A7 – Image Segmentation

Introduction

In image segmentation, a region of interest (ROI) is picked out from the rest of the image (background) such that further processing can be done on it. Selection rules are based on features unique to the ROI.

In grayscale images segmentation can be done through thresholding if the ROI has a distinct grayscale range from the background. Consider the image shown in Figure 1. It is desired to pick out the handwritten text.

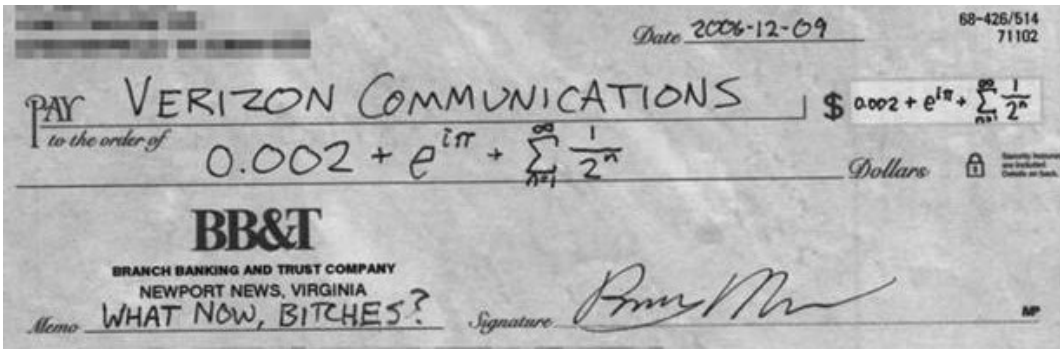

Figure 1: . Grayscale image of a check.

Try this short exercise:

1. Take a look at its grayscale histogram.

```
I = imread('cropped_grayscale_check.jpg');
[count, cells] = imhist(I, 256);
plot (cells, count);
```

2. From the histogram shown in Figure 2 the large peak corresponds to the background pixels. They occupy more space than the text. Choose a threshold.

```
BW = I < 125;
imshow (BW);
```

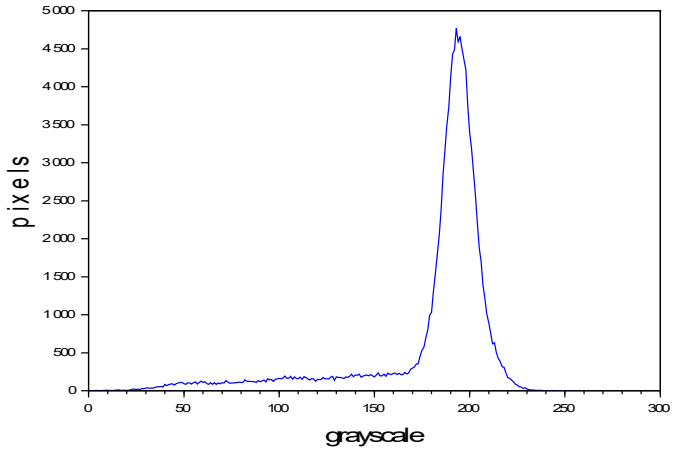

Figure 2: Histogram of check image.

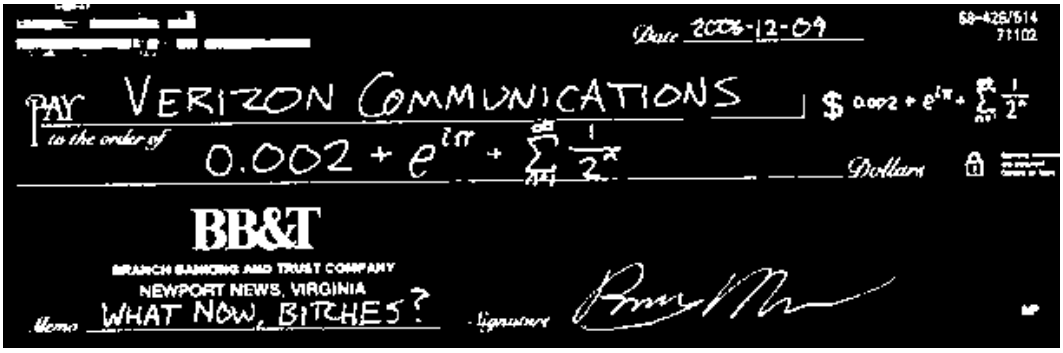

Figure 3: Segmented image of check.

Now the text is clearly picked out from the background and further processing can be done on it. Try what happens when you use different thresholds.

As you have seen, segmenting regions of interest can easily be done in grayscale images by thresholding but it is sometimes not possible to do so. Figure 4 shows the original and grayscale version of the same image. If our task is to segment the rust-colored box in the middle of the image, we can not do it in grayscale since the box has the same grayscale value as the background upon conversion.

Figure 4. The rust-colored box is difficult to segment in grayscale.

Color has been used to segment images of skin regions in face and hand recognition, land cover in remote sensing, and cells in microscopy.

Figure 5: Color of real 3D objects will have shading variations.

Inherently, 3D objects will have shading variations. To some extent shadows can be seen as differing brightness levels of the same color. For example, a red ball such as in Figure 5 will appear to have various shades of red from top to bottom. For this reason, it is better to represent color space not by the RGB but by one that can separate brightness and chromaticity (pure color) information. One such color space is the ***normalized chromaticity coordinates*** or NCC.

Per pixel, let I = R+G+B. Then the normalized chromaticity coordinates are

$$r=\frac{R}{R+G+B}=\frac{R}{I} \quad , \quad g=\frac{G}{I} \quad , \quad b=\frac{B}{I} \quad . \qquad (1)$$

We note that r+g+b = 1 which implies r,g and b can only have values between 1 and 0 and b is dependent on r and g since b = 1-r-g. Therefore, it is enough to represent chromaticity by just two coordinates, r and g. Thus, from R G B, the color space has been transformed to r g I where chromatic information is in r and g while brightness information is in I. We have thus reduced color information from 3 dimensions to 2. Thus, when segmenting 3D objects it is better to first transform RGB into rgI Figure 6 shows the r-g color space. When r and g are both zero, b = 1. Therefore, the origin corresponds to blue while points corresponding to r=1 and g=1 are points of pure red and green, respectively. Any

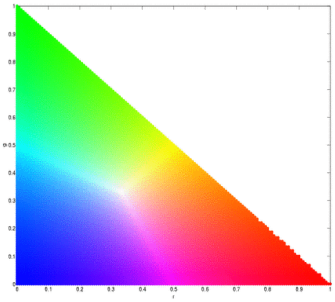

Figure 6. Nomalized chromaticity space. X-axis is r and y-axis is g.

color therefore can be represented as a point in NCC space. Thus the pixel chromaticities of the red ball in Figure 1 will only occupy a small blob in the red region regardless of its shading variation.

Parametric vs Non-Parametric Probability Distribution Estimation

Segmentation based on color can be performed by determining the probability that a pixel belongs to a color distribution of interest. This implies that the color histogram of the region of interest must first be extracted. To do so, one crops a subregion of the ROI and compute the histogram from it. The histogram when normalized by the number of pixels is already the probability distribution function (PDF) of the color. To tag a pixel as belonging to a region of interest or not is to find its probability of belonging to the color of the ROI. Since our space is r and g we can have a joint probability p(r) p(g) function to test the likelihood of pixel membership to the ROI. We can assume a Gaussian distribution independently[1] along r and g, that is, from the r-g values of the cropped pixel we compute the mean μ_r, μ_g and standard deviation σ_r, σ_g from the pixel samples. The probability that a pixel with chromaticity r belongs to the ROI is then

$$p(r) = \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left\{ -\frac{(r-\mu_r)^2}{2\sigma_r^2} \right\} \qquad (2)$$

A similar equation is computed for g. The joint probability is taken as the product of p(r) and p(g).

Histogram backprojection

Fitting an analytic function to the histogram is one way to get a pixel-membership function however it is only as good as the fit of the PDF function assumed. In non-parametric estimation, the histogram itself is used to tag the membership of pixels. Histogram backprojection is one such technique where based on the color histogram, a pixel is given a value equal to its histogram value in chromaticity space. This has the advantage of faster processing because no computations are needed, just a look-up of histogram values.

Obtaining a 2D histogram can be implemented rapidly by converting the r,g

1 We may also assume a joint distribution that considers r and g as components of a vector but to simplify calculations we first consider 1-D distributions each for r and g.

values into integers and binning the image values in a matrix. The matrix can then be concatenated into a look-up table with the indices being a function of the bin indices for r and g.

Below is a sample code for creating a 2D histogram in Scilab.

When the Macbeth chart shows up, use the mouse to select a color patch. Enter 'y' if you want to repeat or 'n' to stop the program.

```
//Maricor Soriano 2017.09.27
//Displays the Macbeth Chart, waits for user to select a region,
// and displays rg histogram of that region.

J = imread('Gretag-Macbeth_ColorChecker.jpg');
go_on = 'y';
while go_on ~= 'n' do

    subplot(1,2,1);
    I = imcropm(J);
    I = double(I); //I is the image of the region of interest
    R = I(:,:,1); G = I(:,:,2); B = I(:,:,3);
    Int= R + G + B;
    Int(find(Int==0))=100000; //to prevent NaNs
    r = R./ Int; g = G./Int;
    BINS = 32;
    rint = round( r*(BINS-1) + 1);
    gint = round (g*(BINS-1) + 1);
    colors = gint(:) + (rint(:)-1)*BINS;
    hist = zeros(BINS,BINS);
    for row = 1:BINS
    for col = 1:(BINS-row+1)
    hist(row,col) = length( find(colors==( ((col + (row-
1)*BINS)))));
    end;
    end;
    subplot(1,2,2)
    Matplot(imrotate(hist*255/max(hist),90));
    f = gcf(); f.color_map = rainbowcolormap(64)

    go_on = input("Again? y/n","s");
    if go_on=='n' then break end;

end;
```

Procedure

1. Capture or download a digital image of a 3D object with parts that have single color (e.g. colored mugs, faces, hands or any exposed skin, brightly colored bottles, etc.).
2. Crop a monochromatic region of interest in the scene.

3. Transform color RGB into normalized chromaticity coordinates.
4. Parametric segmentation – derive the Gaussian PDF in the r and g values separately of the ROI and segment the whole image.
5. Non-parametric segmentation – Obtain the 2D histogram of the ROI. To test if your histogram is correct, compare the location of the peaks with the rg chromaticity diagram in Figure 6. Justify your results. Use the histogram itself to segment the image using histogram backprojection.
6. In your results and discussion, compare the outcomes of the two techniques.

