John Doe
CAP 5400 - Digital Image Processing
Assignment 1

Objective

The objective of this project was to experiment with 3 different types of thresholding: global thresholding, color thresholding, and adaptive thresholding. Each of these methods is to be performed within both a specific Region of Interest (ROI) and the entire image. Several different thresholds and regions of interest are to be demonstrated and compared.

Background

Global thresholding uses a single threshold (T) for the entire image. Meaning that the original image is scanned, and a decision is made for each pixel in the image based upon that threshold. This operation can be summarized by the following equation:

$$\hat{I}(i,j) = \begin{cases} 0 & \text{if } I(i,j) < T \\ 255 & \text{if } I(i,j) \geq T \end{cases}$$

The region of interest implementation for this operation adds a single comparison to the algorithm. The purpose of this comparison is to determine if the pixel being examined in the original image is within the ROI. If it is, the the global thresholding operation described above is applied. If not, then the pixel is simply copied to the output image. For the purposes of this project, the ROI is defined as square region in which an (x,y) position specifies the top-left pixel in the region and width increases to the right and the height increases downwards.

Color thresholding works similarly to global thresholding for gray level images. Some global threshold is defined, and each pixel in the image is examined individually using that threshold. The difference is that when working with RGB colors, you cannot simply specify a different threshold for each color component. Instead, a desired color (C) is chosen and the Euclidean distance (D) from color C to the color of the pixel being examined is calculated. If that distance is less than some threshold (T) defined by the user, then that pixel is set to white. Otherwise, the pixel is set to black. This works since the RGB color space is conceptualized as 3-dimensional rectangular coordinate system in which “similar” colors are clustered spatially.

$$D = \sqrt{(R - R_c)^2 + (G - G_c)^2 + (B - B_c)^2}$$

$$\hat{I}(i,j) = 0 \quad if D < T$$

$$\hat{I}(i,j) = 255 \quad if D \geq T$$

Adaptive (or variable) thresholding works differently than the other two thresholding methods evaluated in this project. For starters, it operates on a local neighborhood. It also

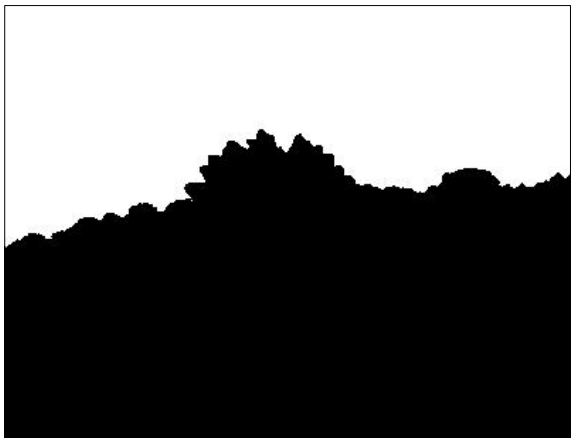
calculates the threshold for each pixels using the neighborhood information instead of using the one specified by the user. It does this by calculating the mean of each W by W neighborhood and adding a constant T. Both W and T are provided by the user. Thus, the threshold for each pixel is defined as:

$$Threshold = mean(W) + T$$

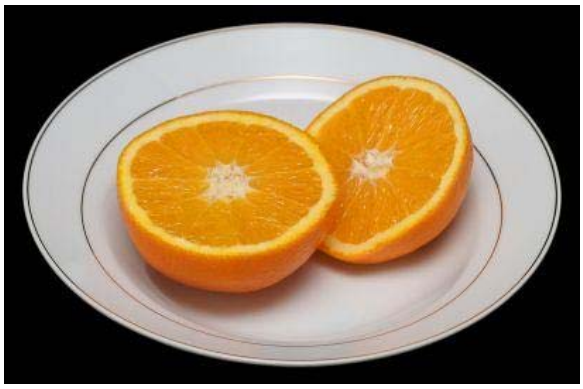
The overall goal of the adaptive approach is to provide a thresholding method that is less sensitive to the intensity variations that are inherently part of the image.

Results

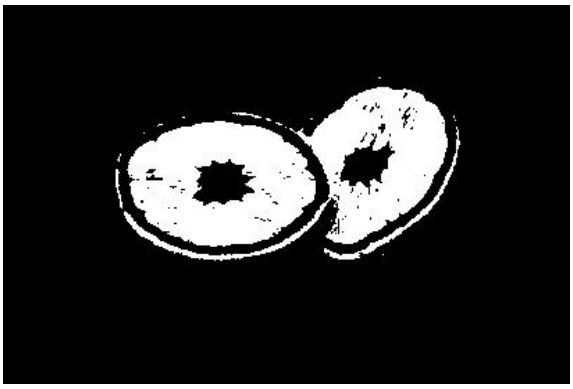
Original Image



RGB = (141,178,248)
T = 65



Original Image



RGB = (225, 145, 4)
T = 40

As can be seen by the above examples, it is very easy it isolate a color when there is a very distinct color belonging to a single object in the image. In the sunflower image it was the sky, and in the orange image it was the oranges themselves. These images are convenient for demonstrating color thresholding since they both posses some distinct color. However, if the color we want to isolate is more distributes throughout the image, the result is not a clean. For example,

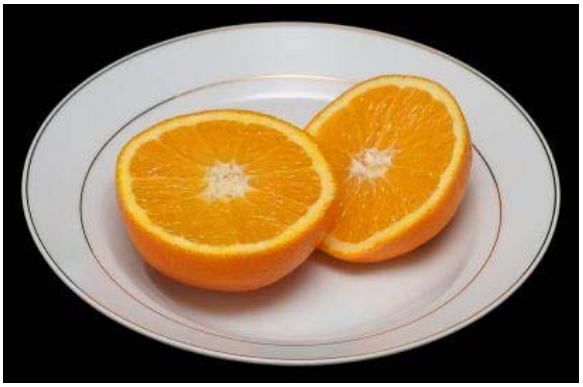
Original Image

RGB = (248,249,7)
T = 130

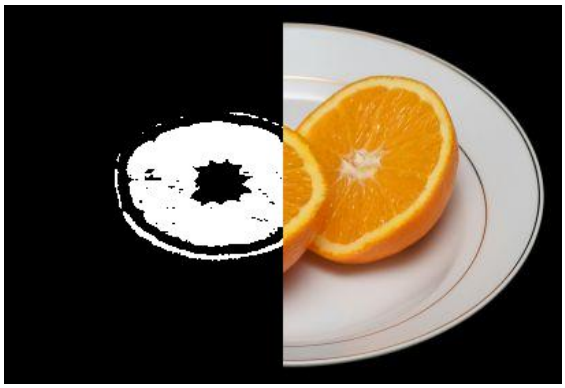
Original Image

RGB = (201, 22, 18)
T = 80

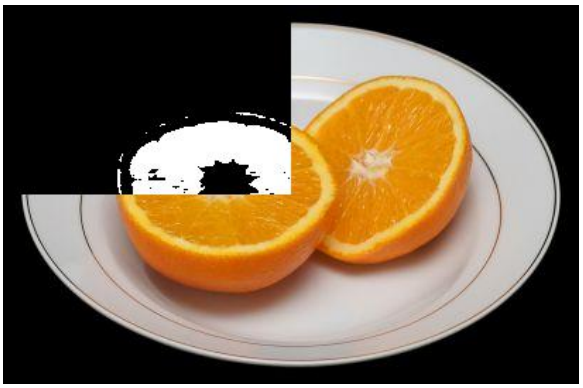
In the sunflower image, a bright yellow from the large sunflower was selected as the color to threshold for. As can be seen, a lot of other colors were picked up in the process. This is mostly because of the large threshold value of 130. However, if the threshold is lower than that the large flower itself isn't completely picked. This same problem can be seen in the dice image, where the red die was supposed to be isolated. As can be seen, the die itself was not isolated very well, and some other unrelated areas of the image were selected as well. This is where the regions of interest come in. If we know ahead of time that the die is going to appear in a specific portion of the image, we can specify an ROI that covers the area of the image we are interested in. Some examples of an ROI applied to both color and gray level images are shown below.



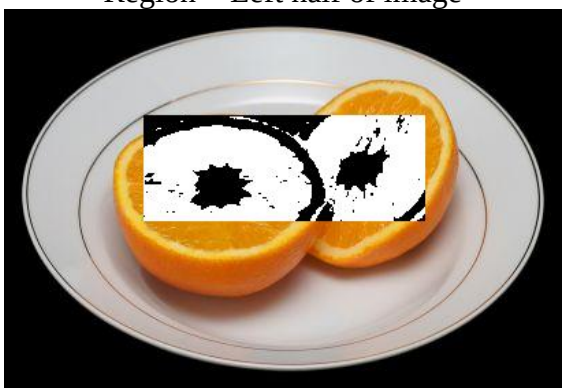
Original Image



RGB = (225, 145, 4)
T = 40
Region = Left half of image

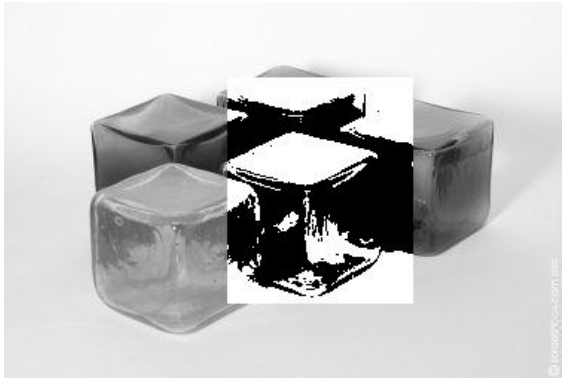


RGB = (225, 145, 4)
T = 40
Region = Top-left quarter of image



RGB = (225, 145, 4)
T = 40
Region = (102,75) top left pixel, width = 206 and height = 75

Original Image



T = 125
Region = (150,50) Top left pixel, width = 125, height = 150

Another problem with color thresholding is that colors that are conceptually the same, will not unnecessarily be seen as the same by the algorithm. For example, in the images below, the color blue, RGB (0,0,255), was specified as the color to isolate.


Original Image


RGB = (0, 0, 255)
T = 5


RGB = (0, 0, 255)
T = 50


RGB = (0, 0, 255)
T = 150

As the threshold goes up, more and more of the dark blue color we specified is detected. However, The cyan color between the green and blue “circles” is never detected; even when the threshold is as high as 150. Thus, it may be difficult for a user to isolate what they see as all the shades of blue in an image.

Finally, the experiments with adaptive thresholding yielded some interesting results. At first, a small value of T and a small value of W were used. This produced an image that appeared to be the result of an edge detector. At first, this didn't make sense, but upon further thought it became obvious that for small windows, most pixels would be close enough to the mean that they would be eliminated. However, those windows in which an edge is present contain a sufficiently large intensity range that some pixels would fall above the mean value, and thus be drawn white in the output image. To test this theory, the size of the window was increased significantly and an image that more closely represented one produced by the global thresholding algorithm resulted. As expected, this image that was produced using variable thresholding retained much more detail than the the image produced using the simple global thresholding method. This was expected because the variable thresholding method is much less dependent on consistent lighting conditions.


Original Image


Simple Thresholding
T = 125


Variable
Thresholding
T = 5
W = 5


Variable
Thresholding
T = 5
W = 15

Conclusions

Most of the algorithms performed as expected. The exception was the variable thresholding algorithm, which produced an initial result that was unexpected. That is, for small W and T, variable thresholding will produce an output image that looks like the output of an edge detector. In fact, I'm willing to bet that there exists an edge detector that operates on similar principles.

Even though the initial results were unexpected, the variable thresholding algorithm held up to it's promise of retaining more detail during binarization. This is possible since the algorithm works with a neighborhood to anagrammatically determine a suitable threshold for each pixel, and is thus less dependent upon a particular lighting condition. This is evidenced in the violin picture in particular where the woman's hair does not show up at all when using global thresholding, but appears with some detail when variable thresholding is used.

As mentioned above, color thresholding works well when the object you want to isolate is composed of a color that is unique, and varies very little in shade. However, if there are other objects of similar color present in the scene, it will be very hard to separate them from the object you want to work with. them. Also, if lighting conditions cause the object of interest to be unevenly illuminated, it may be difficult to extract the object in its' entirety. This is because the uneven illumination may cause the object to take on a wide range of shades. This problem is evidenced in the rgb image above when color thresholding failed to extract any of the cyan color, regardless of the fact that it was closely related to the blue that was being extracted.

Finally, an incremental implementation of the mean calculation used in variable thresholding was not attempted for this project. It was simply not needed as the computational time never exceeded reasonable expectations. However, if this operation were to be applied to a video file, it would become very important to speed up the calculation. Perhaps this will be shown in a later project.