IDC410
A course on Image Processing and Machine Learning
(Lecture 03)

Shashikant Dugad,

IISER Mohali

Histogram Equalisation

Image Contrast: Histogram Equalization

- Consider a gray scale image of size **N x N**, thus, having N^2 pixels. Position of a pixel is denoted by (y, x)
- If the intensity, I(y, x), of a pixel is represented by m bits then.
  - the maximum intensity is $I_{max} = (2^m - 1)$
- Image contrast is optimized by doing smart intensity transformation function!
  - $I'(y,x) = T(I(x,y))$ T is a function of intensity transformation
- For a given original image, obtain intensity histogram, n = h(I), where, a) n is number of pixels having intensity I and b) range of intensity is $0 <= I < (2^m - 1)$ ➔Total # of bins in histogram are $(2^m - 1)$
  - Note: $\sum n = \sum h(I) = N^2$
- Probability of a pixel having intensity I is: $p(I) = \frac{n}{N^2}$ where, $\sum p(I) = 1$

Image Contrast: Histogram Equalization

- Let us apply following constraints on te intensity transformation function
  - T(I) must be a strictly increasing function. This makes it an injective (one-to-one) function.
  - $0 \leq T(I) \leq L-1$. This makes T(I) surjective.
- The above two conditions make T(I) a bijective function, thus inversible
  - Therefore, there exist a function that provides, $I = T^{-1}(I')$
- Cumulative Distribution Function (CDF) for input image can be defined as:

$$F_r(I) = P(r \leq I) = \sum_{i=0}^{I} p_r(i) = \frac{1}{N^2} \sum_{i=0}^{I} n_i$$

- CDF for final image can be written as,

$$F_s(x) = P(s \leq x) = P(T(r) \leq x) = P\left(r \leq T^{-1}(x)\right) = F_r(T^{-1}(x))$$

Image Contrast: Histogram Equalization

We put the first condition of $T(r)$ precisely to make the above step hold true. The second condition is needed as s is the intensity value for the output image and so must be between o and (L-1).

So, a pdf of s can be obtained by differentiating $F_S(x)$ with respect to x. We get the following relation:

$$p_s(s) = p_r(r)\frac{dr}{ds}$$

Now, if we define the transformation function as follows:

$$s = T(r) = (L-1)\int_0^r p_r(x)dx$$

Then using this function gives us a uniform pdf for s.

$$\frac{ds}{dr} = (L-1)\frac{d}{dr}\int_0^r p_r(x)dx = (L-1)p_r(r)$$

Image Contrast: Histogram Equalization

The above step used Leibnitz's integral rule. Using the above derivative, we get:

$$p_s(s) = p_r(r)\frac{\mathrm{d}r}{\mathrm{d}s} = p_r(r)\frac{1}{(L-1)p_r(r)} = \frac{1}{L-1}$$

So the pdf of s is uniform. This is what we want.

Now, we extend the above continuous case to the discrete case. The natural replacement of the integral sign is the summation. Hence, we are left with the following histogram equalization transformation function.

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{N^2}\sum_{j=0}^{k} n_j$$

Since s must have integer values, any non-integer value obtained from the above function is rounded off to the nearest integer.

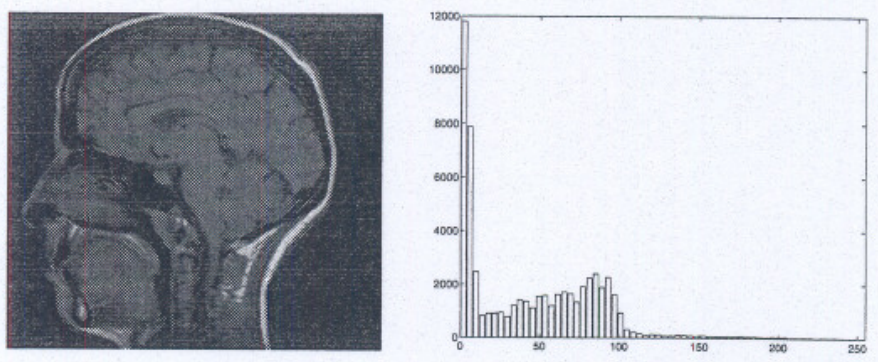
Histogram Equalization: Actual Calculation

Image Size: 8 x 9 **Pixel intensity: 3 bits** **L=8**

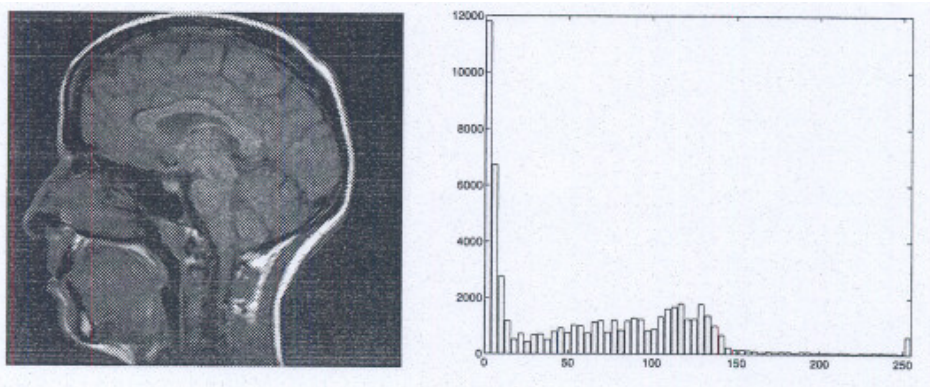
Pixel Intensity in Original Image (I)	Frequency (f)	I*f	Probbility Distribution Function (PDF)	Cumulative PDF (CPDF)	(L-1) *CPDF	Pixel Intensity (I') in Final Image	I' * f
0	2	0	0.02778	0.02778	0.19444	0	0
1	4	4	0.05556	0.08333	0.58333	1	4
2	6	12	0.08333	0.16667	1.16667	1	6
3	8	24	0.11111	0.27778	1.94444	2	16
4	10	40	0.13889	0.41667	2.91667	3	30
5	12	60	0.16667	0.58333	4.08333	4	48
6	14	84	0.19444	0.77778	5.44444	5	70
7	16	112	0.22222	1.00000	7.00000	7	112
Total =	**72**	**336**	**1**			**Total =**	**286**
Average Intensity =	**4.6667**					**Average Intensity =**	**4**

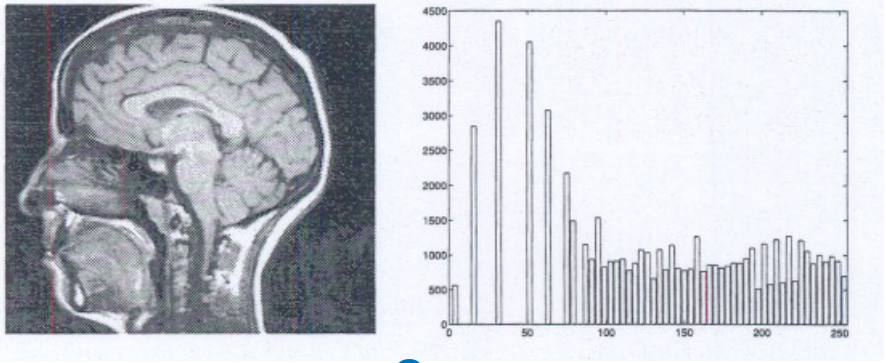
Image Contrast Results

A



B





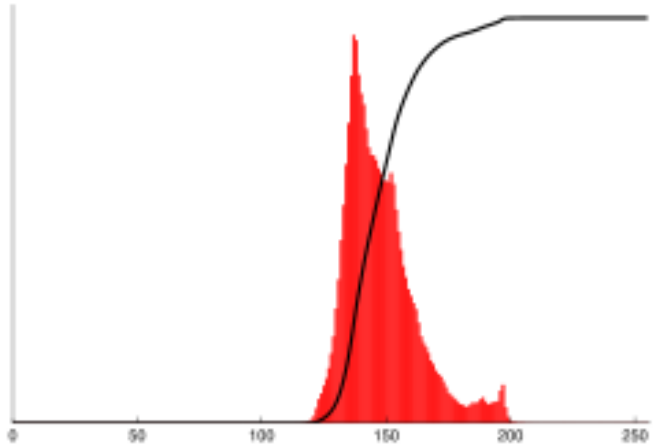
C

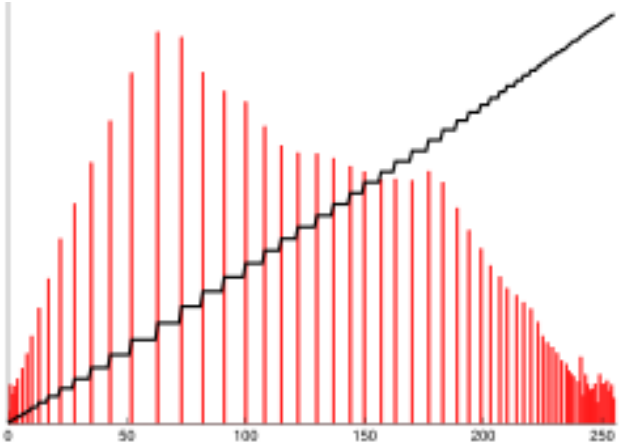
A: Original

B: Histogram Scaling

C: Histogram Equalization

Image before and after Histogram Equalization





Source: https://en.wikipedia.org/wiki/Histogram_equalization

Thresholding

Noise Removal: Thresholding

- **Binary Thresholding:**

$M(i, j) = 2^m - 1$ if pixel intensity $P(i, j) > p_0$

$M(i, j) = 0$ if pixel intensity $P(i, j) <= p_0$

- **Normal Thresholding:**

$M(i, j) = P(i, j)$ if pixel intensity $P(i, j) > p_0$

$M(i, j) = 0$ if pixel intensity $P(i, j) <= p_0$

- **Regional Thresholding:**

$M(i, j) = P(i, j)$ if pixel intensity $p_0 < P(i, j) < p_1$

$M(i, j) = 0$ if pixel intensity $P(i, j) <= p_0 \,||\, P(i, j) >= p_1$

- **Iterative Threshold**
- **Double Thresholding for Region Growing**
- **Otsu Thresholding**
- **Adaptive Thresholding**

Iterative Threshold

1. Select an initial estimate of the threshold μ, say, the average intensity of the image.

$$\mu = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} M(i,j)}{n * m}$$

2. Partition the image into two groups, R_1 and R_2, using the threshold μ .
3. Calculate the mean grey values μ_1 and μ_2 of the partitions R_1 and R_2
4. Select the new Threshold

$$\mu = \frac{\mu_1 + \mu_2}{2}$$

5. Repeat step 2-4 until μ_1 and μ_2 (therefore, μ) are approximately same in successive iterations

Double Thresholding for Region Growing

1. Select two thresholds μ_1 and μ_2 (You may use the prescription on previous slide to get μ_1 and μ_2)

2. Partition the image into three regions:
   - R1: containing all pixels with gray values below μ_1 ,
   - R2: containing all pixels with gray values between μ_1 and μ_2 inclusive and
   - R3, containing all pixels with gray values above μ_2 .

3. Visit each pixel assigned to region R2. If the pixel has a neighbour in region R1, then reassign the pixel to region R1

4. Repeat step 3 until no pixels are reassigned.

5. Reassign any pixels left in region R2 to region R3.

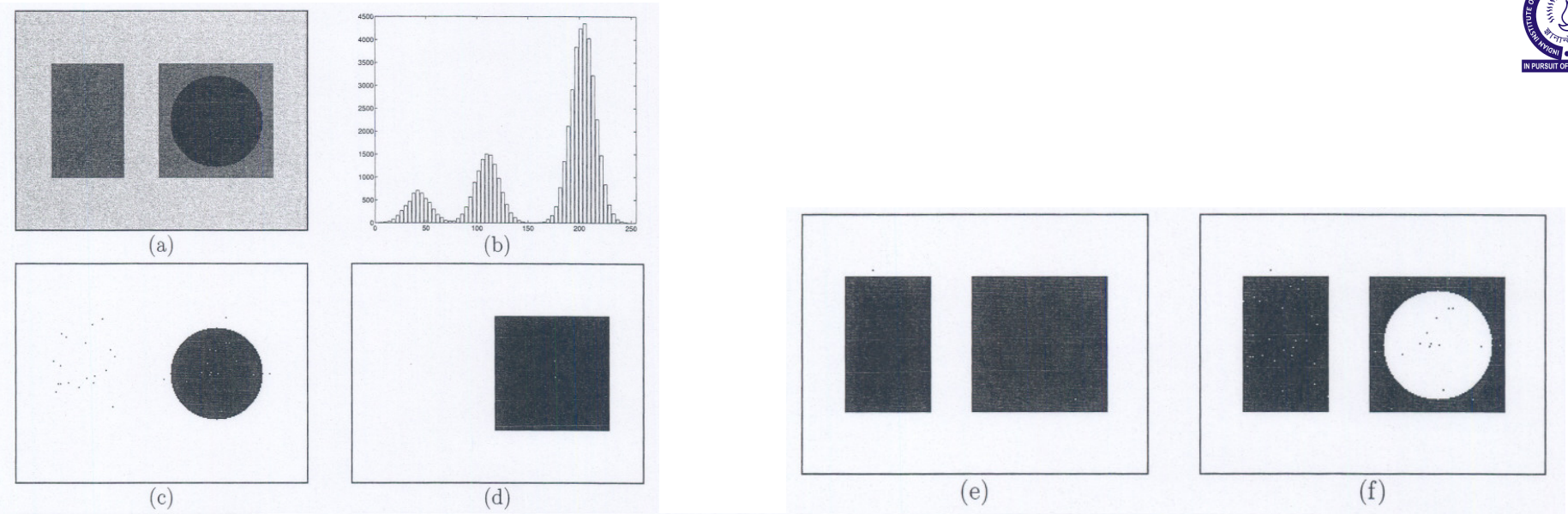


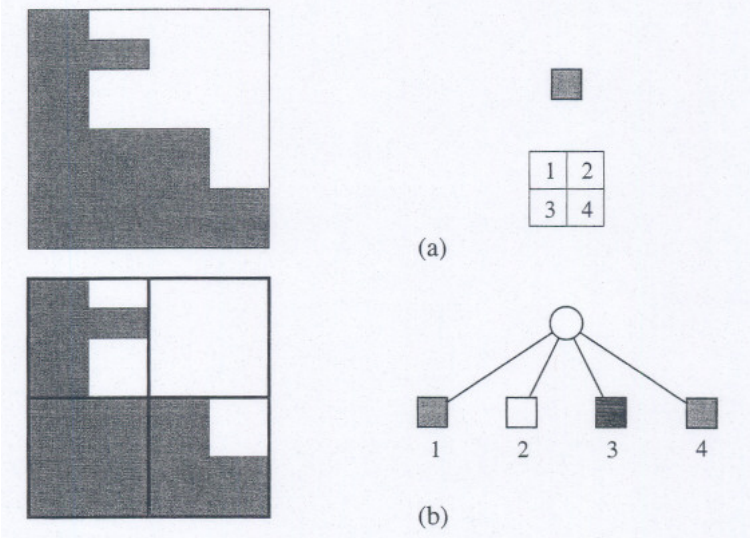
Figure 3.8: An image that is difficult to segment using a single threshold and its segmentation using double thresholding. (a) Original image. (b) Histogram of original image. (c) Core region, $T_1 = 70$. (d) Results after growing the core region using a second threshold, $T_2 = 155$. Compare this with the results using regular thresholding shown in (e) and (f). (e) Threshold $= T_2$. (f) $T_1 \leq$ Threshold $\leq T_2$. Note that it is impossible to segment the entire right square without using region growing threshold.

Regions

Regions are used in many contexts and can be represented in many alternative forms. Different representations are suitable in different applications. Some applications require computations only for a single region, while others require relationships among different regions of an image. In this section, we will discuss a few commonly used representations of regions and study their features. It must be mentioned here that regions can also be represented as closed contours.

Region: Quad Tree

A quad tree contains three types of nodes: white, black, and gray. A quad tree is obtained by recursive splitting of an image. A region in an image is split into four subregions of identical size, if all points in the region are either white or black, then this region is no longer considered as a candidate for splitting; if it contains pixels of both kinds, it is considered to be a **gray-region** and is further split into four subregions. An image obtained using this recursive splitting is represented in a tree structure. The splitting process is repeated until there are no gray regions in the tree. Each node in this structure is either a leaf node or has four children, thus the name *quad tree.*



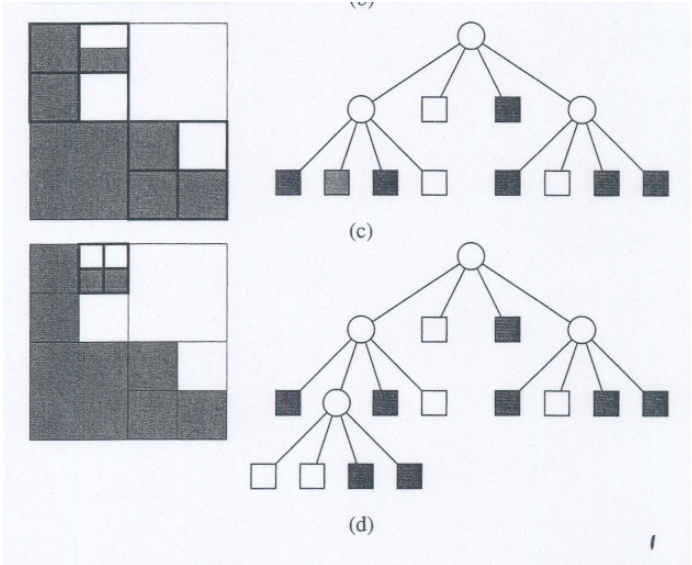


Figure 3.10: The building of a quad tree. (a) Original image, "gray region." (b) Original split into four subregions (the left node in the tree corresponds to the top left region in the image). Note that two of these regions are also gray regions. (c) Splitting the gray regions from (b) into four subregions. One of these regions is still a gray region. (d) Splitting of the last gray region and the final quad tree.

Otsu Thresholding Algorithm

- **Source: https://en.wikipedia.org/wiki/Otsu%27s_method**
- **In image processing, Otsu's method, named after Nobuyuki Otsu, is used to perform automatic image thresholding.**
- **The algorithm returns a single intensity threshold that separate pixels into two classes, foreground and background.**
- **The threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance.**

Otsu Thresholding Algorithm

- **The algorithm exhaustively searches for the threshold that minimizes the intra-class variance, (maximises intra class variance) which is defined as a weighted sum of variances of the two classes**:

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

- **Weights ω_0 and ω_1 are the probabilities of the two classes separated by a threshold t, σ_0^2 and σ_1^2 are variances in each of of these two classes**

$$p(i) = \frac{n_i}{N \times M}. \qquad \omega_0(t) = \sum_{i=0}^{t-1} p(i). \quad \text{and.} \quad \omega_1(t) = \sum_{i=t}^{L-1} p(i) \quad \text{and.} \quad \omega_0(t) + \omega_1(t) = 1$$

Otsu Thresholding Algorithm

- μ_0, μ_1 and μ_T are the means of regions a) below threshold t, b) above threshold t and entire region respectively,

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)} \qquad \mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)} \qquad \mu_T = \sum_{i=0}^{L-1} ip(i)$$

- For 2 classes, minimizing the intra-class variance is equivalent to maximizing inter-class variance

$$\sigma_b^2(t) = \sigma^2 - \rho_\omega^2(t) = \omega_0(t)(\mu_0 - \mu_T)^2 + \omega_1(t)(\mu_1 - \mu_T)^2.$$
$$= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

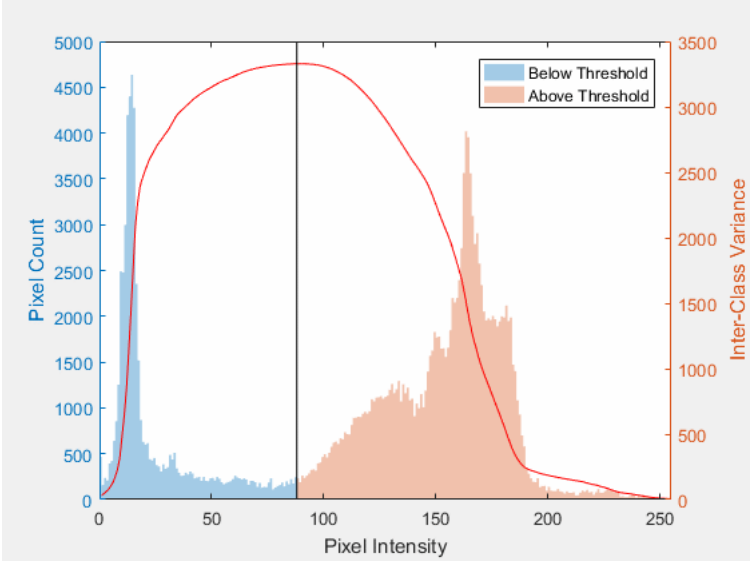
Otsu Thresholding: Implementation

- **Step 1: Compute histogram of a given image and obtain probabilities [*p(i)*] at each intensity level *i***
- **Step 2: Obtain ω_0, ω_1, μ_0 and μ_1 (hence σ_b) at each intensity value of t varying between 0 to L-1**
- **Select *that* value of t as the final threshold at which $\sigma_b(t)$ has maximum value**

By Pikez33
http://en.wikipedia.org/wiki/File:Image_processing_post_otsus_algorithm.jpg,
https://commons.wikimedia.org/w/index.php?curid=10634740

Original image

An example image thresholded using Otsu's algorithm



By Lucas(CA) - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=67144384

Adaptive Thresholding

- **ADAPTIVE_THRESH_MEAN_C:** Threshold value is the mean of neighbourhood area.
- **ADAPTIVE_THRESH_GAUSSIAN_C:** Threshold value is the weighted sum of neighbourhood values where weights are a gaussian window.