Kyungpook National University

Artificial Brain Research - ABR

BEP

Lesson and Assignment 4

Thresholding and Otsu

Gwenaelle Cunha Sergio

Winter 2014

1 Summary of Lecture Material

1.1 Thresholding

Application of thresholding methods are: binarization, contrast emphasize and image enhancement. Here, we'll explore binarization. Binarization is used to represent an image with pixels of 1 bit instead of 8. It's used to a pre-process an image to find features in it and differ background from object.

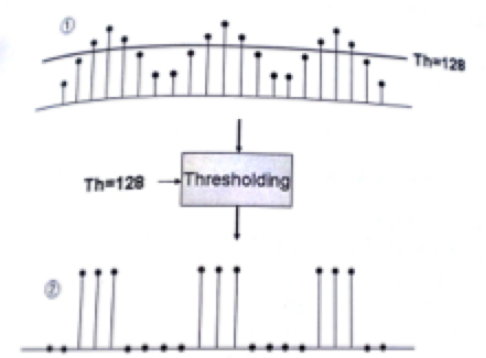

Figure 1: Signal

The thresholding can be static, threshold value T is constant regardless of the image, or dynamic, threshold value T can be changed based on the image or methods (1/2, p-tile, iterative, otsu, valley emphasis).

1.1.1 Static binarization

In this method, the resulting image *g(x,y)* depends on the following relation:

$$g(x,y) = \begin{cases} 1 & f(x,y) > T \\ 0 & otherwise \end{cases}$$

Fig. 2 shows a few examples of applying static binarization to a given image.

1.1.2 P-tile and 1/2 Binarization

The first step in this method is to calculate the number of pixels in the image. For p-tile binarization, the threshold is then 1/p of that number. The 1/2 binarization is a singularity of the p-tile, where p = 2, meaning that the threshold value is half of the total number of pixels.

Fig. 3 shows a few examples of applying p-tile and 1/2 binarization to a given image.

Figure 2: Static Binarization

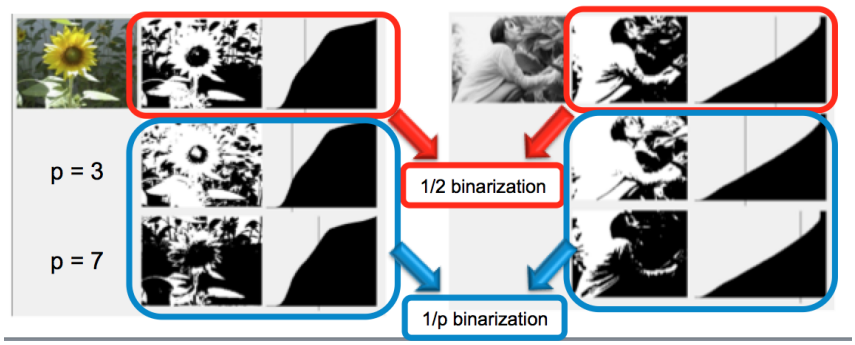

Figure 3: P-tile and 1/2 Binarization

1.1.3 Iterative Binarization

The steps to execute the method, exemplified in **Fig. 4**, are:

- Define default threshold point T, which is usually the average of the total intensity;
- The image is divided into the two group, $C1$ and $C2$, according to the selected T;
- Each group, $C1$ and $C2$, calculates its corresponding average of intensity, μ_1 and μ_2;
- The new threshold value is determined using the below equation.

$$T' = (\mu_1 + \mu_2)/2$$
$$\text{where } \mu_1 = mean(C1) \text{ and } \mu_2 = mean(C2)$$

The restrictive condition can be one of the following:

1. T-T' $\leq$ specific value
2. Specific number of iterative

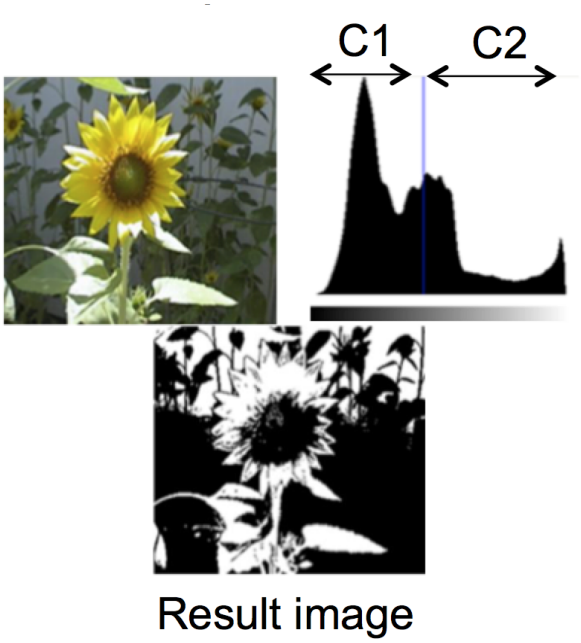

Figure 4: Iterative Binarization

1.1.4 Otsu

This method assumes that the image contains two classes of pixels: foreground and background. Given that, it "calculates the optimum threshold separating those two classes so that their combined spread, or intra-class variance (see equation below), is minimal." [1]

[1] $http://en.wikipedia.org/wiki/Otsu's\_method$

$$\sigma_{\omega}^2 = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

where ω_i are the weights, probabilities of the two classes separated by a threshold t, and the σ_i^2 are the variances of these classes. See **Fig. 5** for an example.

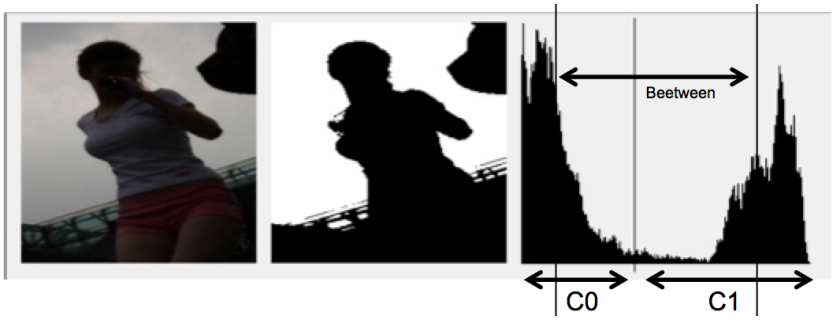

Figure 5: Otsu

1.1.5 Valley Emphasis

The Valley Emphasis method, see **Fig. 6**, is a modification of the Otsu method that selects a threshold value that has a small probability of occurrence, which is a valley in the gray-level histogram [2]. The method achieves its objective by adding a weight $1 - p(t)$ in the Otsu formula and maximizing the between group variance, just like the Otsu method.

$$t* = ArgMax_{0 \leq t < L}\{(1 - p_t)(\omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t))\}$$

Figure 6: Valley Emphasis

[2] http://msn.iecs.fcu.edu.tw/report/data/ori_paper/2006-12-21/2006-Automatic thresholding for defect detection.pdf

2 Assignment

2.1 Implement a binarization method

2.1.1 Static

The algorithm for the static binarization method is as follows:

```
cv::vector<cv::Mat> staticBin;
for (int i = 0; i < 3; i++) {
    staticBin.push_back(cv::Mat(mat.size(), mat.type()));
}
int T[3] = {20, 100, 200};
for (int t = 0; t < 3; t++){
    for (int i = 0; i < mat.rows; i++){
        for (int j = 0; j < mat.cols; j++){
            if (mat.at<uchar>(i,j) > T[t]) {
                staticBin[t].at<uchar>(i,j) = 255;
            } else {
                staticBin[t].at<uchar>(i,j) = 0;
            }
        }
    }
}
```

Fig. 7 shows a demonstration of its execution. From top left to bottom right, we have the original image and the resulting images for the threshold equals to 20, 100 and 200, respectively.

2.1.2 P-tile(1/2)

The algorithm for the P-tile binarization method is as follows:

```
cv::vector<cv::Mat> ptileBin;
for (int i = 0; i < 3; i++) {
    ptileBin.push_back(cv::Mat(mat.size(), mat.type()));
}
int p[3] = {2, 3, 7};
int pixels = mat.cols;
int T[3] = {pixels/p[0], pixels/p[1], pixels/p[2]};
for (int t = 0; t < 3; t++){
    for (int i = 0; i < mat.rows; i++){
        for (int j = 0; j < mat.cols; j++){
```

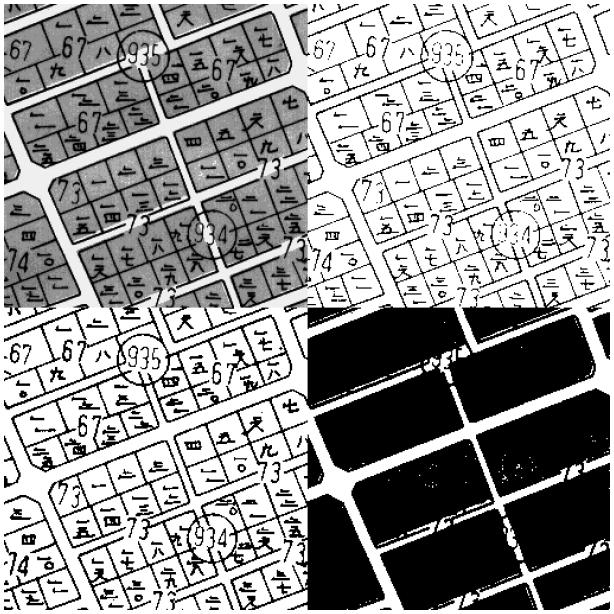

Figure 7: Static binarization result

```
                if (mat.at<uchar>(i,j) > T[t]) {
                    ptileBin[t].at<uchar>(i,j) = 255;
                } else {
                    ptileBin[t].at<uchar>(i,j) = 0;
                }
            }
        }
    }
```

Fig. 7 shows a demonstration of its execution. From top left to bottom right, we have the original image and the resulting images for p equals to 2 (2-tile), 3 and 7, respectively.

2.1.3 Iterative

The algorithm for the iterative binarization method is as follows, and **Fig. 9** shows a demonstration of its execution.

- Calculate default T:

```
int sum = 0, T = 0;
int total = mat.rows*mat.cols;
for (int i = 0; i < mat.rows; i++){
    for (int j = 0; j < mat.cols; j++){
        sum += mat.at<uchar>(i,j);
    }
}
T = cvRound(sum/total);
```

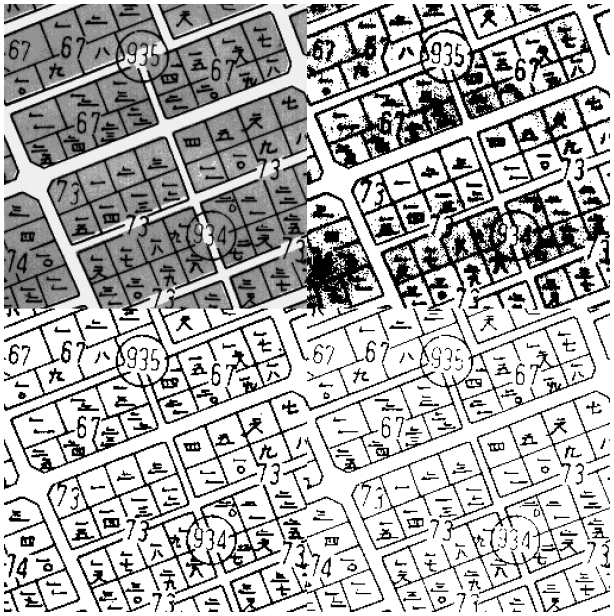

Figure 8: P-tile binarization result

- Calculate μ_1 and μ_2:

```
int sum2 = 0, mu1 = 0, mu2 = 0, cont1 = 0, cont2 = 0;
for (int i = 0; i < mat.rows; i++){
    for (int j = 0; j < mat.cols; j++){
        if (sum2 < sum/2) {
            mu1 += mat.at<uchar>(i,j);
            sum2 = mu1;
            cont1++;
        } else {
            mu2 += mat.at<uchar>(i,j);
            sum2 += mat.at<uchar>(i,j);
            cont2++;
        }
    }
}
mu1 /= cont1;
mu2 /= cont2;
```

- Calculate new T and obtain resulting image:

```
T = cvRound((mu1+mu2)/2);

for (int i = 0; i < mat.rows; i++){
    for (int j = 0; j < mat.cols; j++){
```

```
            if (mat.at<uchar>(i,j) > T) {
                iterativeBin.at<uchar>(i,j) = 255;
            } else {
                iterativeBin.at<uchar>(i,j) = 0;
            }
        }
    }
```

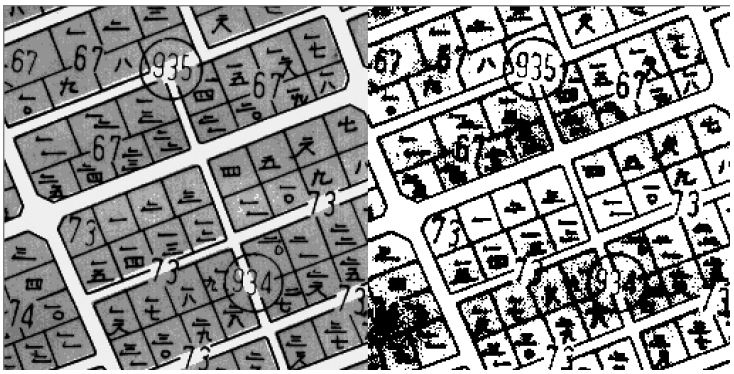

Figure 9: Iterative binarization result

2.1.4 Otsu

The algorithm for the static binarization method is as follows, and **Fig. 10** shows a demonstration of its execution.

```
cv::threshold(mat, otsuBin, 0, 255, CV_THRESH_BINARY | CV_THRESH_OTSU);
```

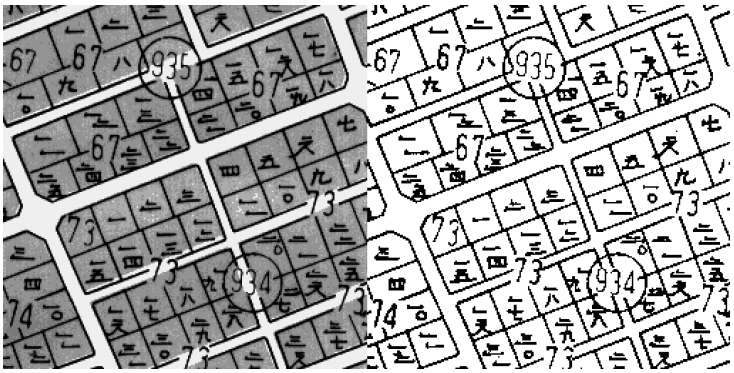

Figure 10: Otsu binarization result

2.2 Paper summarizing: Otsu's method

Summary of: OTSU, Nobuyuki. *"A Threshold Selection Method from Gray-Level Histograms"*. The paper in question presents a nonparametric and unsupervised method to automatically select a threshold for picture segmentation. It uses the zeroth and the first-order cumulative moments of the gray-level histogram to select the optimal threshold by the discriminant criterion.

Ideally, a histogram has a valley between two peaks, each representing objects and background. The threshold is in the bottom of this valley. However, in reality, it isn't always possible to detect this point with precision, if at all Sometimes the valley is too flat, or there's too much noise, or the peaks are too uneven producing no traceable valley.

There are some techniques to remedy those issues, like the valley sharpening technique, which restricts the histogram to the pixels with large absolute values of derivative; and the difference histogram method, which selects the threshold at the gray level with the maximal amount of difference.

So far, none of the methods mentioned have a criteria to evaluate the "goodness" of the threshold, and that's what's discussed in the paper. It assumes the elementary case of threshold selection, where only the gray-level histogram suffices without other previous knowledge. **Fig. 11** shows the result of applying this algorithm to an image.

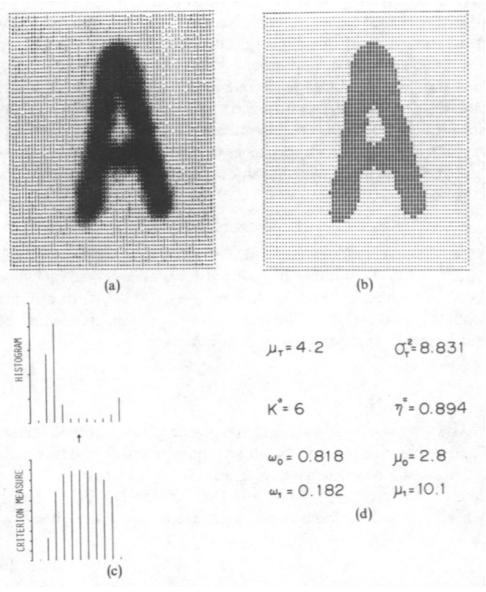

Figure 11: Paper example

2.3 Proof the Otsu's equation

Within class variance:

$$\sigma_\omega^2 = \omega_b(t)\sigma_b^2(t) + \omega_f(t)\sigma_f^2(t)$$

Take:

$$\mu = \omega_b\mu_b + \omega_f\mu_f$$
$$\omega_b + \omega_f = 1$$

Between class variance:

$$\begin{aligned}
\sigma_b^2(t) &= \sigma^2 - \sigma_\omega^2(t) \\
&= \omega_b(\mu_b - \mu)^2 + \omega_f(\mu_f - \mu)^2 \\
&= \omega_b(\mu_b^2 - 2\mu_b\mu + \mu^2) + \omega_f(\mu_f^2 - 2\mu_f\mu + \mu^2)
\end{aligned}$$

- Left side:

$$\begin{aligned}
\omega_b(\mu_b^2 - 2\mu_b\mu + \mu^2) &= \omega_b\mu_b^2 - 2\omega_b\mu_b(\omega_b\mu_b + \omega_f\mu_f) + \omega_b(\omega_b\mu_b + \omega_f\mu_f)^2 \\
&= \omega_b\mu_b^2 - 2\omega_b\mu_b(\omega_b\mu_b + \omega_f\mu_f) + \omega_b^3\mu_b^2 + \omega_f^2\mu_f^2\omega_b + 2\omega_b^2\mu_b\omega_f\mu_f \\
&= \mu_b^2(\omega_b - 2\omega_b^2 + \omega_b^3) - 2\omega_b\mu_b\omega_f\mu_f + \omega_f^2\mu_f^2\omega_b + 2\omega_b^2\mu_b\omega_f\mu_f \\
&= \omega_b\mu_b^2(1 - 2\omega_b + \omega_b^2) - 2\omega_b\mu_b\omega_f\mu_f(1 - \omega_b) + \omega_f^2\mu_f^2\omega_b \\
&= \omega_b\mu_b^2(1 - \omega_b)^2 - 2\omega_b\mu_b\omega_f^2\mu_f + \omega_f^2\mu_f^2\omega_b \\
&= \omega_b\mu_b^2\omega_f^2 - 2\omega_b\mu_b\omega_f^2\mu_f + \omega_f^2\mu_f^2\omega_b
\end{aligned}$$

- Right side:

$$\omega_f(\mu_f^2 - 2\mu_f\mu + \mu^2) = \omega_f\mu_f^2\omega_b^2 - 2\omega_f\mu_f\omega_b^2\mu_b + \omega_b^2\mu_b^2\omega_f$$

- Continuation of between class variance:

$$\begin{aligned}
\sigma_b^2(t) &= \omega_b(\mu_b^2 - 2\mu_b\mu + \mu^2) + \omega_f(\mu_f^2 - 2\mu_f\mu + \mu^2) \\
&= (\omega_b\mu_b^2\omega_f^2 - 2\omega_b\mu_b\omega_f^2\mu_f + \omega_f^2\mu_f^2\omega_b) + (\omega_f\mu_f^2\omega_b^2 - 2\omega_f\mu_f\omega_b^2\mu_b + \omega_b^2\mu_b^2\omega_f) \\
&= \omega_b\omega_f(\mu_b^2\omega_f - 2\mu_b\omega_f\mu_f + \omega_f\mu_f^2 + \mu_f^2\omega_b - 2\mu_f\omega_b\mu_b + \omega_b\mu_b^2) \\
&= \omega_b\omega_f[\omega_f(\mu_b^2 - 2\mu_b\mu_f + \mu_f^2) + \omega_b(\mu_f^2 - 2\mu_f\mu_b + \mu_b^2)] \\
&= \omega_b\omega_f[(\omega_f + \omega_b)(\mu_b^2 - 2\mu_b\mu_f + \mu_f^2)] \\
&= \omega_b\omega_f(\mu_b^2 - 2\mu_b\mu_f + \mu_f^2) \\
&= \omega_b\omega_f(\mu_b - \mu_f)^2
\end{aligned}$$

2.4 Survey another measuring threshold method

Another threshold method is called adaptive threshold [3], which takes an image as an input and outputs a binary image representing the segmentation. The threshold is calculated for each pixel in the image and if its value is below the threshold, it is set to the background value, otherwise it assumes the foreground value.

The main approach to find the threshold is the Chow and Kaneko approach. It assumes that the smaller the image regions, the more suitable they are for thresholding, since they are more likely to have an approximately uniform illumination. This method divides an image into an array of overlapping sub-images and then investigates its histogram to find the optimum threshold for each one. The threshold for each single pixel is then found by interpolating the results of the sub-images. This method's drawback is that it's very expensive computationally speaking and, therefore, is not appropriate for real-time applications.

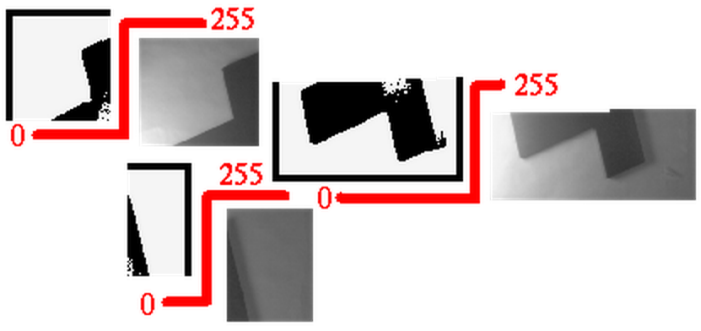

Figure 12: Adaptive Threshold

[3] http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm

2.5 Find short point using image processing

By applying the Otsu binarization method, we have as a result **Fig. 13**. From it, we can see that there are more than 200 points.

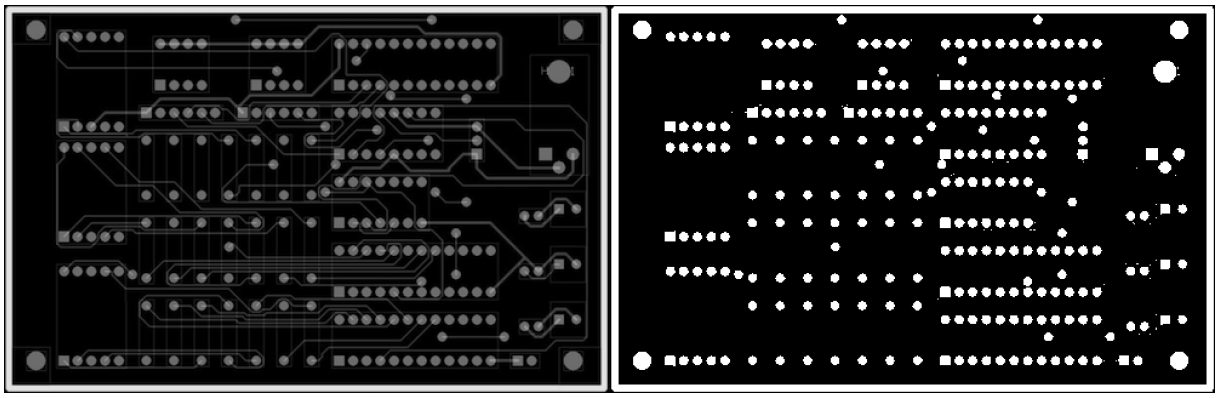

Figure 13: Short points