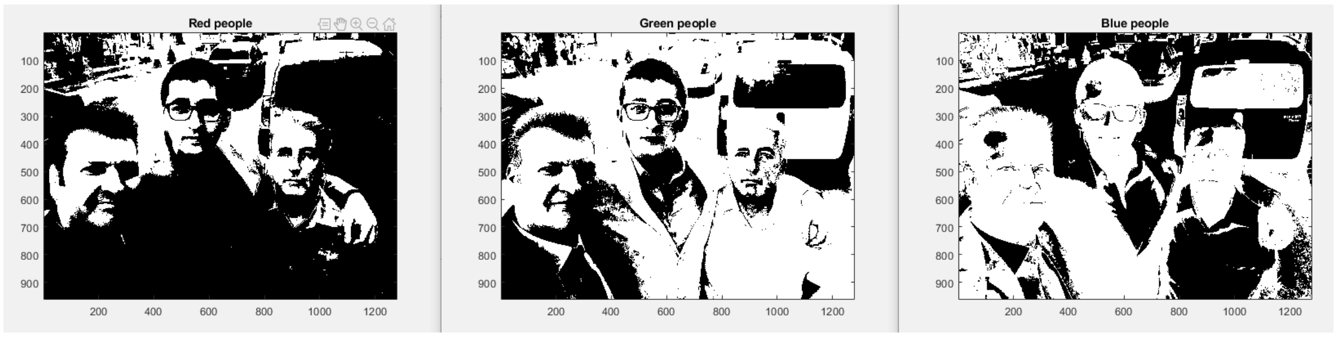

Figure 2: From left to right -> red, green, and blue thresholding

In this image, we can see the single-color thresholds for red, green, and blue. For red, threshold is $200 > image(:,:,1) > 125$, for green threshold is $200 > image(:,:,2) > 70$, and for blue threshold is $200 > image(:,:,2) > 70$ where *image* is the original image being processed. Using the red threshold, we can eliminate the clothes of the individuals and the car behind the one on the right. Also, individuals' faces are clearly visible. However, we may not use this threshold directly, since the top part of the image is also selected, creating noise. By applying logical *and* operation on the thresholds red and green, we improve the face boundaries and reduce the noise. Lastly, applying logical *and* operation on the result of the previous operation and the blue threshold, we minimize the noise on the top side of the image as well. The resulting binary image is as follows:

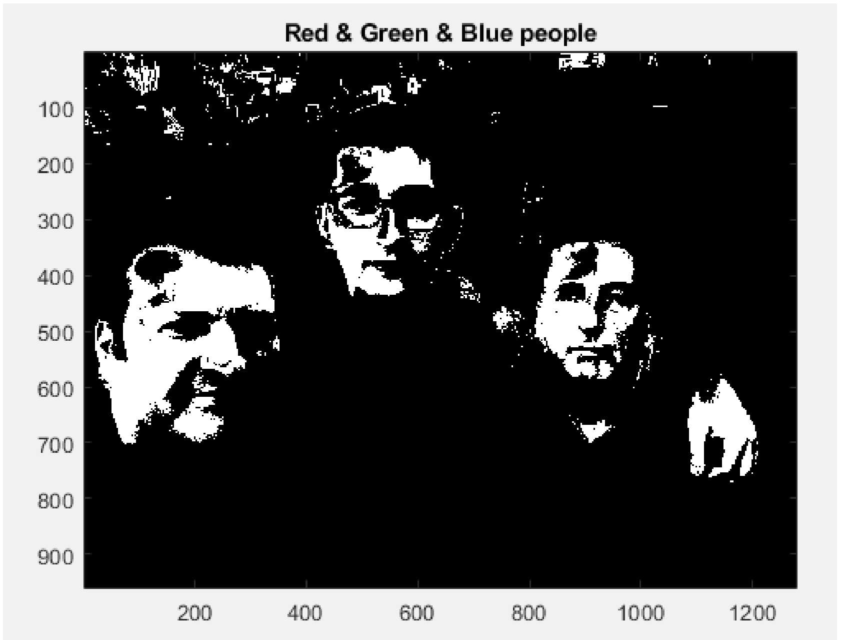

Figure 3: Combined binary thresholds for red, green, and blue

In this image, we eliminated most noise from the background with a small amount remaining on the top side of the image; however, if we further reduce the boundaries of the thresholds we eliminate parts of the faces as well. In order to capture all three faces with reasonable amount of noise, threshold boundaries must encapsulate the lowest and highest color tones. By narrowing the interval, we reduce the area of individuals' faces that are visible in the resulting binary image, because color values are getting excluded from the interval.

The opposite is also problematic: if we increase the boundaries of thresholds for improving face boundaries, general noise in the threshold increases as well. As we enlarge the boundaries of the threshold, we are able to encapsulate more area from individuals' faces, however we increase number of color tones that do not exist in individuals' faces as well. Thus, noise in the resulting binary image gradually increases as the interval enlarges.

To solve the problem described above, we first examined the color histogram of the original image and selected some intervals in combination with human skin tone's red-green-blue values[1]. Then by repeatedly experimenting with thresholds and applying different logical operations on them, we acquired the figure above. It is the result of applying logical *and* operation on all three thresholds without order, using the threshold boundaries explained before.

To summarize this part; we first applied the unique thresholds for each color, then combined the red and green thresholds with logical *and* operation for eliminating most noise from the image, and lastly combining all three thresholds for maximum noise elimination.

PART 2: CARS

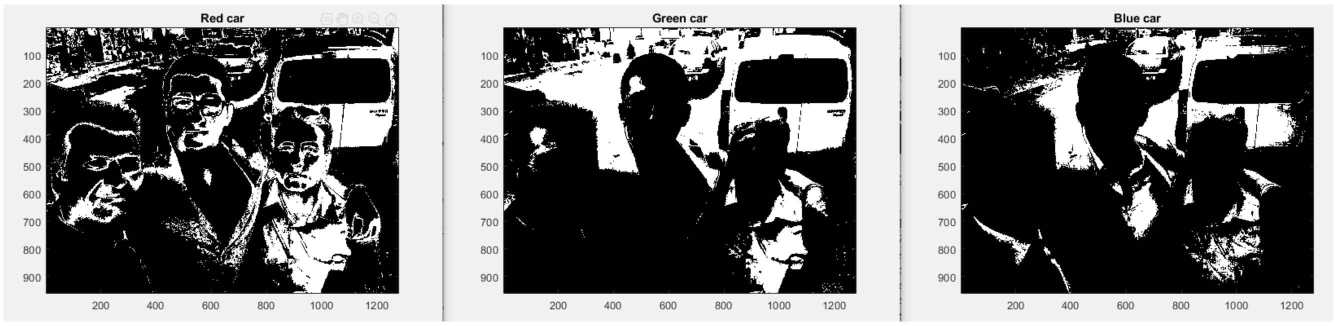

Figure 4: From left to right -> red, green, and blue thresholding

In this image, we can see the single-color thresholds for red, green, and blue. For red, threshold is $135 > image(:,:,1) > 100$, for green threshold is $200 > image(:,:,2) > 140$, and for blue threshold is $image(:,:,2) > 185$ where *image* is the original image being processed. In this section, our aim is to detect the white car on the right and the gray one at the top of the image with reasonable noise. We can see that the blue threshold eliminates individuals' faces well whereas the green one eliminates their clothing well. We can also see that, if we combine the two thresholds with logical *and* operation, we can reduce the noise in almost everywhere but the top part. So, we first apply logical *and* operation on the green and the blue threshold. Then, to minimize the noise in the top part of the image, we apply the logical *and* operation on the result of the previous operation and the red threshold. The resulting binary image is as follows:

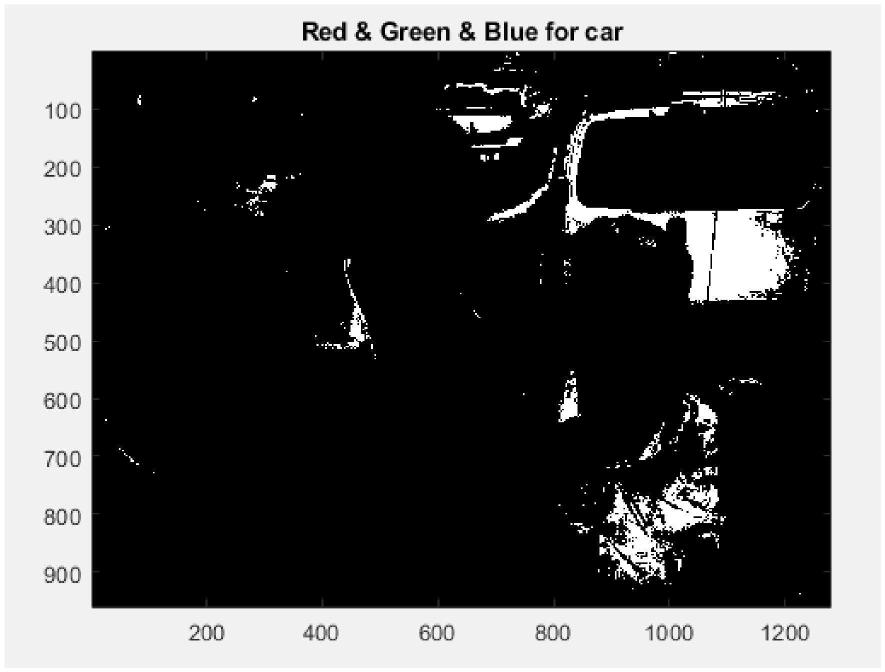

Figure 5: Combined binary thresholds for red, green, and blue

The threshold boundaries are selected as follows: by examining the color histograms we selected some initial intervals using Otto's method. Then, empirically we experimented with the threshold values for maximum noise elimination and detection rate.

Clearly, the resulting binary image has lots of noise. The reasons are as follows: firstly, the car on the right is partially block by a person. As the original image contains a blue filter, the person's hair and the color of the car are of similar tone. Secondly, the back window of the car on the right is black in the image. If we try to extract the parts that are black or white, general noise in the image would be high; since the threshold must be low enough to accept the black tones. Lastly, top part of the image has high brightness compared to other parts. Thus, color tones in the specified area are close to white. If we try to extract the gray car on the top with small amount of noise, we may not accurately capture the car on the right and vice versa. So, to capture both cars with reasonable accuracy, we need to have some noise in the picture.

To summarize this part; we first applied the unique thresholds for each color, then combined the green and blue thresholds with logical *and* operation for accurate detection of the car on the right, and lastly combined all three thresholds for maximum noise elimination.

PART 3: ACCESSORIES

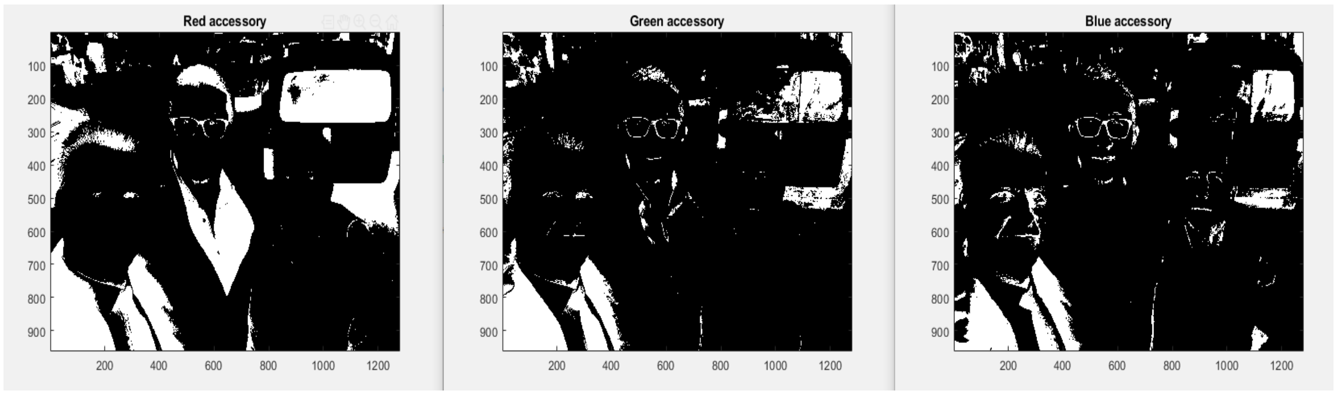

Figure 6: From left to right -> red, green, and blue thresholding

In this image, we can see the single-color thresholds for red, green, and blue. For red, threshold is $45 > image(:,:,1)$, for green threshold is $25 > image(:,:,2)$, and for blue threshold is $44 > image(:,:,2)$ where *image* is the original image being processed. Here, our aim is to detect the glasses of the individual on the middle and the tie of the one on the left. We can see that no image perfectly removes noise, so the best choice is to apply logical *and* operation on all three binary images for maximum noise removal. The result is as follows:

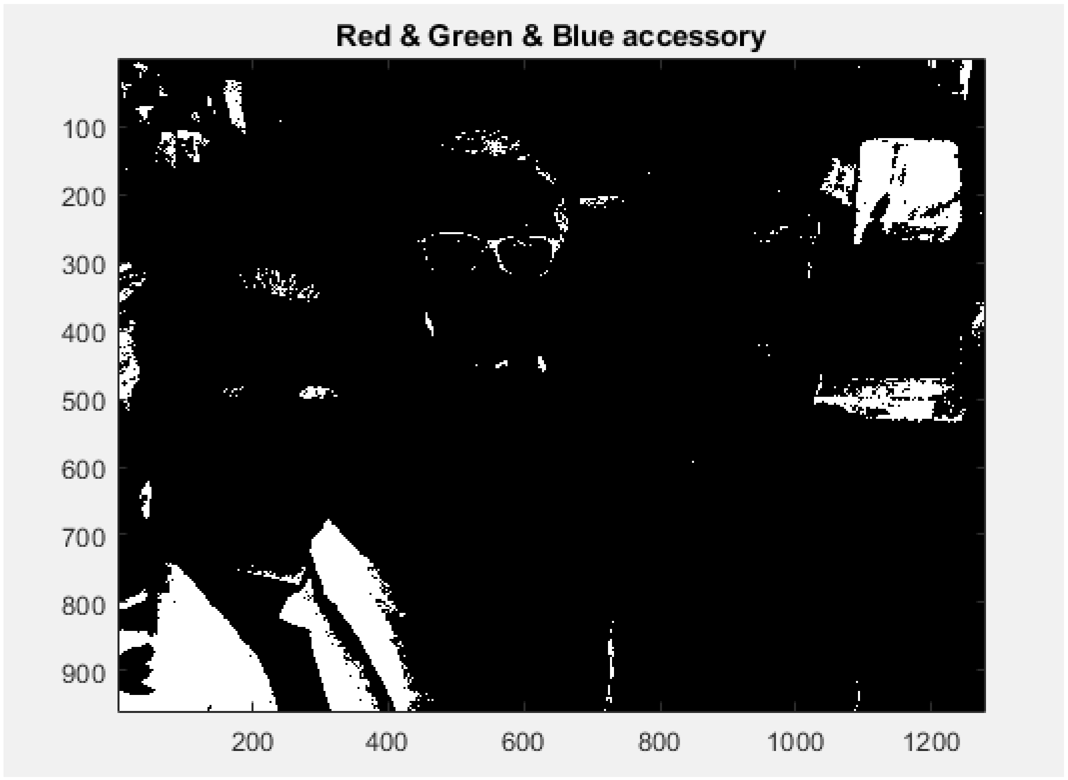

Figure 7: Combined binary thresholds for red, green, and blue

Since the color of these accessories are close to black, and the image is populated with black tones, the general noise is high. For example, the suit of the individual on the left and his tie are both accepted in all 3 thresholds, just as the back window of the car on the right. Since the difference in terms of color is too tight, if we try to narrow down the threshold intervals we lose accuracy. One solution to this problem might be to increase the brightness of the image, however, that requires further work.

To summarize this part; we first applied the unique thresholds for each color, then combined all three with logical *and* operation. Since the color of our targets can be found frequently throughout the image, general noise is high.

SUMMARY

- In part 1, we applied 3 unique color thresholds and obtained 3 binary images. For maximum noise elimination and accuracy, we applied logical and operation on them. However, because of the poor brightness settings, our accuracy is limited.
- In part 2, we applied 3 unique color thresholds and obtained 3 binary images. For maximum noise elimination and accuracy, we applied logical and operation on them. However, because of the poor brightness settings on the top section of the image and the color difference between our targets, our accuracy is limited.
- In part 2, we applied 3 unique color thresholds and obtained 3 binary images. For maximum noise elimination and accuracy, we applied logical and operation on them. However, since the color of our targets can be found frequently in the image and the tight threshold boundaries, our accuracy is limited.

DISCUSSION

In this report, we experimented with binary thresholding for object detection in images. Since the original image has poor quality and low contrast, binary thresholding performed poorly. Specifically, detecting blue colors were difficult, since the image contains a blue filter. Also, in areas where brightness settings are poor, such as the top side of the image colors shift towards white, which makes detecting objects in that region difficult. Also, detecting opposite colors and differentiating black objects one by one (as in black tie, black window, black glasses, black hair, etc.) were complicated as well. However, we believe the final results are satisfactory, since we were able to detect objects of interest with reasonable accuracy in limited amount of time. Also, because our goal was to partition an object from the background, binary thresholding is an effective choice of algorithm.

MATLAB SCRIPT

```
image = imread("image.jpg");
figure; imshow(image); axis image;
title("Original Image");

%Color histograms
redDistribution = image(:, :, 1);
greenDistribution = image(:, :, 2);
blueDistribution = image(:, :, 3);
[red, x] = imhist(redDistribution);
[green, x] = imhist(greenDistribution);
[blue, x] = imhist(blueDistribution);
plot(x, red, "Red", x, green, "Green", x, blue, "Blue"); % histograms

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Faces
redDistributionPeople = 200 > image(:, :, 1) & image(:, :, 1) > 125;
figure;
imagesc(redDistributionPeople); colormap(gray); axis image;
title("Red people");
```

```
greenDistributionPeople = 200 > image(:, :, 2) & image(:, :, 2) > 70;
figure;
imagesc(greenDistributionPeople); colormap(gray); axis image;
title("Green people");

blueDistributionPeople = 150 > image(:, :, 3) & image(:, :, 3) > 30;
figure;
imagesc(blueDistributionPeople); colormap(gray); axis image;
title("Blue people");

merged = (redDistributionPeople & greenDistributionPeople) & blueDistributionPeople;
figure;
imagesc(merged); colormap(gray); axis image;
title("Red & Green & Blue people");
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cars
redDistributionCar = 135 > image(:, :, 1) & image(:, :, 1) > 100;
figure;
imagesc(redDistributionCar); colormap(gray); axis image;
title("Red car");

greenDistributionCar = 200 > image(:, :, 2) & image(:, :, 2) > 140;
figure;
imagesc(greenDistributionCar); colormap(gray); axis image;
title("Green car");

blueDistributionCar = image(:, :, 3) > 185;
figure;
imagesc(blueDistributionCar); colormap(gray); axis image;
title("Blue car");

merged2 = (redDistributionCar & greenDistributionCar) & blueDistributionCar;
figure;
imagesc(merged2); colormap(gray); axis image;
title("Red & Green & Blue for car");
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Acccessory
redDistributionAccessory = 45 > image(:, :, 1);
figure;
imagesc(redDistributionAccessory); colormap(gray); axis image;
title("Red accessory");

greenDistributionAccessory = 25 > image(:, :, 2);
figure;
imagesc(greenDistributionAccessory); colormap(gray); axis image;
title("Green accessory");

blueDistributionAccessory = 44 > image(:, :, 3);
figure;
imagesc(blueDistributionAccessory); colormap(gray); axis image;
title("Blue accessory");

merged3 = (redDistributionAccessory & greenDistributionAccessory) & blueDistributionAccessory;
figure;
imagesc(merged3); colormap(gray); axis image;
title("Red & Green & Blue accessory");
```

REFERENCES

[1] M. A. H. Akhand, "*RGB values for different human skin color tones",* 2016, in M. A. H. Akhand, "Improvement of Haar Feature Based Face Detection in OpenCV Incorporating Human Skin Color Characteristic,*" International Journal of Computer Applications & Information Technology,* vol. 1, no. 8, *2*016.