**Halftone image processing:**

**Segmentation by intensity levels:** it’s very adequate to select a grey level for threshold in order to do the segmentation, where in an ideal case the histogram has a deep and sharp valley between two peaks representing objects and background. However for most real pictures, it is often difficult to detect the valley bottom precisely, especially in such cases as when the valley is flat and broad, imbued with noise, or when the two peaks are extremely unequal in height, often producing no traceable valley. There are techniques proposed to overpass this difficulty as for e.g. *valley sharpening* technique which restricts the histogram to the pixels with large absolute values of derivatives, and the difference histogram method, which selects the threshold at the gray level with the maximal amount of difference.

For a picture with *L* intensity levels, where the number of pixels at a certain *i* level is *ni*, where the total number of pixels is *N = n1 + n2 + … + nL*, the grey level histogram is normalized and regarded as a probability distribution: 

If we dichotomize the pixels into two classes CO and C 1 (background and objects, or vice versa) by a threshold at level k; CO denotes pixels with levels [1… k], and C1 denotes pixels with levels [k + 1… L]. Then the probabilities of class occurrence and the class mean levels, respectively, are given by:



are the zeroth and the first-order cumulative moments of the histogram up to the kth level, respectively, and: is the total mean level of the original picture. We can easily verify the following relation for any choice of k: (9)

The class variances are given by:



In order to evaluate the "goodness" of the threshold (at level k), we introduce the following discriminant criterion measures (or measures of class separability) used in the discriminant analysis:



are the within-class variance, the between-class variance, and the total variance of levels, respectively. Then our problem is reduced to an optimization problem to search for a threshold k that maximizes one of the object functions (the criterion measures).

**Binarization of graphic images & Optimal thresholding methods to binarization:** Thresholding creates binary images from grey-level ones by turning all pixels below some threshold to zero and all pixels about that threshold to one. If g(x, y) is a thresholded version of f(x, y) at some global threshold T:



**Problems:** The major problem with thresholding is that we consider only the intensity, not any relationships between the pixels. There is no guarantee that the pixels identified by the thresholding process are contiguous. When thresholding is used, we have to play with it, sometimes losing too much of the region and sometimes getting too many extraneous background pixels.

**Local thresholding:** Another problem with global thresholding is that changes in illumination across the scene may cause some parts to be brighter (in the light) and some parts darker (in shadow) in ways that have nothing to do with the objects in the image. We can deal, at least in part, with such uneven illumination by determining thresholds locally.

**Automated methods for finding thresholds:** There are different ways to look at the problem:

* **Known distribution:** If you know that the object you’re looking for is brighter than the background and occupies a certain fraction 1/p of the image, you can set the threshold by simply finding the intensity level such that the desired percentage of the image pixels are below this value. This is easily extracted from the cumulative histogram:



Simply set the threshold *T* such that *c*(*T*) = 1*/p*. (Or, if you’re looking for a dark object on a light background, *c*(*T*) = 1 *−* 1*/p*.)

* **Finding peaks and valleys:** One extremely simple way to find a suitable threshold is to find each of the modes (local maxima) and then find the valley (minimum) between them. While this method appears simple, there are two main problems with it:
1. The histogram may be noisy, thus causing many local minima and maxima. To get around this, the histogram is usually smoothed before trying to find separate modes.
2. The sum of two separate distributions, each with their own mode, may not produce a distribution with two distinct modes.
* **Clustering (k-means variation):** We create two groups of pixels, one with one range of values and one with another. Thresholding is difficult in this case because these ranges usually overlap. We minimize the error of classifying a background pixel as a foreground one or vice versa, we do by minimizing the area under the histogram for one region that lies on the other region’s side of the threshold. The problem is that we don’t have the histograms for each region, only the histogram for the combined regions. A way to do this is to consider the values in the two regions as two clusters. The idea is to pick a threshold such that each pixel on each side of the threshold is closer in intensity to the mean of all pixels on that side of the threshold than the mean of all pixels on the other side of the threshold. In other words, let μB(T) be the mean of all pixels less than the threshold and μO(T) be the mean of all pixels greater than the threshold. We want to find a threshold such that the following holds:



* **Clustering (The Otsu method):** Another way is to set the threshold so as to try to make each cluster as tight as possible, thus minimizing their overlap. Obviously, we can’t change the distributions, but we can adjust where we separate them (the threshold). As we adjust the threshold one way, we increase the spread of one and decrease the spread of the other. The goal then is to select the threshold that minimizes the combined spread. We can define the *within-class* variance as the weighted sum of the variances of each cluster:



Computing this within-class variance for each of the two classes for each possible threshold involves a lot of computation, but there’s an easier way. If you subtract the within-class variance from the total variance of the combined distribution, you get something called the between-class variance: 

where *σ*2 is the combined variance and *μ* is the combined mean. Notice that the between-class variance is simply the weighted variance of the cluster means themselves around the overall mean. Substituting *μ* = *nB*(*T*)*μB*(*T*) +*nO*(*T*)*μO*(*T*) and simplifying, we get:



So, for each potential threshold *T* we:

1. Separate the pixels into two clusters according to the threshold.
2. Find the mean of each cluster.
3. Square the difference between the means.
4. Multiply by the number of pixels in one cluster times the number in the other.

This depends only on the difference between the means of the two clusters, thus avoiding having to calculate differences between individual intensities and the cluster means. The optimal threshold is the one that maximizes the between-class variance (or, conversely, minimizes the within-class variance).



* **Mixture modeling:** Another way to minimize the classification error in the threshold is to suppose that each group is Gaussian-distributed. Each of the distributions has a mean (μB and μO respectively) and a standard deviation (σB and σO respectively) independent of the threshold we choose:



*Mixture modeling* assumes that there already exists two distributions and we must find them. Once we know the parameters of the distributions, it’s easy to determine the best threshold.

If the two distributions are reasonably well separated (some overlap but not too much), we can choose an arbitrary threshold *T* and assume that the mean and standard deviation of each group approximates the mean and standard deviation of the two underlying populations. We can then measure how well a mix of the two distributions approximates the overall distribution:



Choosing the optimal threshold thus becomes a matter of finding the one that causes the mixture of the two estimated Gaussian distributions to best approximate the actual histogram (minimizes *F*). Unfortunately, the solution space is too large to search exhaustively, so most methods use some form of gradient descent method. Mixture modeling also extends to models with more than two underlying distributions (more than two types of regions).

* **Thresholding along boundaries:** If we want our method to give stay fairly true to the boundaries of the object, we can first apply some boundary-finding method and then sample the pixels only where the boundary probability is high. Thus, our threshold method based on pixels near boundaries will cause separations of the pixels in ways that tend to preserve the boundaries.