Technical University of Cluj - Napoca
Computer Science Department

Image Processing

(Year III, 2-nd semester)

Binary Images: Tresholding (I)

Binary Images: Simple Geometrical Properties(II)

Presentation outline

- **Introduction**
- Thresholding methods
- Otsu Method
- Threshold selection by best approximation with a two level image

- Binary Images: geometric properties
- Projections
- Run-Length Coding

Binary Image Processing

INTRODUCTION

The simplest type of image which is used widely in a variety of industrial and medical applications is **binary**, i.e. a black-and-white or silhouette image.

Advantages

-Easy to acquire: simple digital cameras; thresholding may be applied to grey-level images.
-Low storage: no more than 1 bit/pixel, often this can be reduced by compression (e.g. run-length coding).
-Simple and fast processing: the algorithms are in most cases much simpler than those applied to grey-level images.

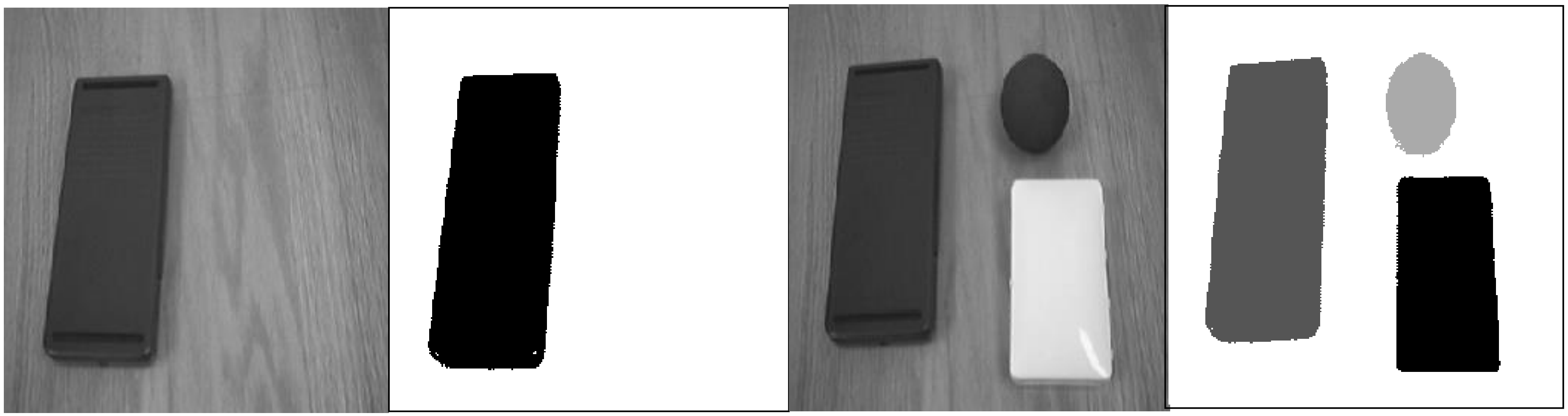
Disadvantages

-Limited application: application is restricted to tasks where internal detail is not required as a distinguishing characteristic.
-Does not extend to 3D: the 3D nature of objects can rarely be represented by silhouettes.
-Specialized lighting is required to obtain reliable binary images without restricting the environment.

Binary Image Processing

THRESHOLDING

In the simplest case, an image may consist of a single object or several separated objects of relatively high intensity, viewed against a background of relatively low intensity. This allows figure/ground separation by thresholding.

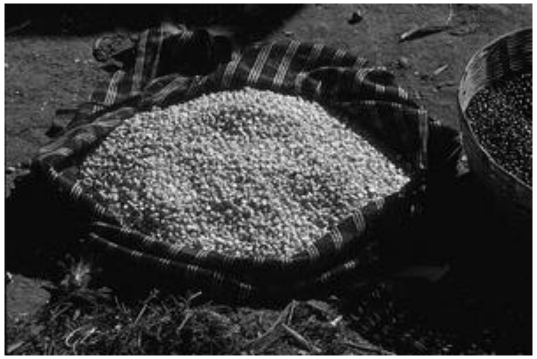
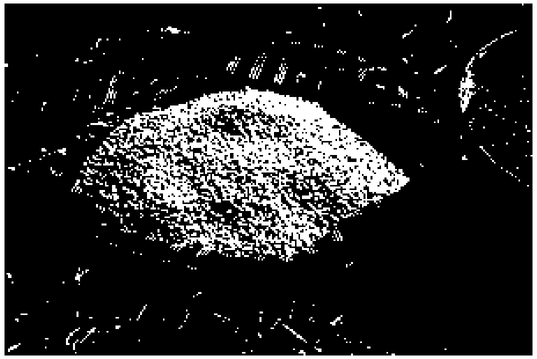


The goal of thresholding is to segment an image into regions of interest and to remove all other regions considered inessential. The simplest thresholding methods use a single threshold in order to isolate objects of interest. In many cases, however, no single threshold provides a good segmentation result over an entire image. In such cases variable and multilevel threshold techniques based on various statistical measures are used.

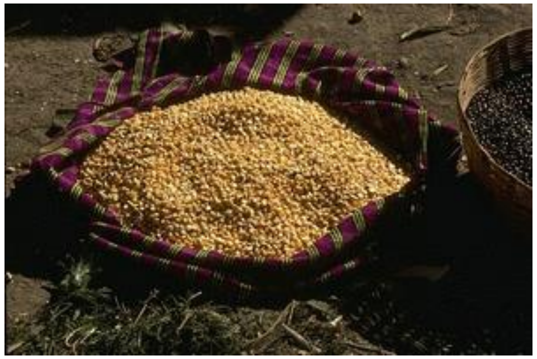
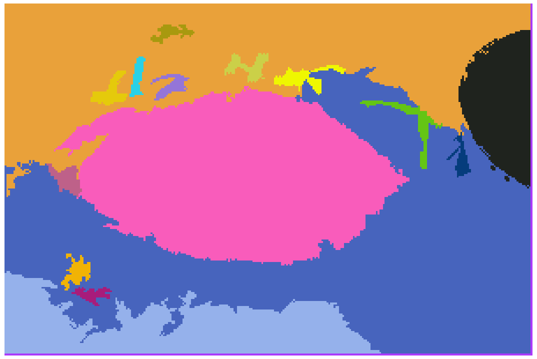
Binary Image Processing

Thresholding –individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value and as “background” pixels otherwise (assuming an object to be brighter than the background)

Segmentation -partitioning of an image into regions

Thresholding

Segmentation

Binary Image Processing

Segmentation -partitioning of an image into regions

$$f(i,j) \rightarrow P_1, P_2, \ldots, P_k$$

Def. 1.1 A region is a subset of an image

Def. 1.2 Segmentation is grouping pixels into regions, such that:

$\sum_{i=1}^{k} P_i$ = entire image (exhaustive partitioning)

$P_i \cap P_j$ = 0 if $i \neq j$ (exclusive partitioning)

Each region P_i satisfies a predicate; that is all points of the partition have same common property.

Pixels belonging to adjacent regions, when taken jointly, do not satisfy the predicate.

Strategies of threshold selection

- Global Thresholding
- Semithresholding
- Multilevel thresholding
- Variable thresholding
  - Threshold selection using mean and standard deviation
  - Threshold selection by Maximizing Between-Class Variance (Otsu method)
  - Threshold selection by searching the best approximation of the gray level image with a two leveled image

Global Thresholding

Let f(x,y) be the source image and b(x,y) the binary image.

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

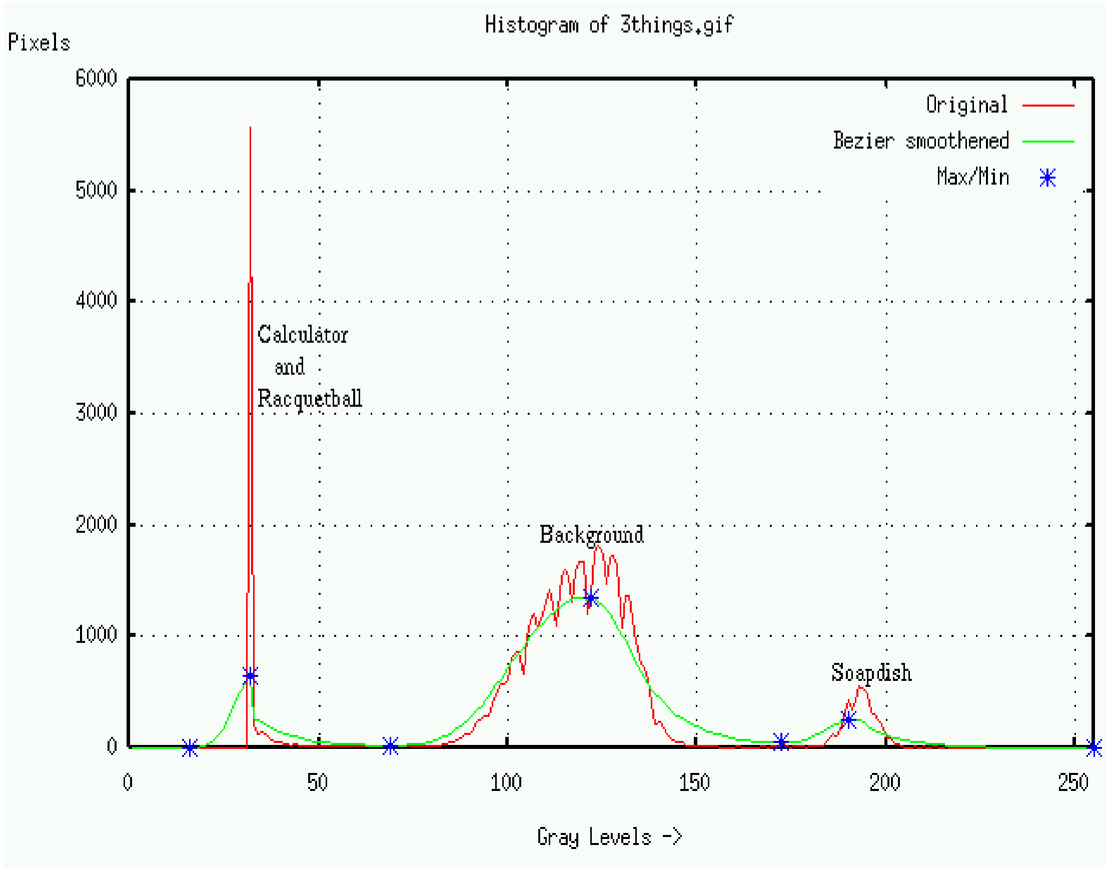
$Object \in [T_1, T_2]$

$$b(x,y) = \begin{cases} 1, & T_1 \le f(x,y) \le T_2 \\ 0, & otherwise \end{cases}$$

$Object \in Z$

where Z is a set of intensity values

$$b(x,y) = \begin{cases} 1, & f(x,y) \in Z \\ 0, & otherwise \end{cases}$$



Histogram analysis

Semithresholding

- Pixels whose values lie within a given threshold range retain their original values. Pixels with values lying outside of the threshold range are set to 0.

$$b(x,y) = \begin{cases} f(x,y) \; if \;\; h \leq f(x,y) \leq k \\ 0 \quad otherwise \end{cases}$$

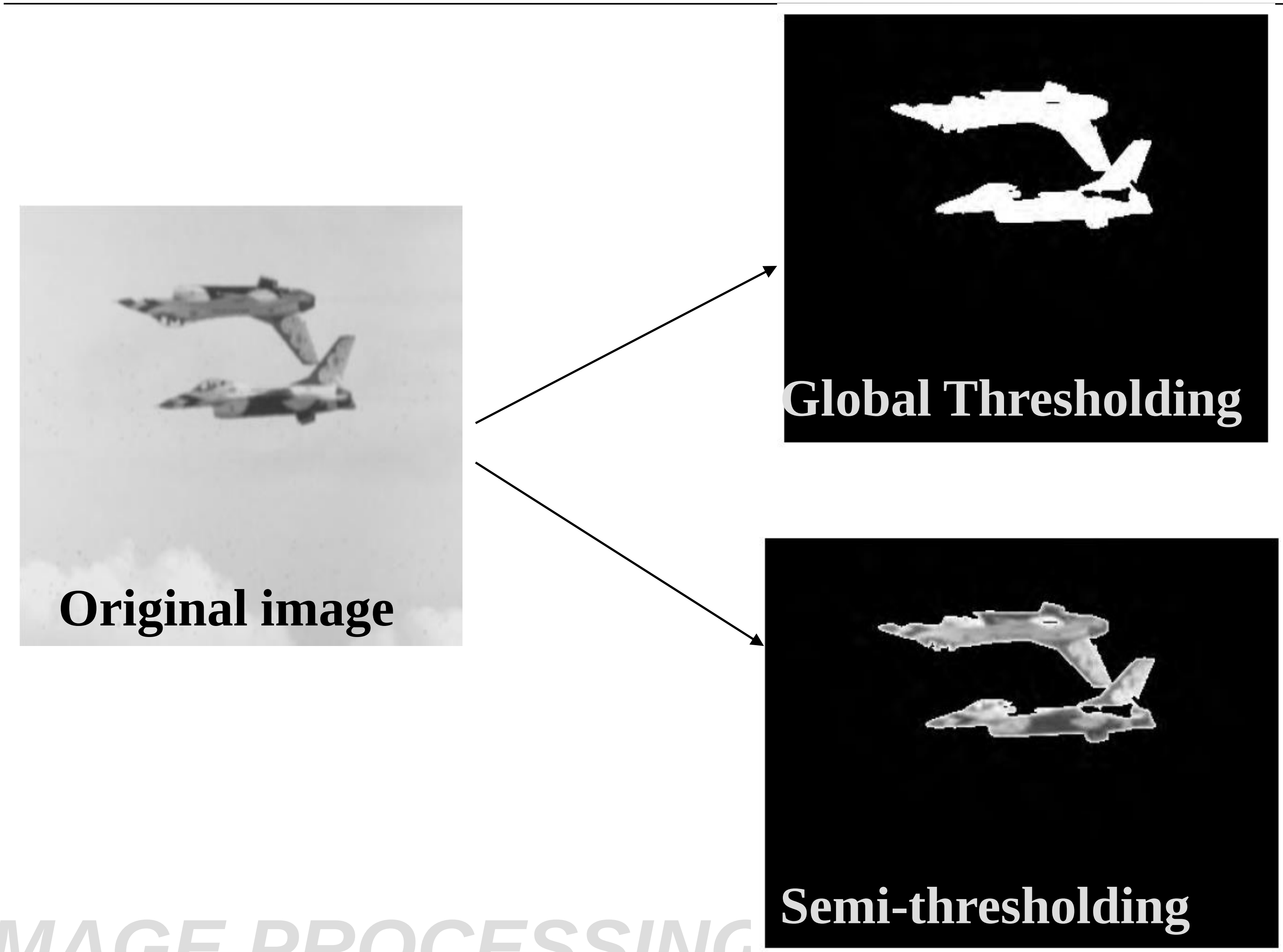
- Regions of high values can be isolated using:

$$b(x,y) = \begin{cases} f(x,y) \; if \;\; f(x,y) \geq k \\ 0 \quad otherwise \end{cases}$$

- Regions of low values can be isolated using:

$$b(x,y) = \begin{cases} f(x,y) \; if \;\; f(x,y) \leq k \\ 0 \quad otherwise \end{cases}$$

Example



Multilevel thresholding

- Multilevel thresholding allows for segmentation of pixels into multiple classes.
- For example, if the image histogram contains three peaks, then it is possible to segment the image using two thresholds. These thresholds divide the value set into three non-overlapping ranges, each of which can be associated with a unique value in the resulting image.
- Let f(x,y) be the source image and let $k_1, \ldots, k_n$ be threshold values satisfying $k_1 > k_2 > \ldots > k_n$. These values partition R into n+1 intervals which are associated with values $v_1, \ldots, v_{n+1}$ in the thresholded result image.
- The threshold image b is defined by:

$$b(x,y) = \begin{cases} v_1 \text{ if } f(x,y) < k_1 \\ v_i \text{ if } k_{i-1} \le f(x,y) < k_i \\ v_{n+1} \text{ if } k_n \le f(x,y) \end{cases}$$

Variable Thresholding

- Variable thresholding allows different threshold levels to be applied to different regions of an image.
- Let f(x,y) be the source image and let d(x,y) denote the local (region) threshold value associated with each point in the image, that is d(x,y) is the threshold value associated with the region in which point (x,y) lies.
- The thresholded image b(x,y) is defined by:

$$b(x,y) = \begin{cases} 1\, if\ \ f(x,y) \geq d(x,y) \\ 0\, if\ \ f(x,y) < d(x,y) \end{cases}$$

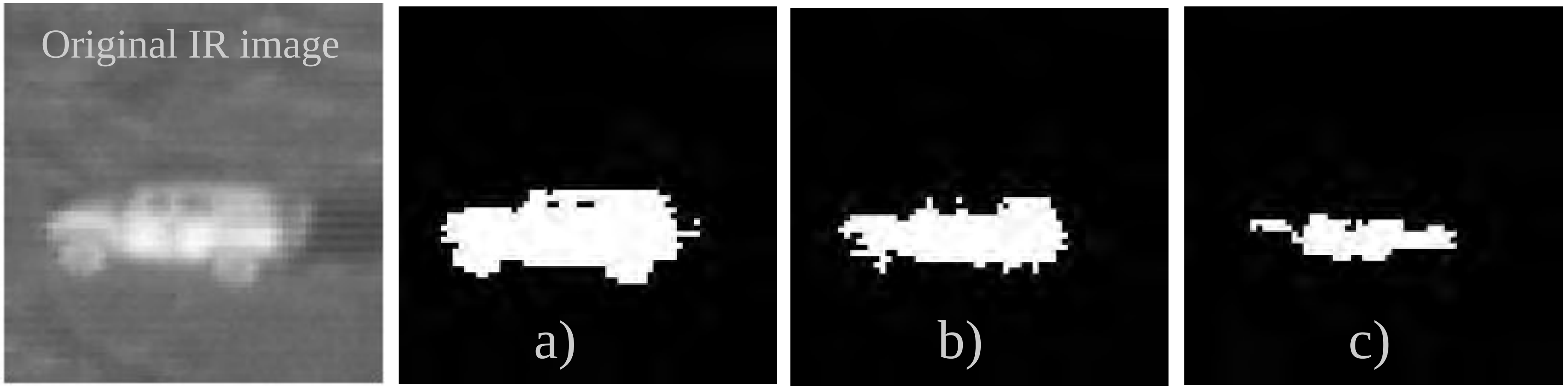
Threshold selection using mean and standard deviation

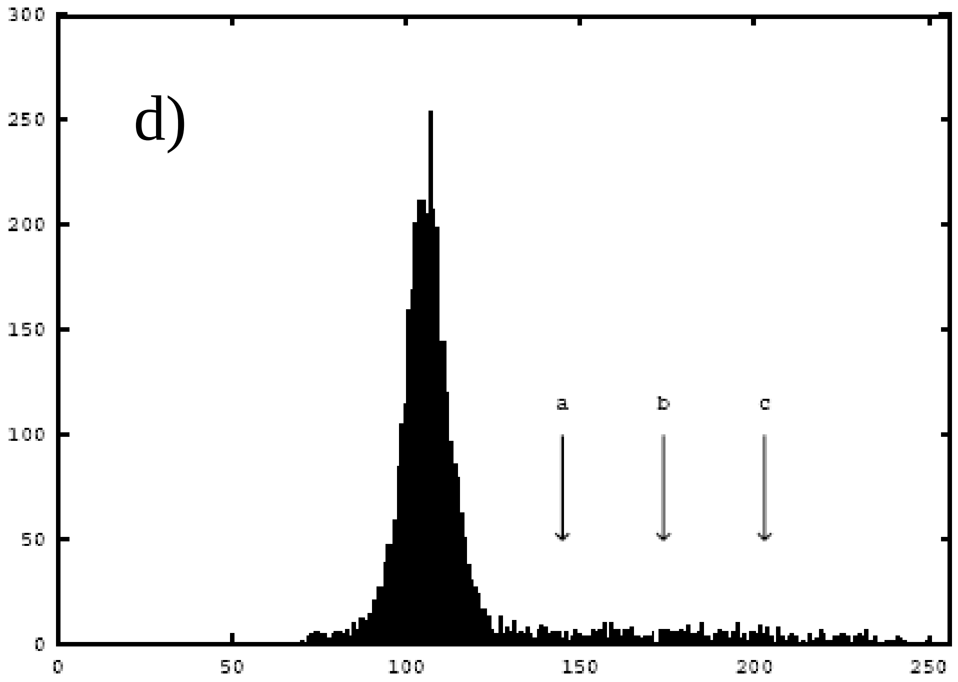
- Let f(x,y) , be an image
- The mean of f(x,y) is: $\mu = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} f(i,j)$
- The standard deviation of the image a is:

$$\sigma = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[f(i,j)-\mu]^2}$$

- The threshold level T is set at: $\mathbf{T = k_1\mu + k_2\sigma}$ where the constants k_1 and k_2 are image type dependent.
- For typical low-resolution IR images $k_1 = k_2 = 1$ seems to work fairly well.
- For higher resolutions $k_1 = 1$ or $k_1 = 1.5$ and $k_2=2$ may yield better results.

Example





a) $k_1=k_2=1$, T=145

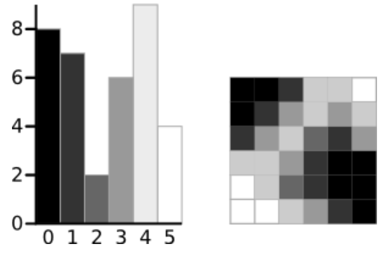
b) $k_1=1$, $k_2=2$, T=174

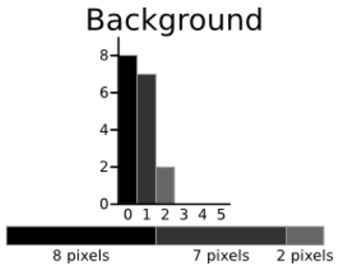
c) $k_1=1.5$, $k_2=1$, T=203

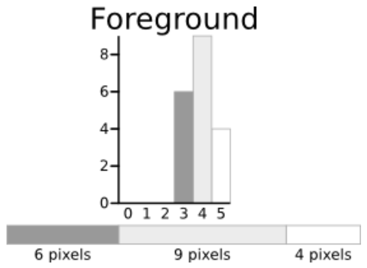
d) histogram of the original image and the threshold levels corresponding to images a), b) and c)

The Otsu Method – one threshold

- Problem statement: we have two groups of pixels, one with one range of values and one with another. Thresholding is difficult because these ranges usually overlap.
- Idea: minimize the error of classifying a background pixel as a foreground one or vice versa.

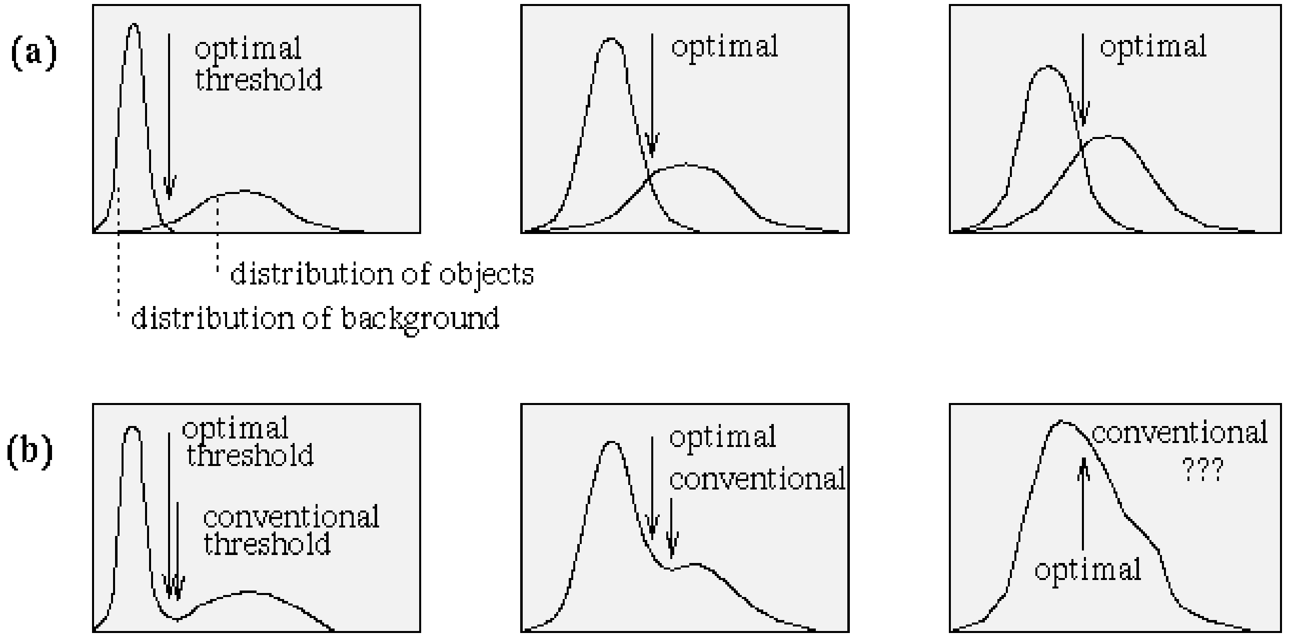






- Minimize the area under the histogram for one region that lies on the other region's side of the threshold. Consider the values in the two regions as two *clusters*.
- Set the threshold so as to try to make each cluster as tight as possible, thus (hopefully!) minimizing their overlap.
- **A measure of group homogeneity is variance.** A group with high homogeneity will have low variance. A group with low homogeneity will have high variance.

Adaptive thresholding



Gray level histograms approximated by two normal distributions; the threshold is set to give minimum probability of segmentation error.

(a) Probability distributions of background and objects

(b) Corresponding histograms and optimal threshold

Otsu’s Thresholding Method (1979)

- Find the threshold that ***minimizes the weighted within-class variance*** which turns out to be the same as ***maximizing the between-class variance***.
- Operates directly on the gray level histogram [*e.g.* 256 numbers, P(i)], so it’s fast (once the histogram is computed).
- Histogram (and the image) are *bimodal.*
- Assumes uniform illumination (implicitly), so the bimodal brightness behavior arises from object appearance differences only.

Otsu's Thresholding Method (2)

- The ***weighted within-class variance*** is: $\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$
- The ***class probabilities are estimated*** as: $q_1(t) = \sum_{i=1}^{t} P(i) \quad q_2(t) = \sum_{i=t+1}^{I} P(i)$
  - Where $P(i) = \frac{H(i)}{I}$, H(i) the i-th entry in the histogram

    $q_2 = 1 - q_1$
- And ***the class means are given*** by: $\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$
- Finally, ***the individual class variances*** are:

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \qquad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

- Now, we could actually stop here. All we need to do is just run through the full range of t values [1, 256] and pick the value that **minimizes the *within class variance***.
- But the relationship between the within-class and between-class variances can be exploited to generate a recursion relation that permits a much faster calculation.

Otsu's Thresholding Method (3)

- The relationship between the total variance and the within group variance can make the calculation of the best threshold less computationally complex.

$$\sigma^2 = \sum_{i=1}^{I}\left[(i-\mu)^2 P(i)\right] \qquad \mu = \sum_{i=1}^{I} iP(i)$$

$$\sigma^2 = \sum_{i=1}^{t}[i-\mu_1(t)+\mu_1(t)-\mu]^2 P(i) + \sum_{i=t+1}^{I}[i-\mu_2(t)+\mu_2(t)-\mu]^2 P(i)$$

$$= \sum_{i=1}^{t}\{[i-\mu_1(t)]^2 + 2[i-\mu_1(t)][\mu_1(t)-\mu]+[\mu_1(t)-\mu]^2\}P(i)$$

$$+ \sum_{i=t+1}^{I}\{[i-\mu_2(t)]^2 + 2[i-\mu_2(t)][\mu_2(t)-\mu]+[\mu_2(t)-\mu]^2\}P(i)$$

But: $\sum_{i=1}^{t}[i-\mu_1(t)][\mu_1(t)-\mu]P(i)=0 \qquad \sum_{i=t+1}^{I}[i-\mu_2(t)][\mu_2(t)-\mu]P(i)=0$

$$\Rightarrow \sigma^2 = \sum_{i=1}^{t}[i-\mu_1(t)]^2 P(i)+[\mu_1(t)-\mu]^2 q_1(t) + \sum_{i=t+1}^{I}[i-\mu_2(t)]^2 P(i)+[\mu_2(t)-\mu]^2 q_2(t)$$

$$\sigma^2 = [q_1(t)\sigma_1^2(t)+q_2(t)\sigma_2^2(t)]+\{q_1(t)[\mu_1(t)-\mu]^2 + q_2(t)[\mu_2(t)-\mu]^2\}$$

$$\mu = q_1(t)\mu_1(t)+q_2(t)\mu_2(t) \qquad 1-q_1(t)=q_2(t)$$

$$\Rightarrow \sigma^2 = \sigma_w^2(t)+q_1(t)[1-q_1(t)][\mu_1(t)-\mu_2(t)]^2$$

Otsu's Thresholding Method (4)

- ***Between/Within/Total Variance***: For any given threshold, the *total variance* is the sum of the ***within-class variances*** (weighted) and the ***between class variance***, which is the sum of weighted squared distances between the class means.

- As shown in the previous slide, we can express the total variance:

$$\sigma^2 = \underbrace{\sigma_w^2(t)}_{\textbf{Within-class}} + \underbrace{q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2}_{\textbf{Between-class},}$$

- Since the total is constant and independent of *t*, the effect of changing the threshold is simply to move the contributions of the two terms back and forth.

- So, ***minimizing the within-class variance is the same as maximizing the between-class variance.***

Otsu's Thresholding Method (5)

- For each potential threshold, T:
  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.
- The optimal threshold is the one that maximizes the between-class variance (or, conversely, minimizes the within-class variance).
- The nice thing about this is that we can compute the quantities *using a recursion relation* as we run through the range of *t* values.

Initialization... $q_1(1) = P(1) \quad ; \quad \mu_1(0) = 0$

Recursion...

$$q_1(t+1) = q_1(t) + P(t+1)$$

$$\mu_1(t+1) = \frac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)}$$

$$\mu_2(t+1) = \frac{\mu - q_1(t+1)\mu_1(t+1)}{1 - q_1(t+1)}$$

Example

Original image

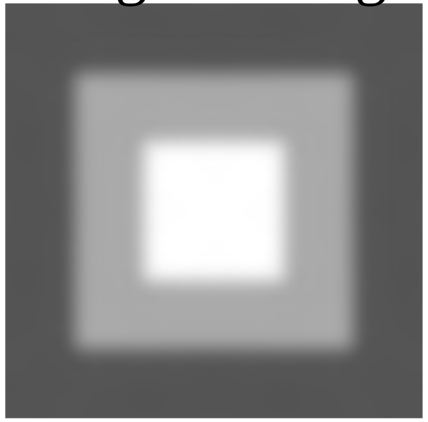
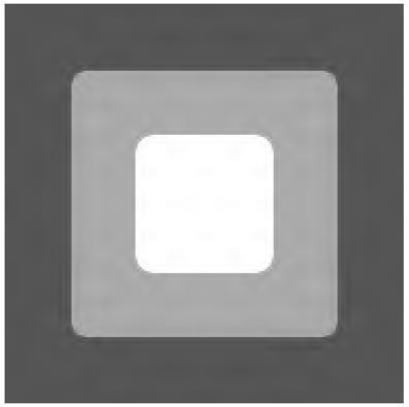
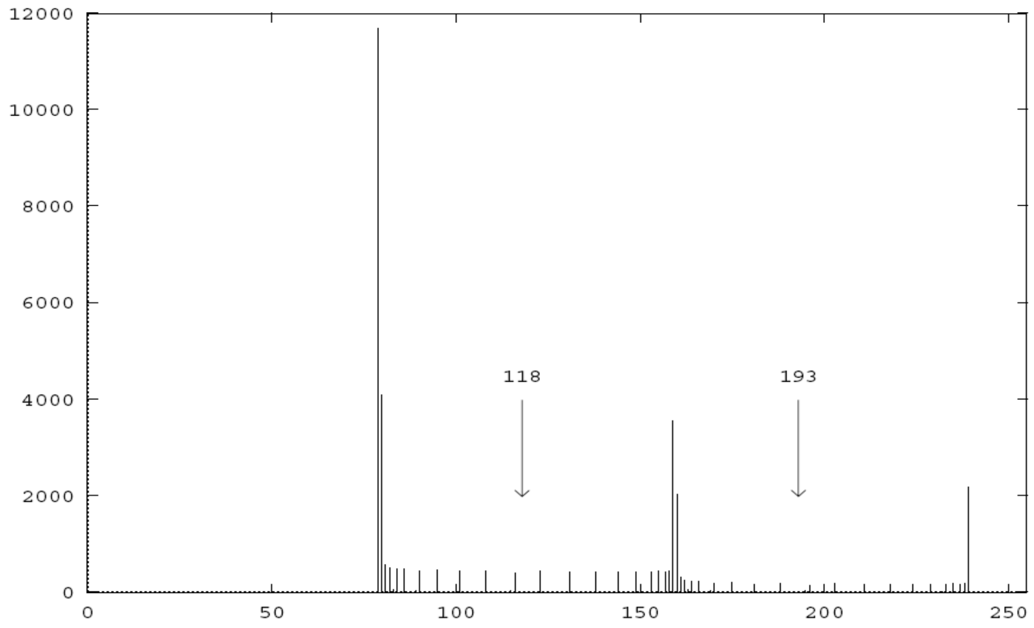


Image after thresholding

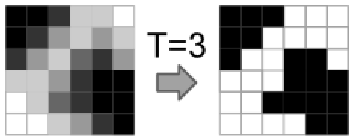




Histogram of the original image with threshold values marked.

Threshold selection by best approximation with a two level image

- Given an image f_{ij} the binary image is $g_{ij} = \begin{cases} a \text{ if } f_{ij} < t \\ b \text{ if } f_{ij} \geq t \end{cases}$
  - –Where t is the threshold
  - –a and b are constants chosen to minimize the distance on the corresponding intervals.
  - –The image has P gray levels
- The Euclidian distance on the interval [0, P-1] is



$$D = \sum_{i=1}^{D}\sum_{j=1}^{D}(f_{ij} - g_{ij})^2 = \sum_{i=1}^{D}\sum_{j=1}^{D}[(f_{ij})^2 - 2f_{ij}g_{ij} + (g_{ij})^2] =$$

$$= \underbrace{\sum_{k=0}^{t-1}k^2H_k - 2a\sum_{k=0}^{t-1}kH_k + a^2\sum_{k=0}^{t-1}H_k}_{\boxed{F(a)}} + \underbrace{\sum_{k=t}^{P-1}k^2H_k - 2b\sum_{k=t}^{P-1}kH_k + b^2\sum_{k=t}^{P-1}H_k}_{\boxed{F(b)}}$$

- The minimum of F(a) and F(b) are the mean values:

$$a = Min(F(a)) = \mu_i = \frac{\sum_{k=0}^{t-1}kH_k}{\sum_{k=0}^{t-1}H_k} \qquad b = Min(F(b)) = \mu_s = \frac{\sum_{k=t}^{P-1}kH_k}{\sum_{k=t}^{P-1}H_k}$$

Threshold selection by best approximation with a two level image(2)

- The distance on the entire image is:

$$D=\sum_{i=1}^{D}\sum_{j=1}^{D}(f_{ij}-g_{ij})^2=\underbrace{\sum_{k=0}^{P-1}k^2H_k}_{\text{const}}-\underbrace{\left[\frac{\left(\sum_{k=0}^{t-1}kH_k\right)^2}{\sum_{k=0}^{t-1}H_k}+\frac{\left(\sum_{k=t}^{P-1}kH_k\right)^2}{\sum_{k=t}^{P-1}H_k}\right]}_{\text{F – must be maximum for an optimal threshold}}$$

- $F=\mu_i^2A_i+\mu_s^2A_s$ where
  - A_i and A_s are the areas bellow and above the threshold
- F has to be evaluated for all t and the t maximizing F is selected.
- The algorithm can be applied for several levels of segmentation $t_1, t_2, \ldots t_n$ by dividing the intervals. At the first step the threshold will be $t_{n/2}$, next the algorithm is applied on the intervals $[0,t_{n/2}]$ and $[t_{n/2},P-1]$ and so on.

Presentation outline

- Introduction
- Thresholding methods
- Otsu Method
- Threshold selection by best approximation with a two level image

- **Binary Images: geometric properties**
- Projections
- Run-Length Coding

Binary Image Processing

Simple Geometrical Properties

Area

$$A = \iint b(x,y)dxdy$$

$$A = \sum_{i=1}^{n}\sum_{j=1}^{m} b(i,j)$$

Position

The usual practice is to chose the center of area. The center of area is the center of mass of a figure of the same shape with constant mass per unit area.

The center of the mass is that point where all the mass of the object could be concentrated without changing the first moment of the object about any axis.

$$\bar{x}\iint b(x,y)dxdy = \iint xb(x,y)dxdy \qquad \bar{i}\sum_{i=1}^{n}\sum_{j=1}^{m} b(i,j) = \sum_{i=1}^{n}\sum_{j=1}^{m} ib(i,j)$$

The moment about the x-axis

$$\bar{y}\iint b(x,y)dxdy = \iint yb(x,y)dxdy \qquad \bar{j}\sum_{i=1}^{n}\sum_{j=1}^{m} b(i,j) = \sum_{i=1}^{n}\sum_{j=1}^{m} jb(i,j)$$

The moment about the y-axis

Binary Image Processing

Position

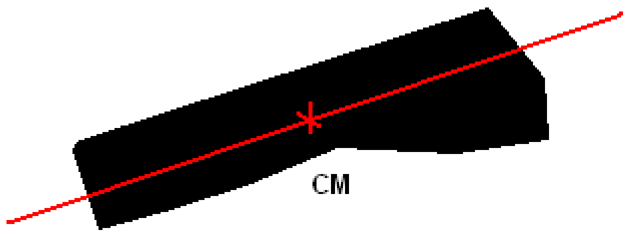
$(\bar{x}, \bar{y})$ And $(\bar{i}, \bar{j})$ represent the position of the center of area

$$\bar{i} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} i b(i,j)}{A} \qquad \bar{j} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} j b(i,j)}{A}$$

Orientation

Let assume that the object is somewhat elongated; then the orientation of the axis of elongation can be used to define the orientation of the object.

The axis of elongation is equivalent with the axis of least second order moment. It gives the line for witch the integral of the square of the distance to points in the object is a minimum.



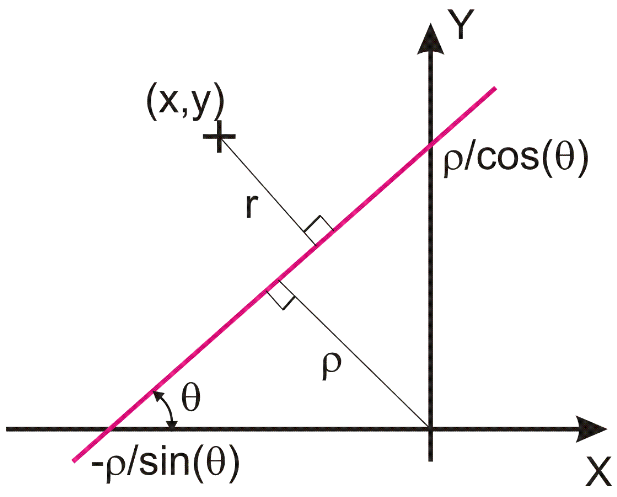
Binary Image Processing

Orientation

Least inertia axis: $E = \iint r^2 b(x,y)\,dxdy$

Where r is the perpendicular distance from the point (x,y) to the line.

To choose a particular line in the plane, we need to specify two parameters. A convenient pair consists of the distance ρ from the origin to the closest point on the line and the angle θ between the x-axis and the line, measured counter clockwise.



The line equation is:

$x \sin\theta - y \cos\theta + \rho = 0$

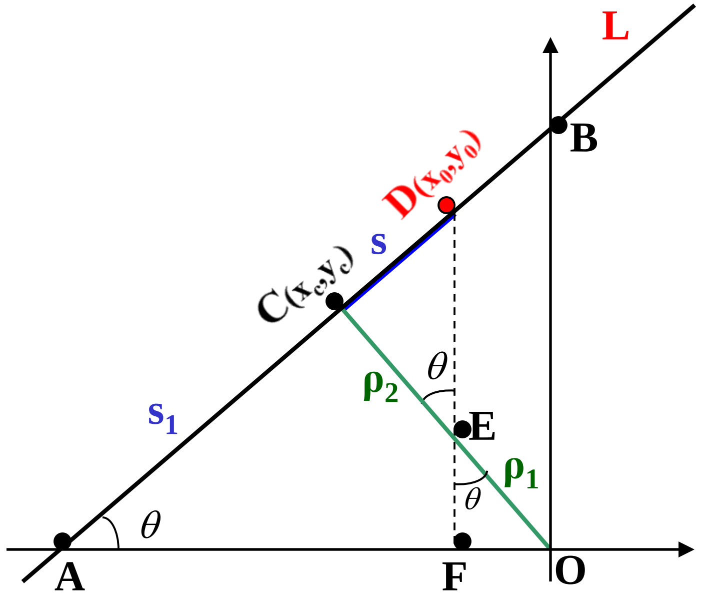
The line intersects the x-axis at $-\rho/\sin\theta$ *and* the y-axis at $+\rho/\cos\theta$.

The closest point on the line to the origin is at $(-\rho\sin\theta, +\rho\cos\theta)$.

Given a point (x,y) on the object, we need to find the closest point (x_0, y_0) on the line, so that we can compute the distance r.

$\rightarrow r^2 = (x \sin\theta - y \cos\theta + \rho)^2$

Binary Image Processing



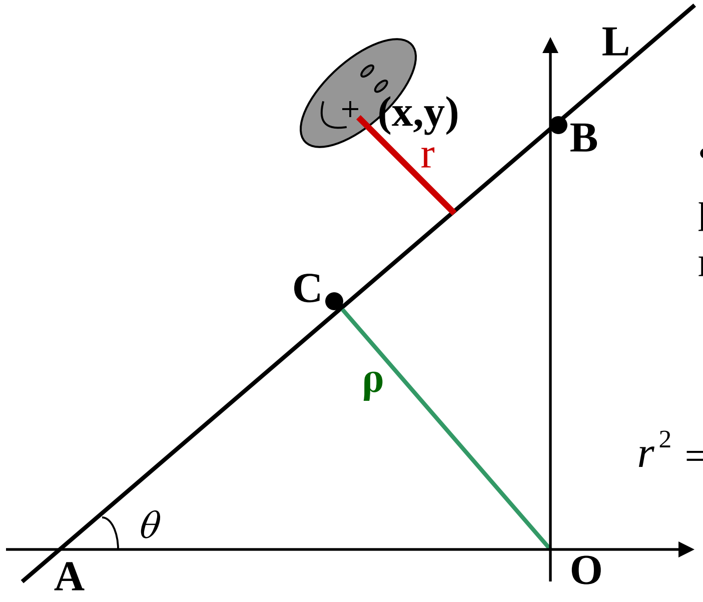
- $\rho = \rho_1 + \rho_2$
- $C(x_c,y_c)$ is the closest point on the line to the origin, where $x_c = -\rho\sin\theta$; $y_c = \rho\cos\theta$
- Try to write the parametric equation for the points on the line: $(x_0,y_0) \in L$
- Consider s the distance along the line from the point closest to the origin: CD = s

$$\left.\begin{aligned} \Delta\text{FEO: } & \sin\theta = \frac{FO}{EO} = \frac{FO}{\rho_1} \Rightarrow FO = \rho_1 \cdot \sin\theta = (\rho - \rho_2)\cdot\sin\theta \\ \Delta\text{ECD: } & \frac{\sin\theta}{\cos\theta} = \frac{CD}{CE} = \frac{s}{\rho_2} \Rightarrow \rho_2 = \frac{s\cos\theta}{\sin\theta} \end{aligned}\right\} x_0 = -\rho\sin\theta + s\cos\theta$$

$$\left.\begin{aligned} \Delta\text{AFD: } & \sin\theta = \frac{FD}{AD} = \frac{FD}{s+s_1} \Rightarrow FD = (s+s_1)\cdot\sin\theta \\ \Delta\text{AOC: } & \frac{\sin\theta}{\cos\theta} = \frac{CO}{AC} = \frac{\rho}{s_1} \Rightarrow s_1 = \frac{\rho\cos\theta}{\sin\theta} \end{aligned}\right\} y_0 = \rho\cos\theta + s\sin\theta$$

Parametric equation for the points on the line

Binary Image Processing



•Given a point (x,y) on the object, we need to find the closest point (x_0,y_0) on the line L so that we can compute the distance r between the point and the line.

$$r^2 = (x - x_0)^2 + (y - y_0)^2$$

•Substitute the parametric equations for x_0 and y_0:

$$r^2 = (x^2 + y^2) + \rho^2 + 2\rho(x\sin\theta - y\cos\theta) - 2s(x\cos\theta + y\sin\theta) + s^2$$

•Differentiate with respect to s and set the result equal to 0:

$$s = x\cos\theta + y\sin\theta$$

•This result can now be substituted back into the parametric equations for x_0 and y_0:

$$\left.\begin{aligned} x - x_0 &= +\sin\theta(x\sin\theta - y\cos\theta + \rho) \\ y - y_0 &= -\cos\theta(x\sin\theta - y\cos\theta + \rho) \end{aligned}\right\} \Longrightarrow r^2 = (x\sin\theta - y\cos\theta + \rho)^2$$

•Comparing this result with the equation for the line, we see that the line is the locus of points for which r=0 !

$$E = \iint_I r^2 b(x,y)dxdy = \iint_I (x\sin\theta - y\cos\theta + \rho)^2 b(x,y)dxdy$$

Binary Image Processing

Orientation

$$E = \iint_I r^2 b(x, y)dxdy = \iint_I (x\sin\theta - y\cos\theta + \rho)^2 b(x, y)dxdy$$

Differentiating with respect to ρ and setting the result to zero lead to

E'$_\rho$*(θ, ρ)=0*

$$E'_\rho = \iint 2(x\sin\theta - y\cos\theta + \rho)b(x,y)dxdy =$$

$$= 2\sin\theta \iint xb(x,y)dxdy - 2\cos\theta \iint yb(x,y)dxdy + \rho \iint b(x,y)dxdy$$

$$E'_\rho = 2A(\bar{x}\sin\theta - \bar{y}\cos\theta + \rho) = 0$$

$\bar{x}$ and $\bar{y}$ is the center of area defined above.

Thus the axis of least second moment passes through the center of area. This suggests a change of coordinates to $x' = x - \bar{x}$ and $y' = y - \bar{y}$

x sinθ – y cosθ + ρ = x' sinθ – y' cosθ

$$E = a\sin^2\theta - b\sin\theta\cos\theta + c\cos^2\theta$$

where a, b, c, are the second order moments given by:

$$a = \iint (x')^2 b(x, y)dx'dy'$$

$$b = 2\iint (x'y')b(x, y)dx'dy'$$

$$c = \iint (y')^2 b(x, y)dx'dy'$$

Binary Image Processing

Orientation

Now we can write the formula of E in the form

$$E = \frac{1}{2}(a+c) - \frac{1}{2}(a-c)\cos 2\theta - \frac{1}{2} b \sin 2\theta$$

Differentiating with respect to θ and setting the result to zero, we have

$$E_\theta^{'} = (a-c)\sin 2\theta - b\cos 2\theta = 0$$

$$\tan 2\theta = \frac{b}{a-c}$$ unless b=0 and a=c

Consequently

$$\sin 2\theta = \pm \frac{b}{\sqrt{b^2 + (a-c)^2}}$$ and $$\cos 2\theta = \frac{a-c}{\sqrt{b^2 + (a-c)^2}}$$

Of the two solutions, the one with the plus signs in the expressions for sin2θ and cos2θ leads to the desired **minimum for E (E_{min})**.

Consequently, the solution with minus signs in the two expressions corresponds to the **maximum value for E (E_{max})**.

Binary Image Processing

$$A_i = \sum_{r=0}^{N-1}\sum_{c=0}^{N-1} I_i(r,c)$$

$$\bar{r}_i = \frac{1}{A_i}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1} r I_i(r,c) \qquad \bar{c}_i = \frac{1}{A_i}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1} c I_i(r,c)$$

$$\tan(2\theta_i) = 2\frac{\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}(r-\bar{r}_i)(c-\bar{c}_i)I_i(r,c)}{\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}(c-\bar{c}_i)^2 I_{i(r,c)} - \sum_{r=0}^{N-1}\sum_{c=0}^{N-1}(r-\bar{r}_i)^2 I_i(r,c)}$$

$$T = 4\pi\left(\frac{A}{P^2}\right)$$ **thinness ratio**

$$F = \frac{E_{min}}{E_{max}}$$ **roundness (F=0 straight line, F=1 circle)**

Presentation outline

- Introduction
- Thresholding methods
- Otsu Method
- Threshold selection by best approximation with a two level image

- Binary Images: geometric properties
- **Projections**
- Run-Length Coding

Projections

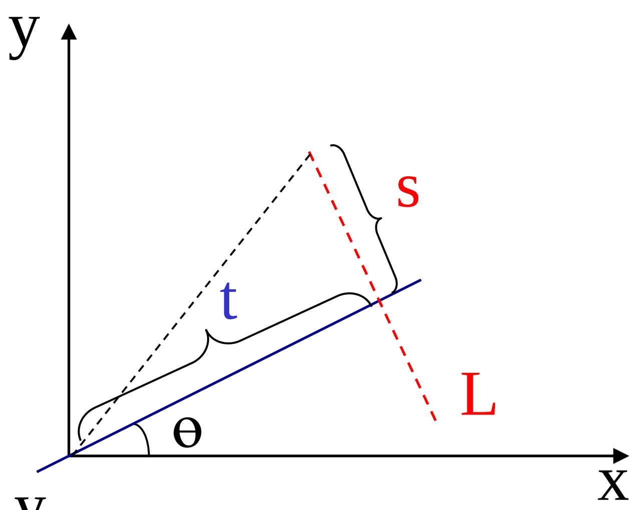
Projections

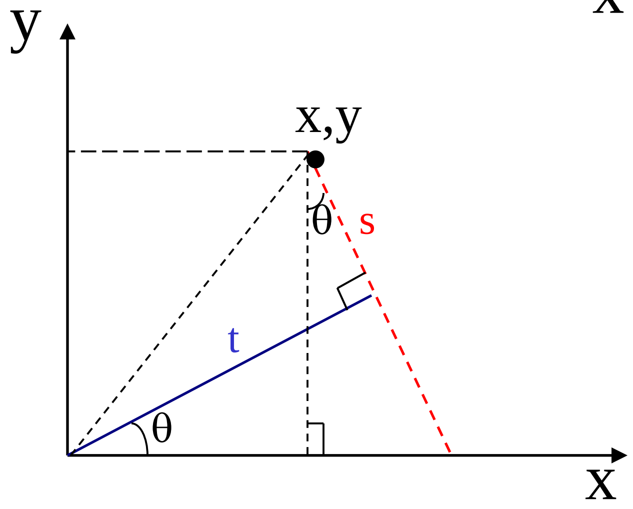
The integral of b(x,y) along the line L gives one value of the projection.

$$p_\theta(t) = \int b(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)ds$$

Vertical projection (θ= 0) $$v(x) = \int b(x,y)dy$$

Horizontal projection (θ= π/2) $$h(y) = \int b(x,y)dx$$





Projections(2)

Projections

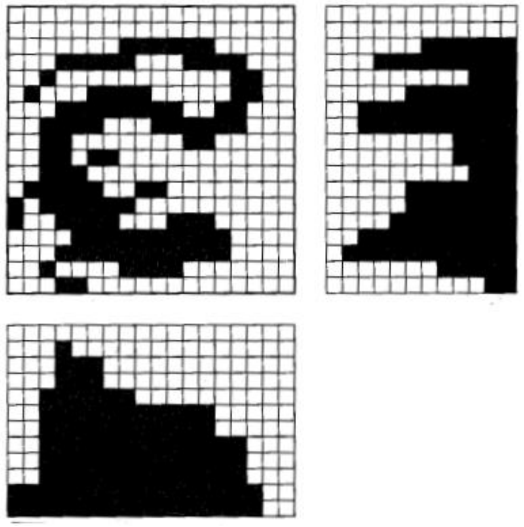
Area $$A = \iint b(x,y)\,dxdy \qquad A = \int v(x)\,dx = \int h(y)\,dy$$

Coordinates of the center of the area

$$\bar{x}A = \iint x b(x,y)\,dxdy = \int x v(x)\,dx$$
$$\bar{y}A = \iint y b(x,y)\,dxdy = \int y h(y)\,dy$$

$$\Updownarrow$$

The first moments of the projections are equal to the first moments of the original image.



Projections(3)

Projections

Orientation

To compute orientation we need the second moments, too. Two of these are easy to compute from the projections:

$$\iint x^2 b(x,y)dxdy = \int x^2 v(x)dx \qquad \iint y^2 b(x,y)dxdy = \int y^2 h(y)dy$$

We cannot compute the integral of the product *xy* from the two projections introduced so far.

We can add the diagonal projection **(θ= π/4)** $d(t) = \int b\left(\frac{1}{\sqrt{2}}(t-s), \frac{1}{\sqrt{2}}(t+s)\right)ds$

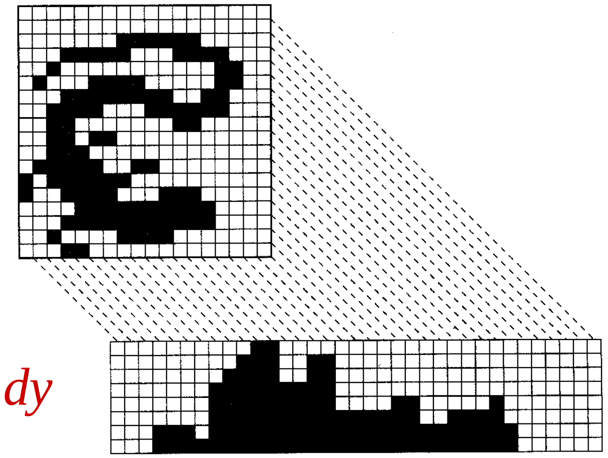
Now consider that:

$$\iint_I \frac{1}{2}(x+y)^2 b(x,y)dxdy = \iint t^2 b(x,y)dsdt = \int t^2 d(t)dt$$

$$\iint_I \frac{1}{2}(x+y)^2 b(x,y)dxdy = \iint_I \left(\frac{1}{2}x^2 + xy + \frac{1}{2}y^2\right)b(x,y)dxdy$$

So:

$$\iint xyb(x,y)dxdy = \int t^2 d(t)dt - \frac{1}{2}\int x^2 v(x)dx - \frac{1}{2}\int y^2 h(y)dy$$



Presentation outline

- Introduction
- Thresholding methods
- Otsu Method
- Threshold selection by best approximation with a two level image

- Binary Images: geometric properties
- Projections
- **Run-Length Coding**

Run-Length Coding

Run-Length Encoding

This methods exploits the fact that along any particular scan line there will usually be long runs of zeros or ones.

Two approaches are commonly used in run-length encoding.

-the start positions and length of runs of 1s for each row are used.

-lengths of runs, starting with the length of the 0 run.

1	1	1	0	0	0	1	1	0	0	0	1	1	1	1	0	1	1	0	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Start and length of 1 runs:	(1,3) (7,2) (12,4) (17,2) (20,3)
	(5,13) (19,4)
	(1,3) (17,6)
Length of 0 and 1 runs:	0,3,3,2,3,4,1,2,1,3
	4,13,1,4
	0,3,13,6

Run-Length coding (2)

Computation of the geometric properties from run-length

Use the convention that r_{ik} is the k-th run of the i-th line and that the first run in each row is a run of zeros (thus all the even runs will correspond to ones in the image). There are m_i runs on the i-th line

Area

$$A = \sum_{i=1}^{n} \sum_{k=1}^{m_i/2} r_{i,2k}$$

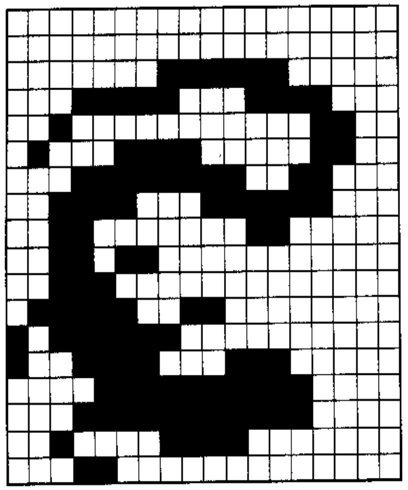
Position

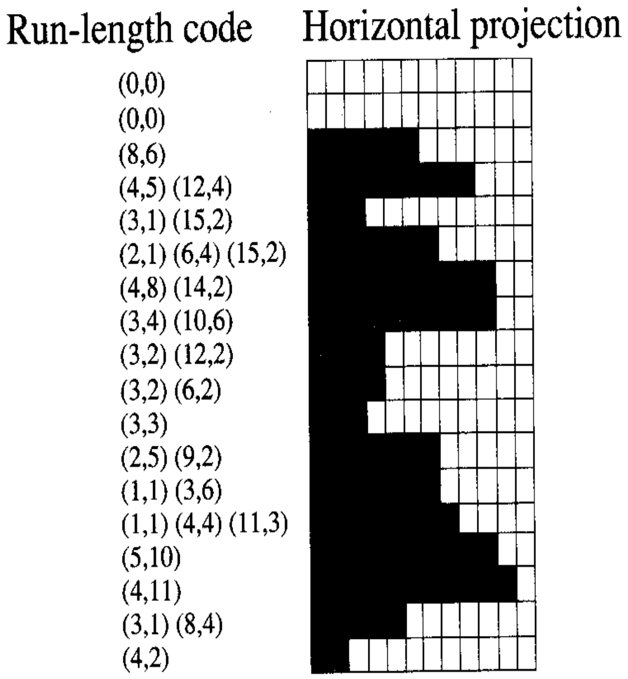
First of all, we obtain the horizontal projection as follows:

$$h_i = \sum_{k=1}^{m_i/2} r_{i,2k}$$

From this we can easily compute the vertical position of the center of area using:

$$\bar{i}A = \sum_{i=1}^{n} i h_i$$





Run-length code

(0,0)
(0,0)
(8,6)
(4,5) (12,4)
(3,1) (15,2)
(2,1) (6,4) (15,2)
(4,8) (14,2)
(3,4) (10,6)
(3,2) (12,2)
(3,2) (6,2)
(3,3)
(2,5) (9,2)
(1,1) (3,6)
(1,1) (4,4) (11,3)
(5,10)
(4,11)
(3,1) (8,4)
(4,2)

Run-Length coding (3)

The vertical projection is obtained by adding up all picture cell values in one column of the image.

It is difficult to compute vertical projection directly from the run lengths. Consider instead the **first difference of the vertical projection**:

$$\bar{v}_j = v_j - v_{j-1}\,,\quad \bar{v}_1 = v_1$$

The first difference of the vertical projection can be obtained by projecting not the image data, but the **first horizontal differences of the image data**:

$$\bar{b}_{i,j} = b_{i,j} - b_{i,j-1}$$

The first difference $\bar{b}_{i,j}$ has the advantage over $b_{i,j}$ itself that is nonzero only at the beginning of each run. It equals +1 where the data change from 0 to a 1, and –1 where the data change from a 1 to a 0.

Run-Length coding (4)

													0	
														$r_{i,k}$
							0	1	1	0			2	8,2,3
						0	1	1	1	0			3	7,3,3
					0	1	1	1	1	1	0		5	6,5,2
		0	1	0	0	1	1	1	1	1	0		6	3,1,2,5,2
	0	1	1	1	1	1	1	1	1	0			8	2,8,3
0	1	1	1	1	1	1	1	1	0				8	2,8,4
0	1	1	1	1	1	1	1	0					7	1,7,5
		0	1	1	1	0							3	3,3,7
													h_i	

Original image

Run-Length coding (5)

$\bar{b}_{i,j}$

⋮

													0
								+1	0	-1			2
							+1	0	0	-1			3
						+1	0	0	0	0	-1		5
			+1	-1		+1	0	0	0	0	-1		6
		+1	0	0	0	0	0	0	0	-1			8
	+1	0	0	0	0	0	0	0	-1				8
	+1	0	0	0	0	0	0	-1					7
			+1	0	0	-1							3
													h_i
0	+2	+1	+2	-1	0	+1	+1	0	-1	-3	-2	0	
0	2	3	5	4	4	5	6	6	5	2	0	0	v_j

Run-Length coding (6)

We can locate these places by computing the summed run-length code

$$\tilde{r}_{ik} = 1 + \sum_{p=1}^{k} r_{i,p}$$

recursively: $\tilde{r}_{i,0} = 1$ and $\tilde{r}_{i,j+1} = \tilde{r}_{i,j} + r_{i,j+1}$

Then, for a transition at $j = \tilde{r}_{ik}$ we add $(-1)^{k+1}$ to the accumulated total for the first difference of the vertical projection

Run-Length coding (7)

													0
							0	1	1	0			2
						0	1	1	1	0			3
					0	1	1	1	1	1	0		5
		0	1	0	0	1	1	1	1	1	0		6
	0	1	1	1	1	1	1	1	1	0			8
0	1	1	1	1	1	1	1	1	0				8
0	1	1	1	1	1	1	1	0					7
		0	1	1	1	0							3
													h_i
0	+2	+1	+2	-1	0	+1	+1	0	-1	-3	-2	0	
0	2	3	5	4	4	5	6	6	5	2	0	0	v_j

$r_{i,k}$	$\tilde{r}_{ik} = 1 + \sum_{p=1}^{k} r_{i,p}$
8,2,3	1,9,11,14
7,3,3	1,8,11,14
6,5,2	1,7,12,14
3,1,2,5,2	1,4,5,7,12,14
2,8,3	1,3,11,14
1,8,4	1,2,10,14
1,7,5	1,2,9,14
3,3,7	1,4,7,14

Run-Length coding (4)

- From this difference we can compute the vertical projection itself using the simple summation

$$v_j = \sum_{p=1}^{j} \bar{v}_p$$

- Or recursively: $v_0 = 0$ and $v_{j+1} = v_j + \bar{v}_{j+1}$
- Given the vertical projection, we can easily compute the horizontal projection of the center of area $A\bar{j} = \sum_{j=1}^{m} j v_j$

- The diagonal projection can be obtained in a way similar to that used to obtain the vertical projection.

References:

- Gerhard X. Ritter, Joseph N. Wilson *Handbook of Computer Vision Algorithms in Image Algebra*
- Linda Shapiro: *Computer Vision*
- Bryan S. Morse, Brigham Young University, Thresholding lecture
- Berthold Klaus, Paul Horn, Robot Vision, The MIT Press, 1986