Universidad Veracruzana

Image Segmentation

Dr. Héctor Gabriel Acosta Mesa

Instituto de Investigaciones en Inteligencia Artificial
Maestría en Inteligencia Artificial
heacosta@uv.mx
www.uv.mx/heacosta

Image Analysis

© R. C. Gonzalez & R. E. Woods, Digital Image Processing, 2nd ed. www.imageprocessingbook.com

Preview

•Transition from image processing methods whose input and output are images, to methods in which the inputs are images, but the outputs are attributes extracted from those images.

•Segmentation subdivides an image into its constituent regions or objects.

•Segmentation of nontrivial images is one of the most difficult tasks in image processing.

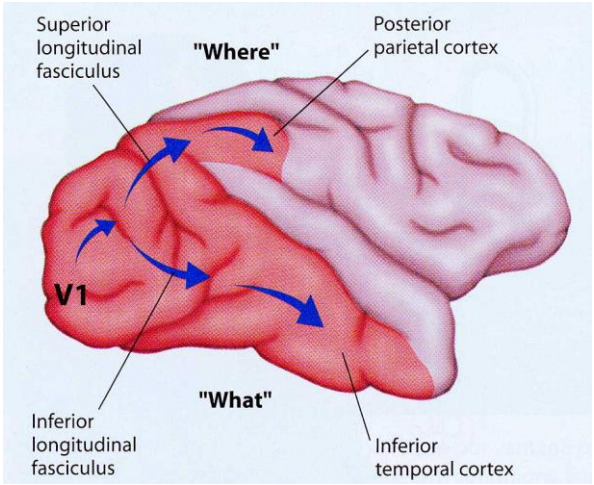
•The usual approach is to focus on selecting the types of sensors most likely to enhance the objects of interest while diminishing the contribution of irrelevant image detail.

Tareas de la Visión Computacional

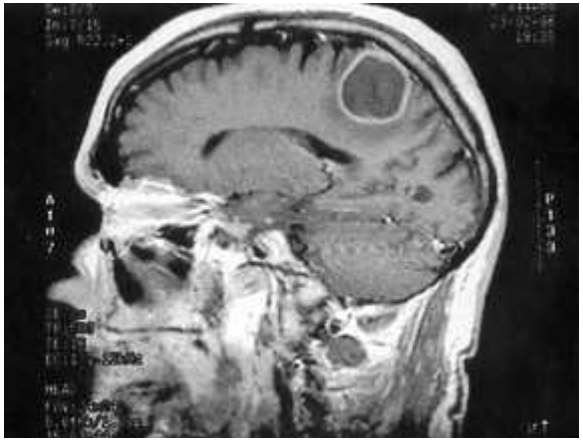
Clasification

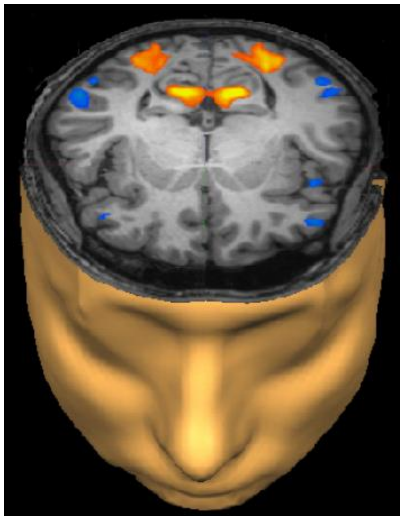
Object detection

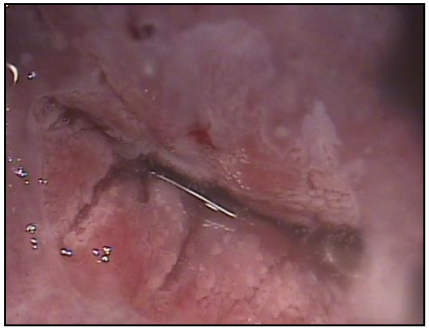
Semantic segmentation

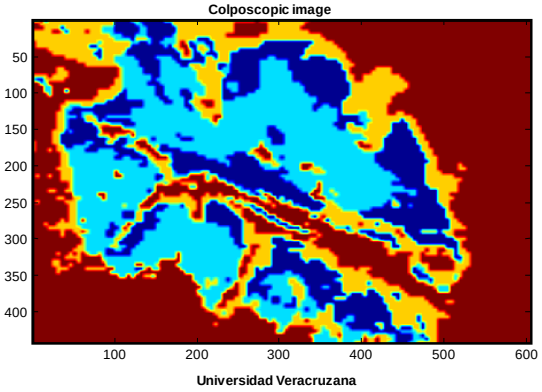


Preview







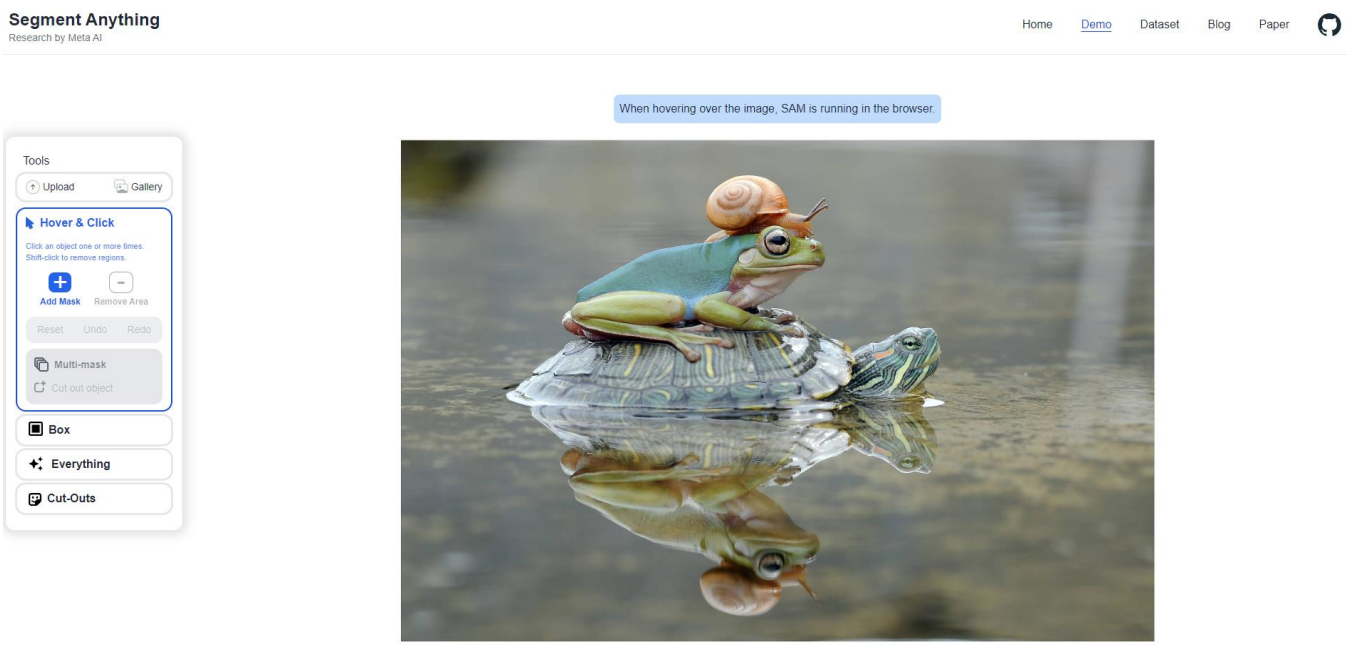

Colposcopic image
50
100
150
200
250
300
350
400
100
200
300
400
500
600
Universidad Veracruzana

The Segmented and Annotated IAPR TC-12 Benchmark

SAM

Segment Anithing:

https://segment-anything.com/



https://segment-anything.com/demo

CV Online Image Database

Index by Topic

1. Action Databases
2. Biological/Medical
3. Face Databases
4. Fingerprints
5. General Images
6. General RGBD and depth datasets
7. Gesture Databases
8. Image, Video and Shape Database Retrieval
9. Object Databases
10. People, Pedestrian, Eye/Iris, Template Detection/Tracking Databases
11. Segmentation
12. Surveillance
13. Textures
14. General Videos
15. Other Collection Pages
16. Miscellaneous Topics

http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm#face

Preview

•Basic properties of intensity values: discontinuity and similarity. In addition to edge detection per se, we also need methods for connecting edge segments.

•Three basic types of discontinuities: points, lines, and edges.

Segmentation (gray scale)

Edges

High pass filters

Regions

Global Thresholding
Adaptive Thresholding
Region-Based
Region Growing
Splitting and Merging

Contours

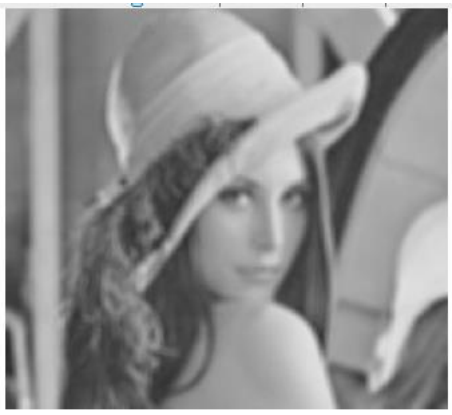
Edge linking
Hough transform
Active contours (snakes)
Active shape models

Edge Detection

Properties of the second derivate around an edge: (1) It produces two values for every edge in an image and (2) an imaginary straight line joining the extreme positive and negative values of the second derivate would cross zero near the midpoint of the edge

Realidad transformada

f D(f) T(f) D(T(f))



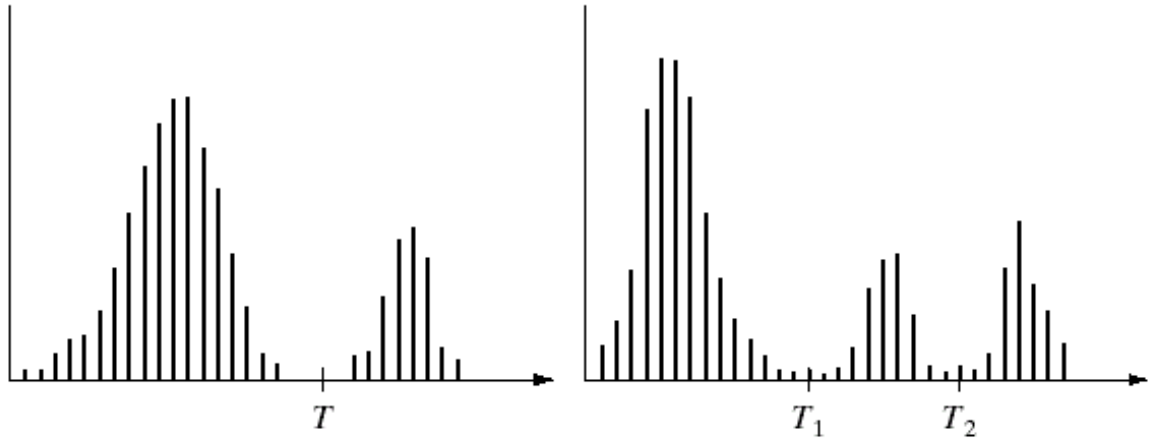
Mejor percepción

Application for Human Perceptions

Universidad Veracruzana

Global Thresholding

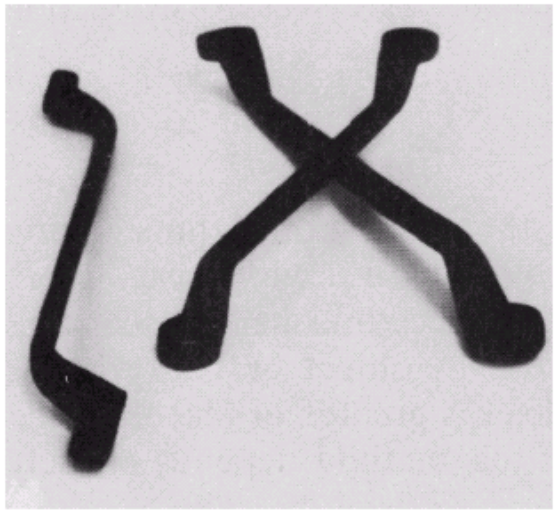
The simplest of all thresholding techniques is to partition the image histogram by using a single global threshold, T, as illustrated in figure 10.26.

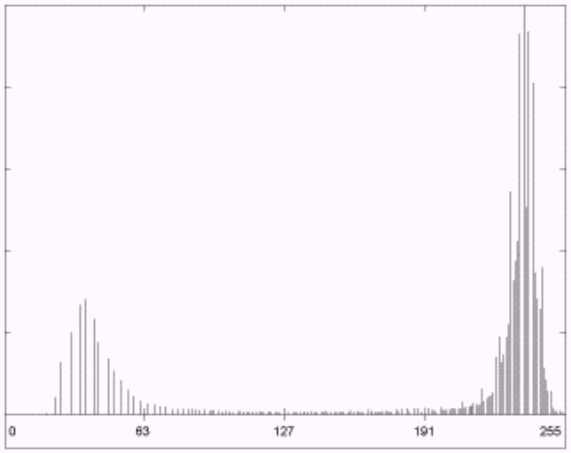


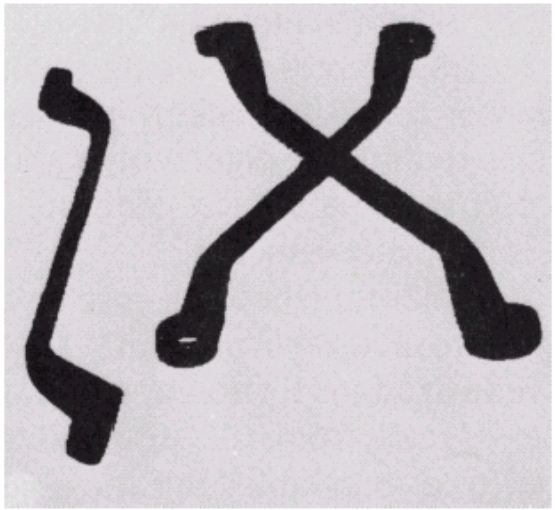
a b

FIGURE 10.26 (a) Gray-level histograms that can be partitioned by (a) a single threshold, and (b) multiple thresholds.

Global Thresholding







a
b c

FIGURE 10.28
(a) Original image. (b) Image histogram. (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

Isodata algorithm

The threshold in the preceding example was specified by using a heuristic approach

1. Select an initial estimate for T.
2. Segment the image using T. This will produce two groups of pixels: G_1 consisting of all pixels with gray level values $>T$ and G_2 consisting of pixels with values $\leq T$.
3. Compute the average gray level values μ_1 and μ_2 for the pixels in regions G_1 and G_2.
4. Compute a new threshold value:

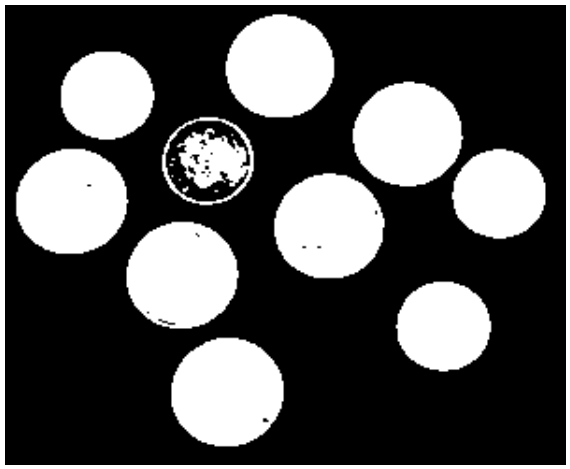
$$T = \frac{1}{2}(\mu_1 + \mu_2).$$

5. Repeat steps 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter T_o.

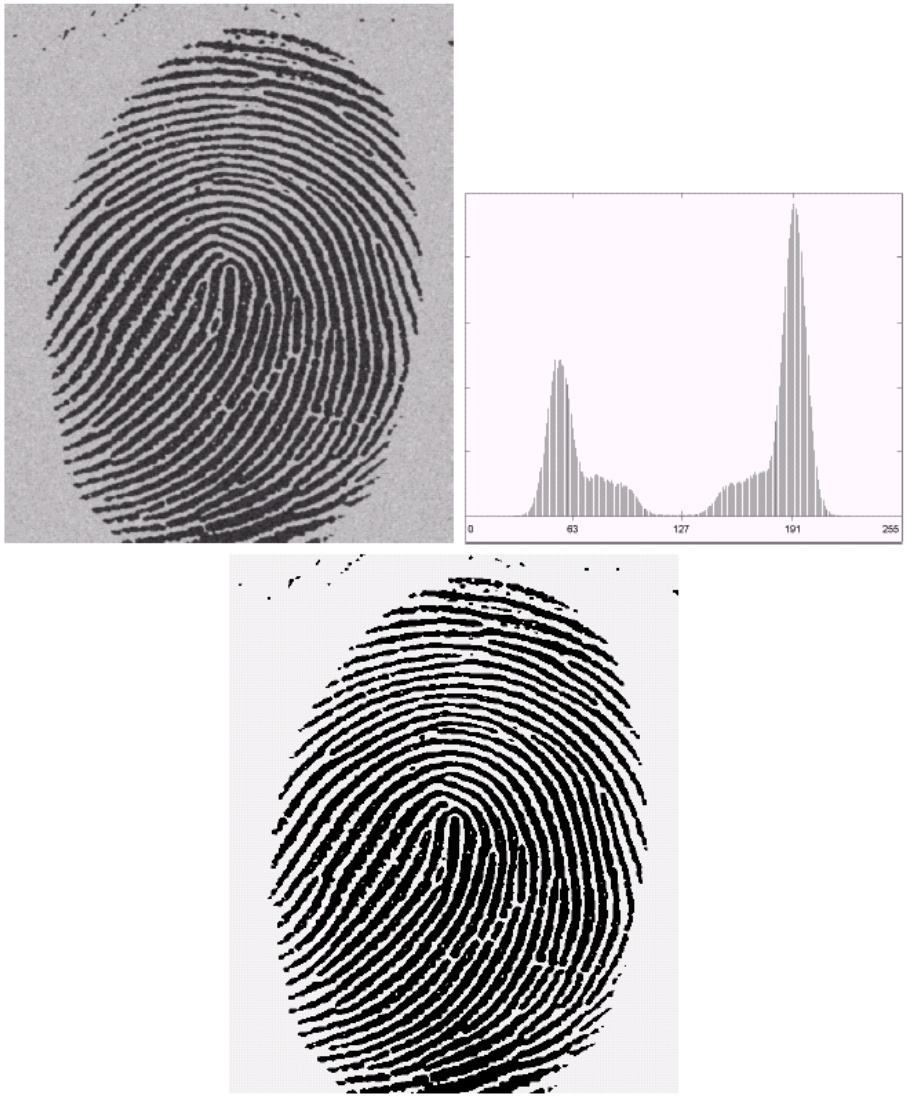
A good initial value for T is the average gray level of the image.

Isodata algorithm

```
I = imread('coins.png');
level = isodata(I);
BW = imbinarize(I,level);
imshow(BW)
```



Isodata algorithm



a b
c

FIGURE 10.29
(a) Original image. (b) Image histogram. (c) Result of segmentation with the threshold estimated by iteration. (Original courtesy of the National Institute of Standards and Technology.)

Otsu’s algorithm

- El método Otsu (propuesto por Nobuyuki Otsu) es uno de los muchos algoritmos de binarización, es decir, convertir una imagen en escala de grises (valor de píxeles que van de 0 a 255) en imagen binaria (el valor de píxeles puede tener solo 2 valores: 0 o 1) [1-2].

- El método asume que la imagen tiene dos clases de píxeles siguiendo el histograma bimodal (píxeles de primer plano y píxeles de fondo) , el umbral óptimo es un umbral global obtenido dinámicamente [1].

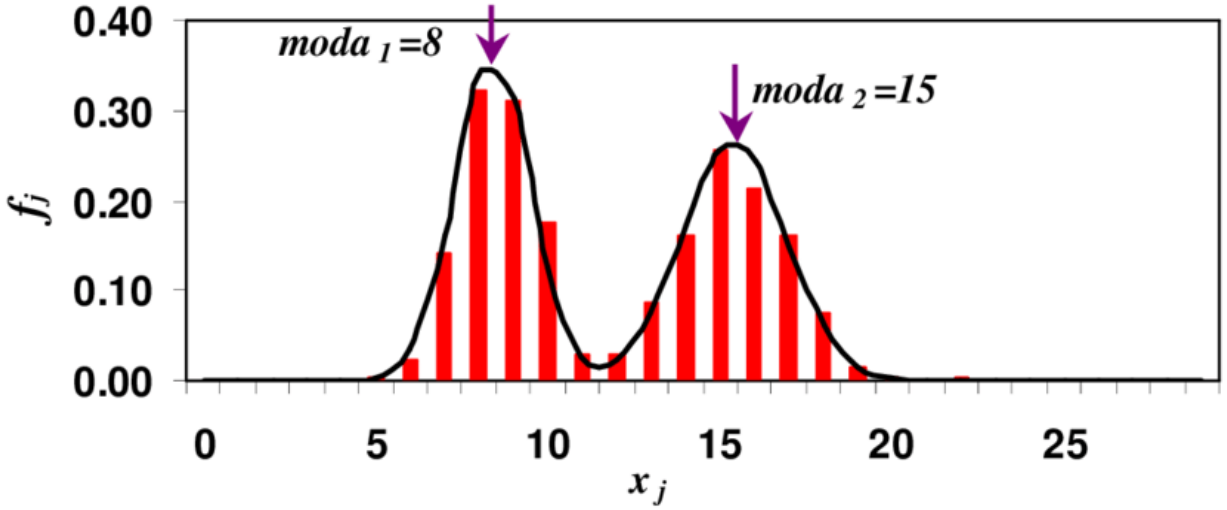


Figura 1. Ejemplo de distribución bimodal [4]

Otsu’s algorithm

- El algoritmo consiste en iterar a través de todos los valores de umbral posibles y calcular la varianza (medida de dispersión) para todos los niveles de píxeles de cada lado del umbral. Es decir los píxeles que se encuentran en el primer plano o en el segundo plano. El objetivo es encontrar el umbral óptimo que separe las clases logrando que su varianza intra-clase sea mínima [1].

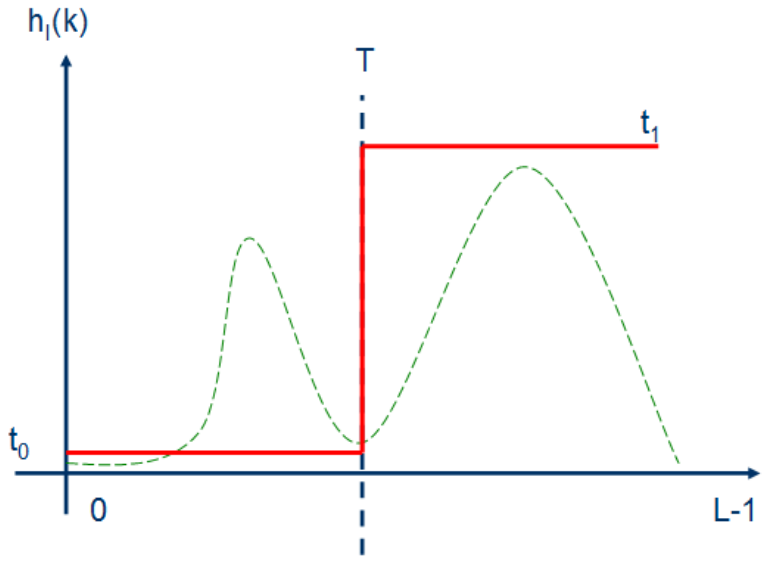


Figura 2. Binarización en histograma bimodal [5]

Otsu’s algorithm

A continuación se muestra un ejemplo de aplicación del método de Otsu, en una imagen de 6x6 en escala de grises con 6 niveles de intensidad [1],

Seleccionando como umbral **t= 3.**

Pi= probabilidad de ocurrencia del nivel de intensidad de gris (i) , en algunas implementaciones lo calculan de 0 a L (número de niveles de intensidad) [2-3].

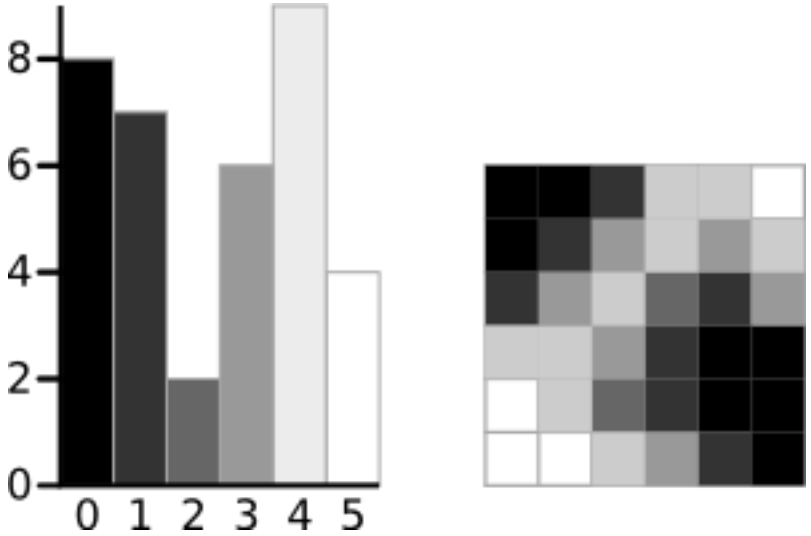


Figura 3. Imagen a escala de grises de 6 niveles y su histograma [5]

Otsu’s algorithm

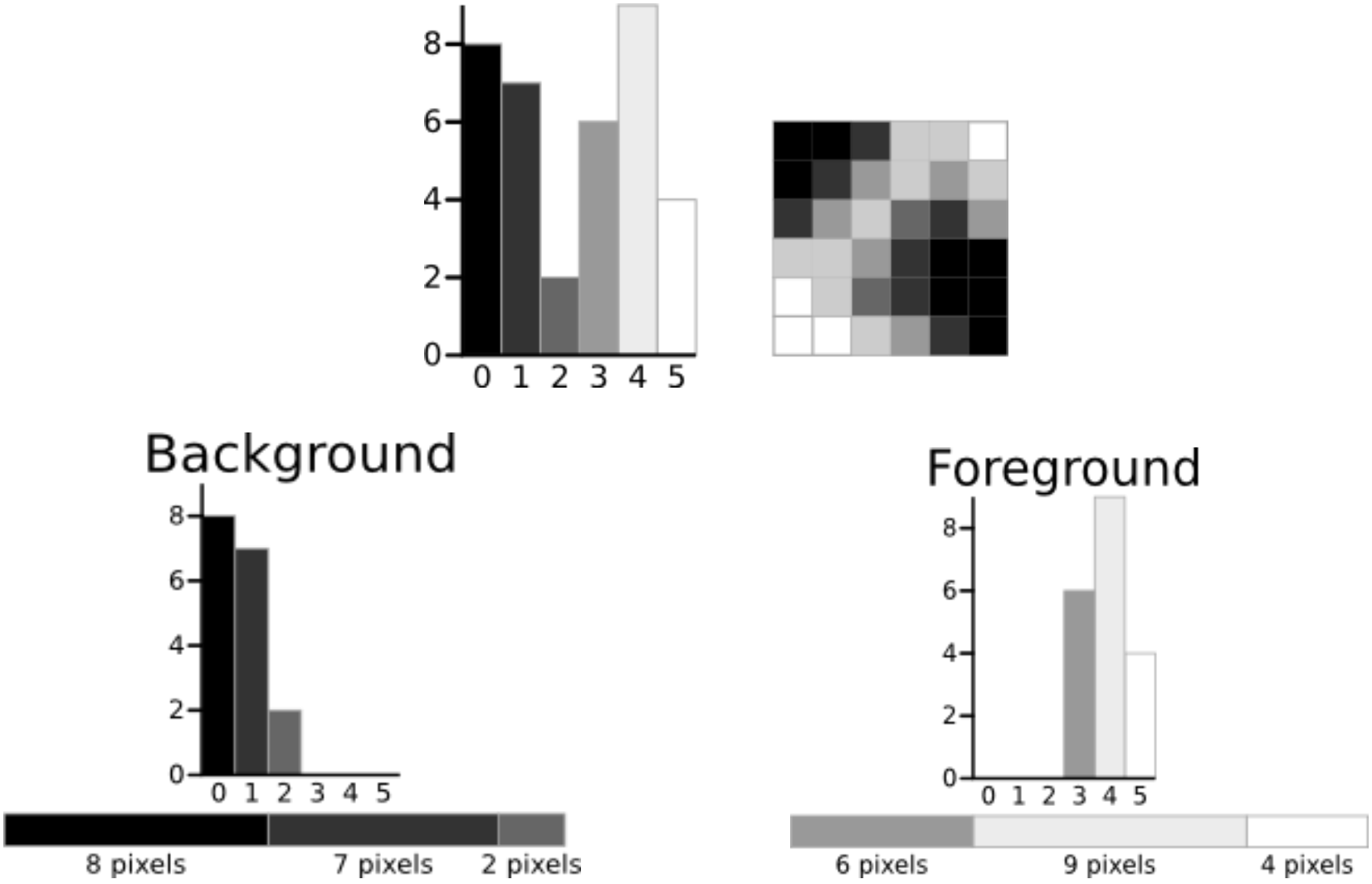
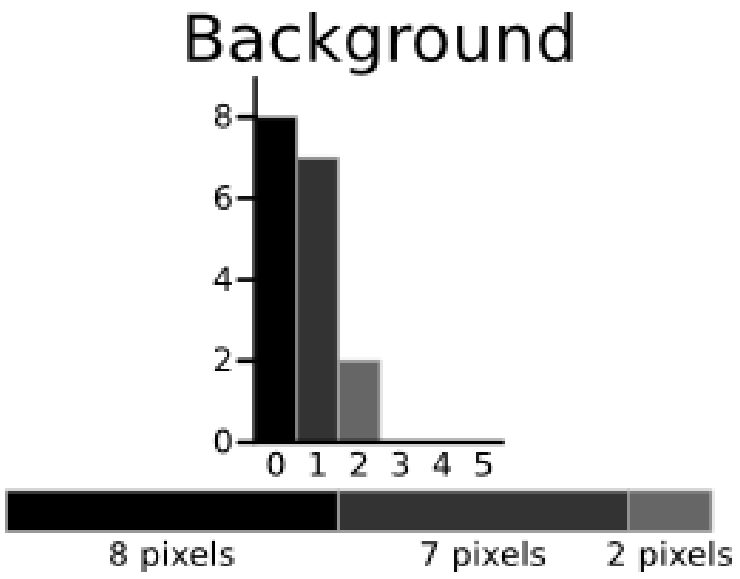


Figura 3.1 Imagen a escala de grises de 6 niveles y su histograma [5]

Otsu's algorithm

1. Calcular las sumas acumulativas (ω) de probabilidades de la clase C1 (niveles de gris [0,…,t-1]) y C2 (niveles de gris [t,…,L-1] [2,3]

Cálculo de $\omega_1(3)$ para la clase 1 (C1).



$$\mathrm{P}i = \frac{fi}{N} \qquad e.\ 1$$

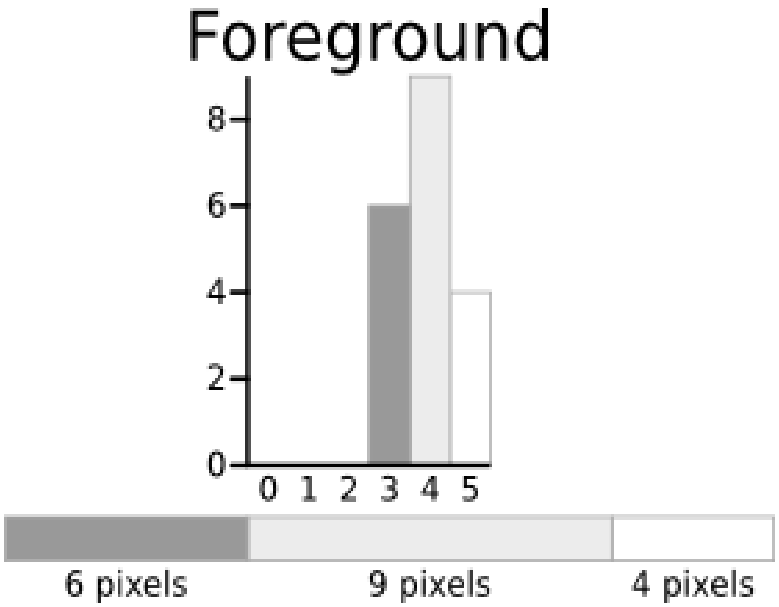
$$\omega_1(t) = \sum_{i=0}^{t-1} \mathrm{P}i \qquad e.\ 2$$

$$\omega_1(3) = \frac{8+7+2}{36} = 0.4722$$

Figura 4. Fondo de la imagen [1]

Otsu's algorithm

Cálculo de $\omega_2(3)$ para la clase 2 (C2).



$$\mathrm{P}i = \frac{fi}{N} \qquad e.\ 1$$

$$\omega_2(t) = \sum_{i=t}^{L-1} \mathrm{P}i \qquad e.\ 3$$

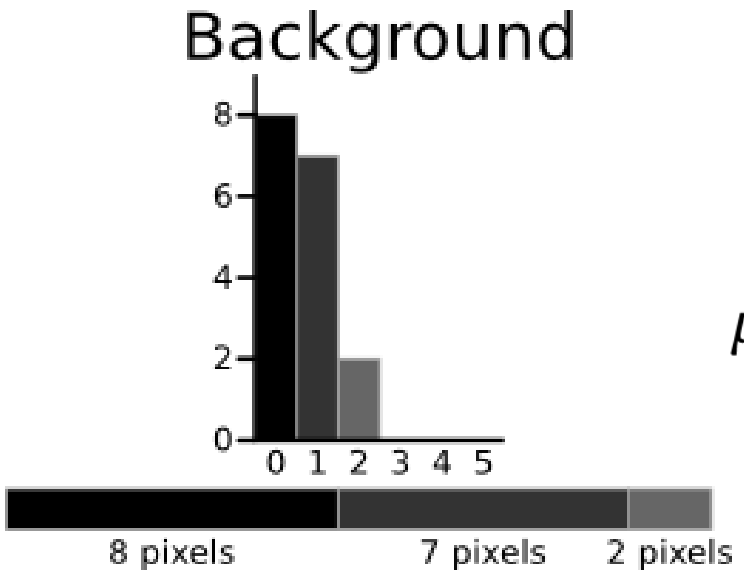
$$\omega_2(3) = \frac{6+9+4}{36} = 0.5278$$

Figura 4. Regiones de interés de la imagen [1]

Otsu's algorithm

2. Calcular la media (μ) para las clases C1 y C2 [2,3].

Cálculo de $\mu_1(3)$ para la clase 1 (C1).



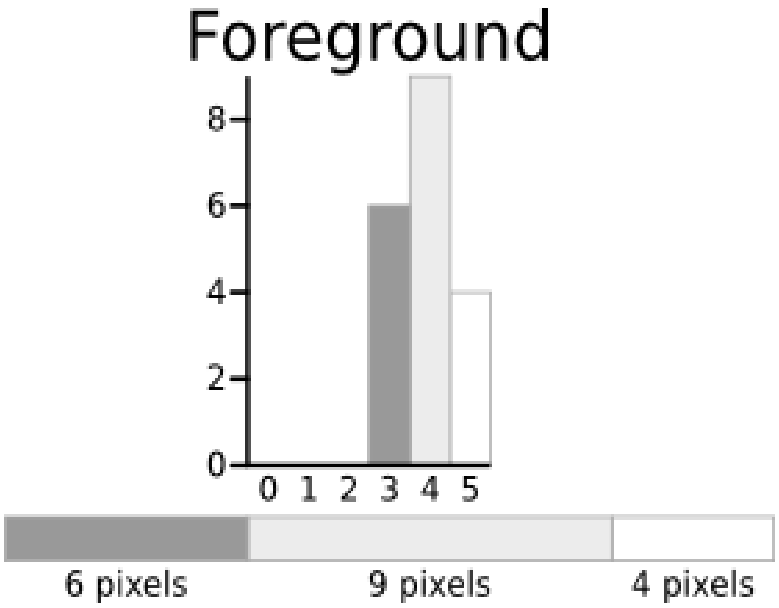
$$\mu_1(t) = \frac{\sum_{i=0}^{t-1} i * \mathrm{P}i}{\omega_1(t)} \qquad e.\ 4$$

$$\mu_1(3) = \frac{(0x0.2222)+(1x0.1944)+(2x0.556)}{0.4722} = 0.6471$$

Figura 4. Fondo de la imagen [1]

Otsu's algorithm

Cálculo de $\mu_2(3)$ para la clase 2 (C2).



$$\mu_2(t) = \frac{\sum_{i=t}^{L-1} i * \mathrm{P}i}{\omega_2(t)} \qquad e.\ 4$$

$$\mu_2(3) = \frac{(3x0.1667)+(4x0.2500)+(5x0.1111}{0.5278} = 3.8947$$

Figura 4. Regiones de interés de la imagen [1]

Otsu's algorithm

3. Calcular la varianza (σ^2) para las clases C1 y C2 [1].

Cálculo de $\sigma^2{}_1(3)$ para la clase 1 (C1).

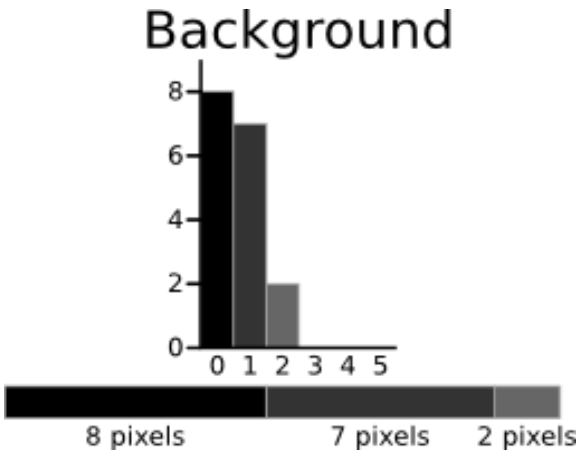


Figura 4. Fondo de la imagen [1]

$$\sigma^2{}_1(t) = \frac{\sum_{i=0}^{t-1} (i - \mu_{1(t)})}{n(0 \ldots t-1)} \qquad e.\ 5$$

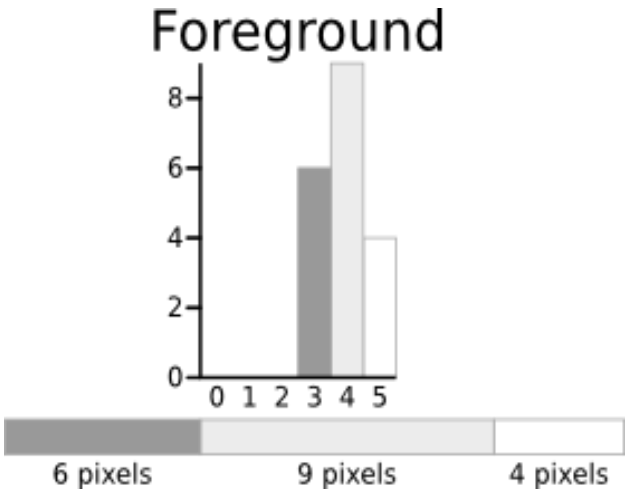
$$\sigma^2{}_1(3) = \frac{((0 - 0.6471)^2\ x\ 8) + ((1 - 0.6471)^2\ x\ 7) + ((2 - 0.6471)^2\ x\ 2)}{17}$$

$$\sigma^2{}_1(3) = \frac{(0.4187\ x\ 8) + (0.1246\ x\ 7) + (1.8304\ x\ 2)}{17}$$

$$\sigma^2{}_{1(3)} = 0.4637$$

Otsu's algorithm

Cálculo de $\sigma^2{}_2(3)$ para la clase 2 (C2).



$$\sigma^2{}_2(t) = \frac{\sum_{i=t}^{L-1} (i - \mu_{2(t)})}{n(t \ldots L-1)} \qquad e.\ 5$$

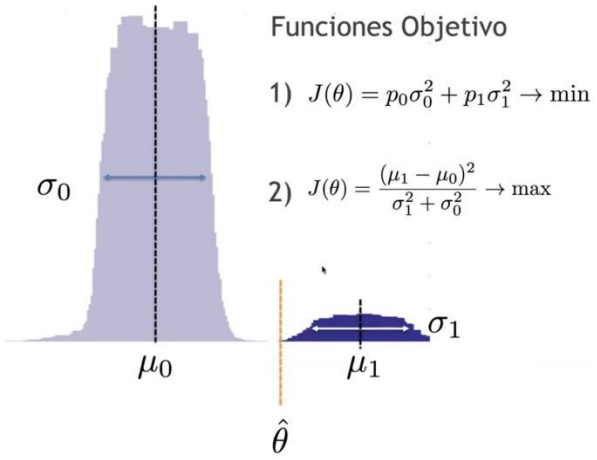
Figura 4. Regiones de interés de la imagen [1]

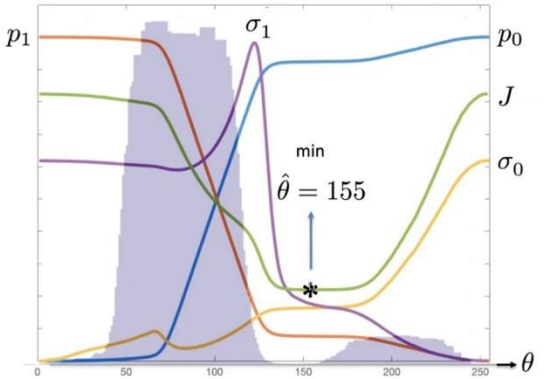
$$\sigma^2{}_2(3) = \frac{((3 - 3.8947)^2\ x\ 6) + ((4 - 3.8947)^2\ x\ 9) + ((5 - 3.8947)^2\ x\ 4)}{19}$$

$$\sigma^2{}_2(3) = \frac{(4.8033\ x\ 6) + (0.0997\ x\ 9) + (4.8864\ x\ 4)}{19}$$

$$\sigma^2{}_{2(3)} = 0.5152$$

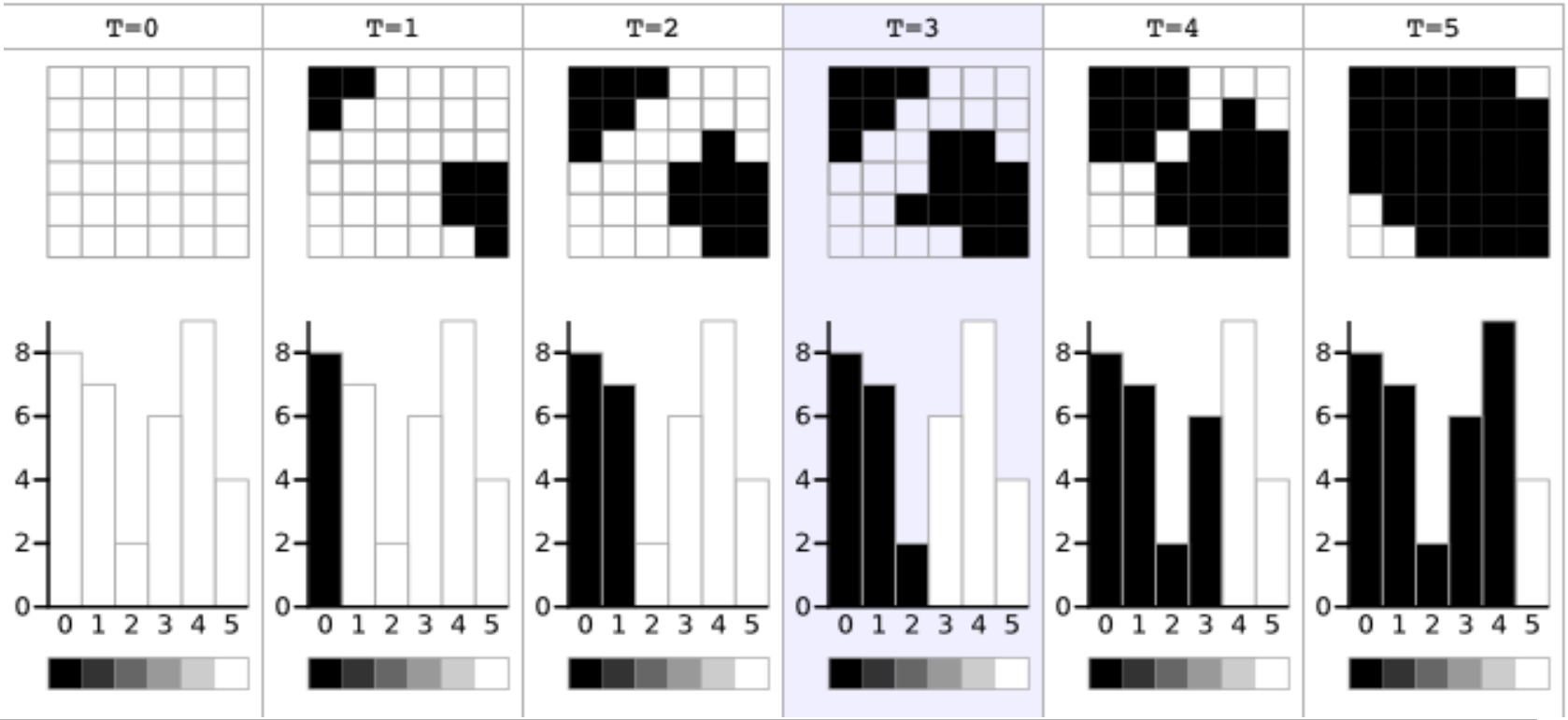
Otsu's algorithm





Resultados: Método de Otsu

Resultados:



Within Class Variance	$\sigma^2_W = 3.1196$	$\sigma^2_W = 1.5268$	$\sigma^2_W = 0.5561$	**$\sigma^2_W = 0.4909$**	$\sigma^2_W = 0.9779$	$\sigma^2_W = 2.2491$

Figura 5. Búsqueda de umbral Otsu [1]

Referencias

- [1]. Greensted, D. A.(Enero 2010) Otsu Thresholding–The Lab Book Pages. Abril 30, 2020, Disponible en : http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html
- [2]. Parra, J.A.. (Octubre 2005). Segmentación por umbralización. Octubre 28, 2019, de Universidad Nacional Quilmes Sitio web: https://es.slideshare.net/JorgeAntonioParraSerquen/segmentacin-por-umbralizacin-mtodo-de-otsu
- [3]. Gonzalez, R. C., & Woods, R. E. (2002). Procesamiento Digital de Imagenes.
- [4]. Gil, S. (2012). Experimentos de física. Usando las TIC y elementos de bajo costo. Buenos Aires, Argentina: Alfaomega.
- [5]. Osorio, J. A. C., Muriel, A., & Vargas, J. A. M. (2011). Comparación cualitativa y cuantitativa de las técnicas básicas de umbralización global basadas en histogramas para el procesamiento digital de imágenes. Scientia et technica, 3(49), 266-272.
- [6]. 123rf. Imagen de hormigas Disponible en https://es.123rf.com/imagenes-de-archivo/hormiga.html?sti=lo8gf4upfkrrwk1isa|&mediapopup=95279143

Otsu’s algorithm in matlab

Thegraythresh function uses Otsu's method, which chooses the threshold to minimize the intraclass variance of the black and white pixels.

Function graythresh takes an image, computes its histogram, and then finds the threshold value that maximizes σ_B^2 . The threshold is returned as a normalized value between 0.0 and 1.0.

T = graythresh(f)

Because the threshold is normalized to the range [0, 1], it must be scaled to the proper range before it is used.

Universidad Veracruzana

Global Thresholding

```
I = imread('coins.png');
```

Calculate a threshold using `graythresh`. The threshold is normalized to the range [0, 1].

```
level = graythresh(I)
```

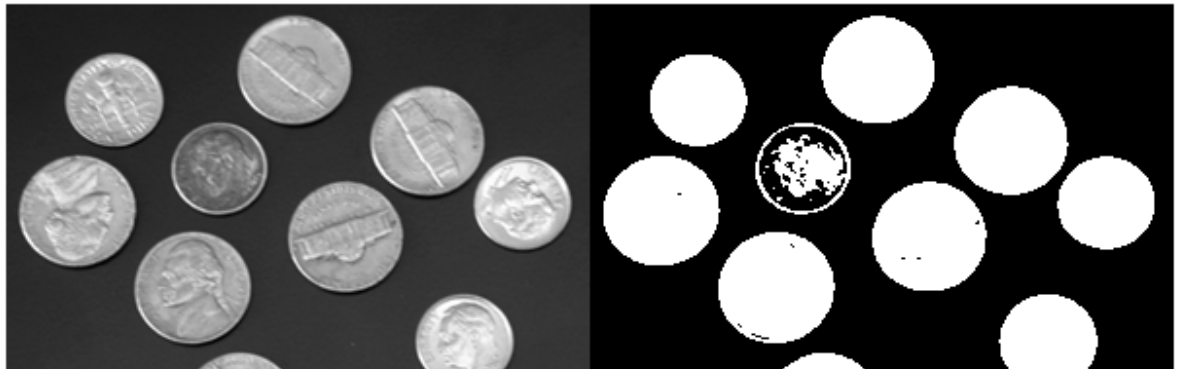
```
level = 0.4941
```

Convert the image into a binary image using the threshold.

```
BW = imbinarize(I,level);
```

Display the original image next to the binary image.

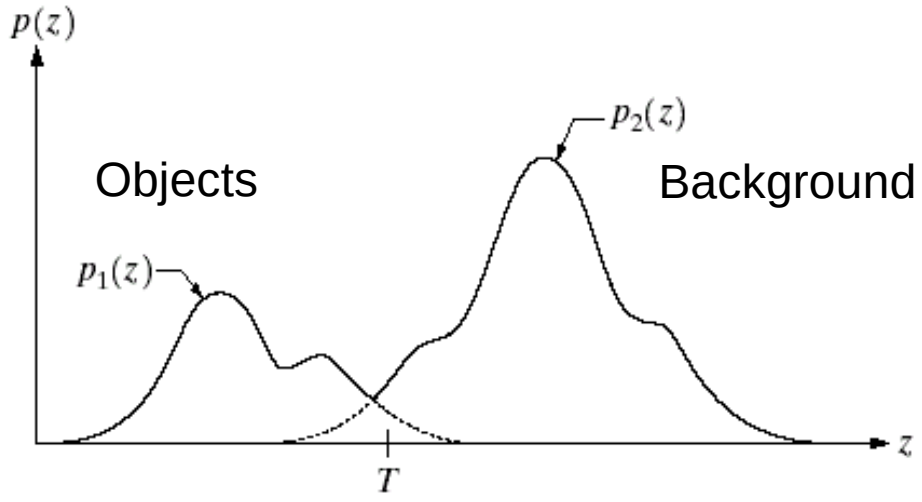
```
imshowpair(I,BW,'montage')
```



Optimal Global and Adaptive Thresholding

Suppose that an image contains only two principal gray-level regions. Let z denote gray-level values. We can view these values as random quantities, and their histogram may be considered an estimate of their probability density function (PDF), $p(z)$, the mixture parameters are proportional to the relative areas of the dark and light regions.

FIGURE 10.32 Gray-level probability density functions of two regions in an image.



Optimal Global and Adaptive Thresholding

The mixture probability density function describing the overall gray-level variation in the image is

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{10.3-5}$$

$$P_1 + P_2 = 1 \tag{10.3-6}$$

Our main objective is to select the value of T that minimizes the average error, the probability of *erroneously* classifying a background point as an object point is

$$E_1(T) = \int_{-\infty}^{T} p_2(z)dz \tag{10.3-7}$$

Optimal Global and Adaptive Thresholding

Similarly, the probability of erroneously classifying an object point as background is

$$E_2(T) = \int_T^{\infty} p_1(z)dz \qquad (10.3\text{-}8)$$

The the overall probability of error is

$$E(T) = P_2 E_1(T) + P_1 E_2(T) \qquad (10.3\text{-}9)$$

To find the threshold value for which this error is minimal requires differentiating $E(T)$ with respect to T and equating the result to 0.

Optimal Global and Adaptive Thresholding

Obtaining an analytical expression for T requires that we know the equations for the two PDFs. One of the principal densities used in this manner is the Gaussian density.

$$p(z) = \frac{P_1}{\sqrt{2\pi\sigma_1}} e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\mu\sigma_2}} e^{-\frac{(z-\mu_2)^2}{2\sigma_2^2}} \qquad (10.3\text{-}11)$$

Special Cases

If $\sigma_1^2 = \sigma_2^2$

If $P_1 = P_2$

If $\sigma = 0$

Other densities

Optimal Global and Adaptive Thresholding

In order to determine the presence or absence dominant modes in the PDF, a minimum mean-square-error approach may be used to estimate gray-level PDF.

$$e_{\text{ms}} = \frac{1}{n}\sum_{i=1}^{n}[p(z_i) - h(z_i)]^2 \qquad (10.3\text{-}15)$$

In general, determinig analytically the parameters that minimize this mean square error is not a simple matter.

Optimal Global and Adaptive Thresholding

Example

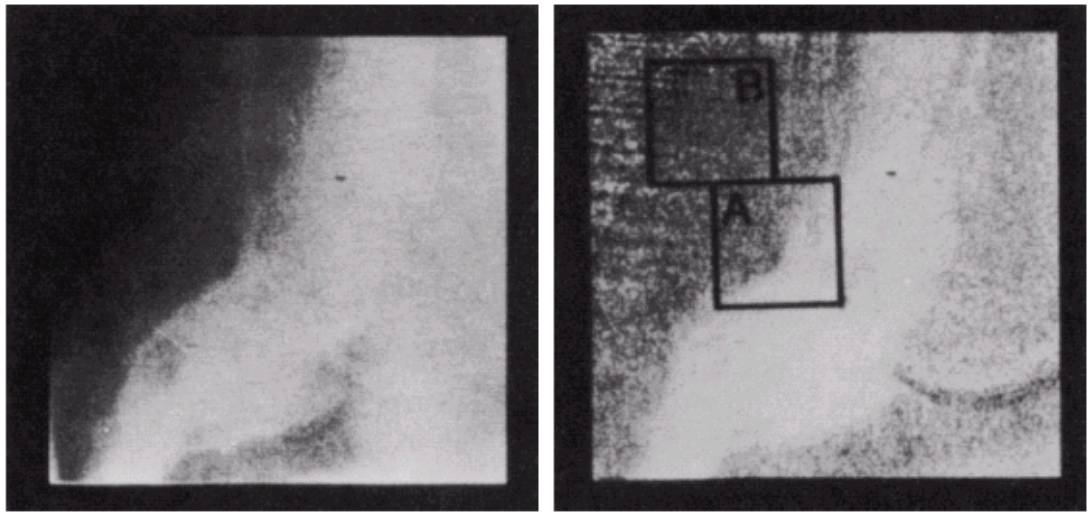
The general problem is to outline automatically the boundaries of heart ventricles in cardioangiograms (X-ray images of a heart that has been injected with a contrast medium)

(1) Each pixel was mapped with a log function
(2) An image obtained before application of the contrast medium was subtracted
(3) Several angiograms were summed in order to reduce random noise

Optimal Global and Adaptive Thresholding

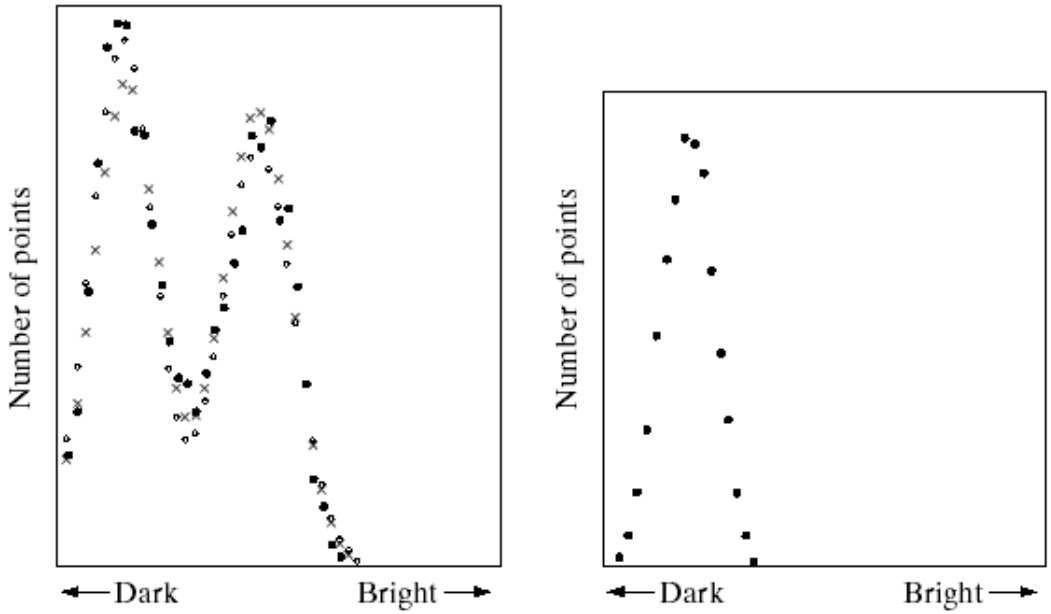
a b

FIGURE 10.33 A cardioangiogram before and after preprocessing. (Chow and Kaneko.)



Optimal Global and Adaptive Thresholding

Each preprocessed image was subdivided into 49 regions (original images are of size 256 × 256 pixels). Each of the 49 resulting overlapped regions contained 64 × 64 pixels.



a b

FIGURE 10.34 Histograms (black dots) of (a) region *A*, and (b) region *B* in Fig. 10.33(b). (Chow and Kaneko.)

Optimal Global and Adaptive Thresholding

After all 49 histograms were computed, a test of bimodality was performed to reject the unimodal histograms. The remaining histograms were then fitted by bimodal Gaussian density curves [see Eq. (10.3-11)] using a conjugate gradient hill-climbing method to minimize the error function given in Eq. (10.3-15).

The optimum thresholds were then obtained as explained before. At this stage of the process only the regions with bimodal histograms were assigned thresholds.

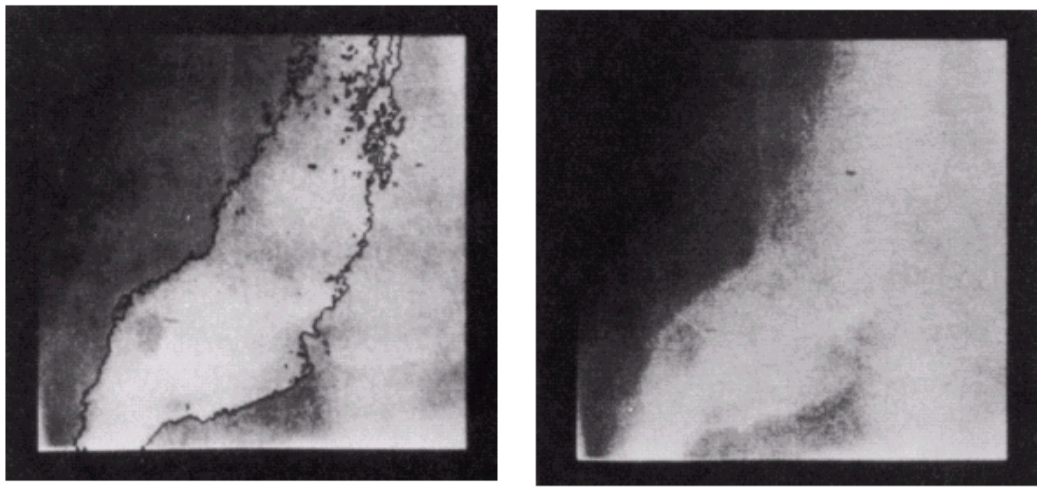
Optimal Global and Adaptive Thresholding

Finally, a binary decision was carried out for each pixel using the rule

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T_{xy} \\ 0 & \text{otherwise} \end{cases}$$

where T_{xy} was the threshold assigned to location (x,y) in the image. Boundaries were obtained by taking the gradient of the binary picture.

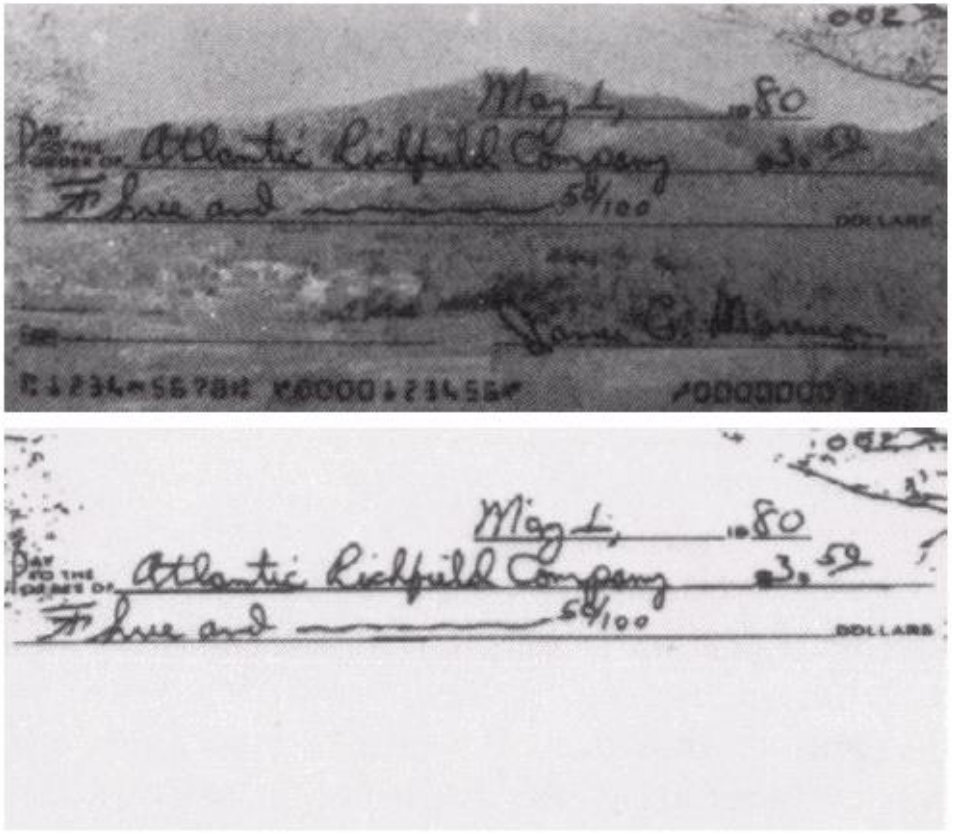
FIGURE 10.35
Cardioangiogram showing superimposed boundaries. (Chow and Kaneko.)



Use of Boundary Characteristics for Histogram Improvement and Local Thresholding

a
b

FIGURE 10.37
(a) Original image. (b) Image segmented by local thresholding. (Courtesy of IBM Corporation.)



Region-Based Segmentation

In this section we discuss segmentation techniques that are based on finding the regions directly.

Let R represent the entire image region. We may view segmentation as a process that partitions R into n subregions, R_1, R_2, …, R_n, such that

(a) $\bigcup_{i=1}^{n} R_i = R.$

(b) R_i is a connected region, $i = 1, 2, \ldots, n.$

(c) $R_i \cap R_j = \varnothing$ for all i and j, $i \neq j.$

(d) $P(R_i) = \text{TRUE}$ for $i = 1, 2, \ldots, n.$

(e) $P(R_i \cup R_j) = \text{FALSE}$ for $i \neq j.$

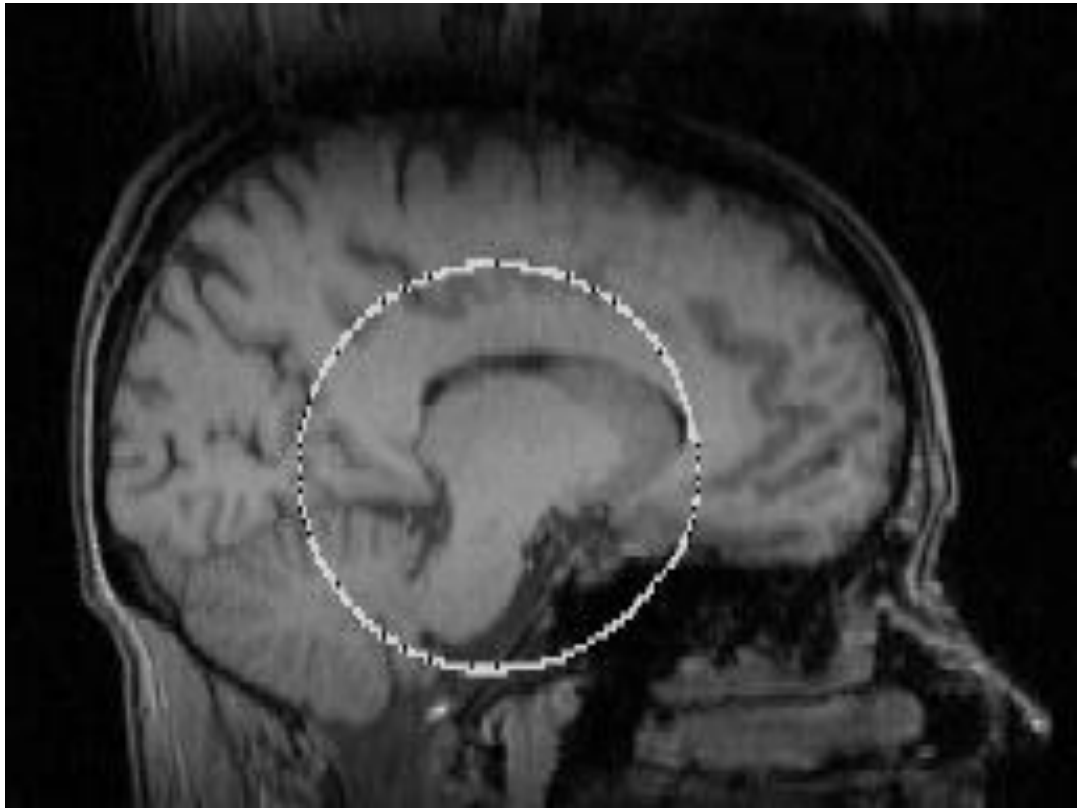
Region Growing

Is a procedure that groups pixels or subregions into larger regions based on predefined criteria. The basic approach is to start with a set of “seed” points and from these grow regions by appending to each seed those neighboring pixels that have properties similar to the seed.

Centroid of clusters can be used as seeds. The selection of similarity criteria depends on the problem. Connectivity information is used in the region-growing process.

Another problem in region growing is the formulation of a stopping rule.

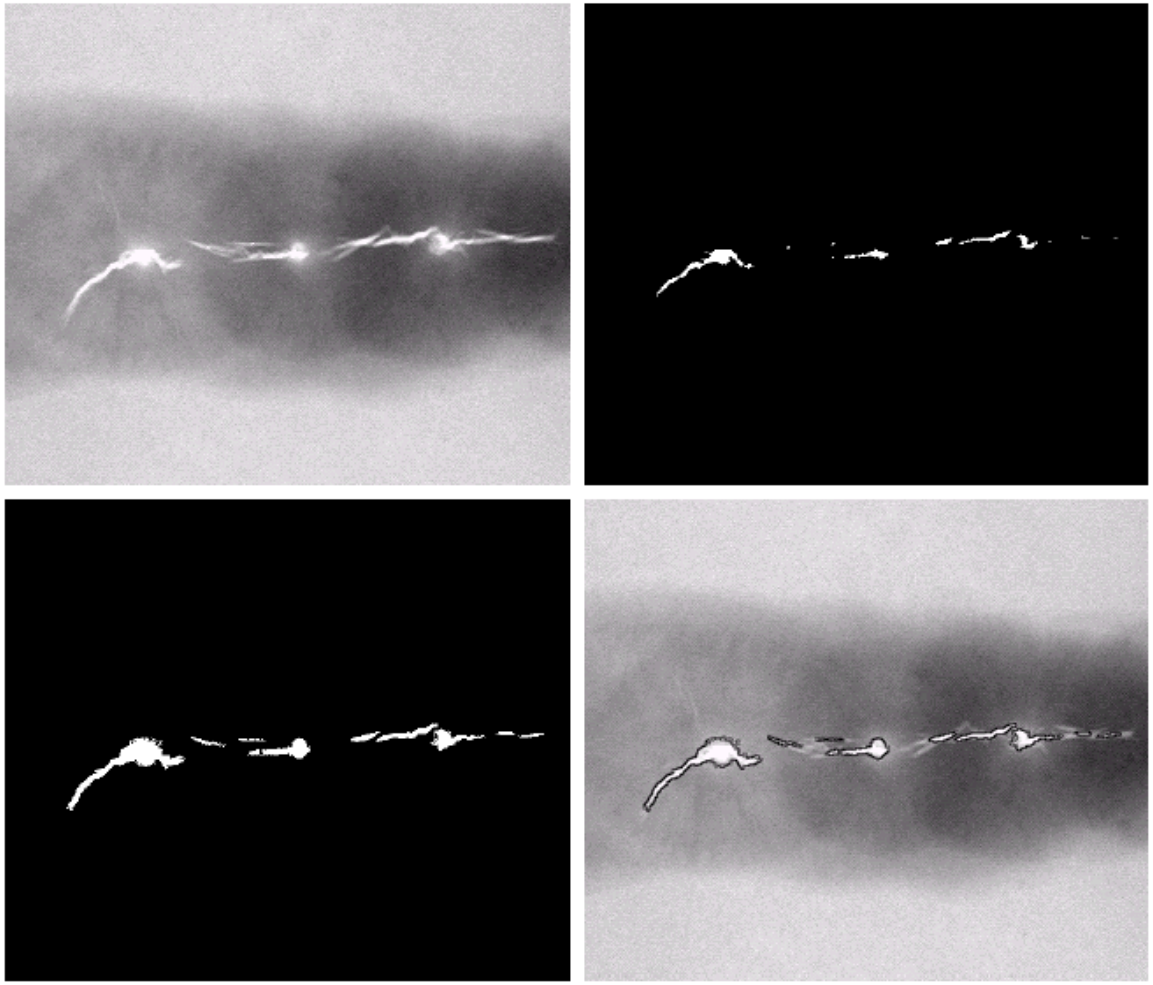
Region Growing



Region Growing

a b
c d

FIGURE 10.40
(a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).



Region Growing

In this particular example we chose two criteria for a pixel to be annexed to a region: (1)The absolute gray-level difference between any pixel and the seed had to be less than 65.

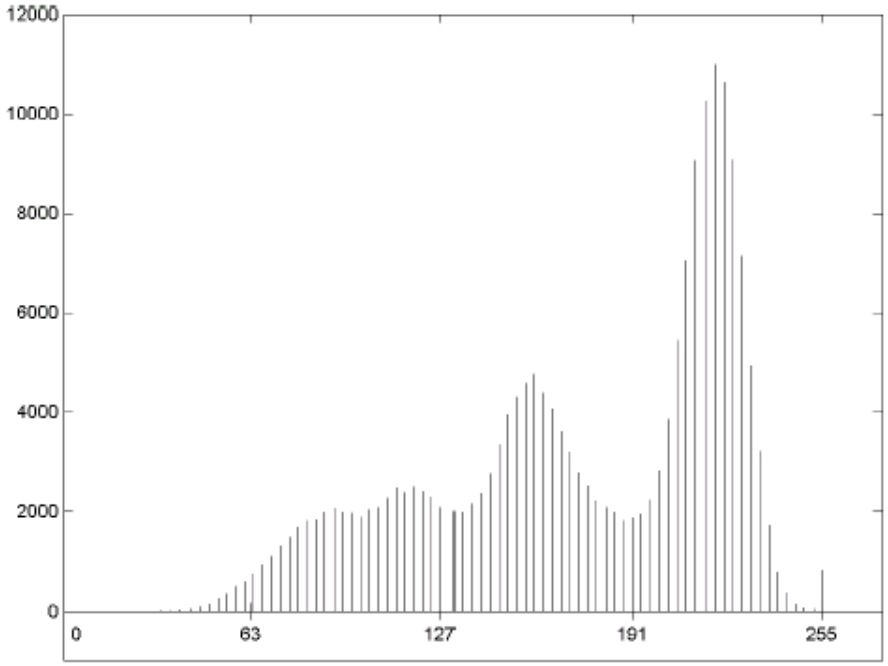


FIGURE 10.41
Histogram of Fig. 10.40(a).

Region Growing

(2) The pixel had to be 8-connected to at least one pixel. If a pixel was found to be connected to more than one region, the regions were merged.

The histogram shown in figure 10.41 is an excellent example of a “clean” multimodal histogram. The use of *connectivity* was fundamental in solving the problem.

Region Growing

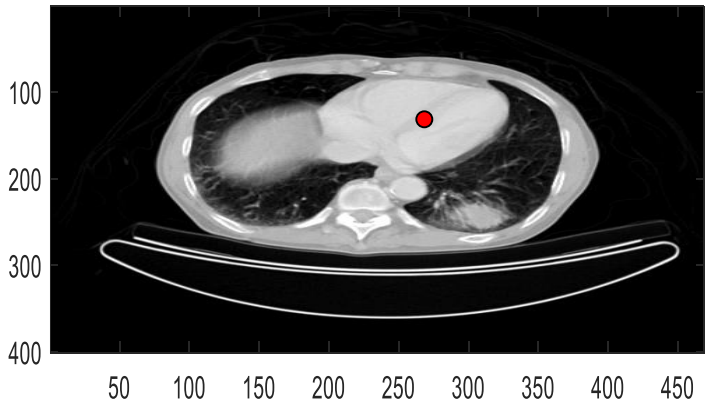
J=regiongrowing(I,x,y,reg_maxdist)

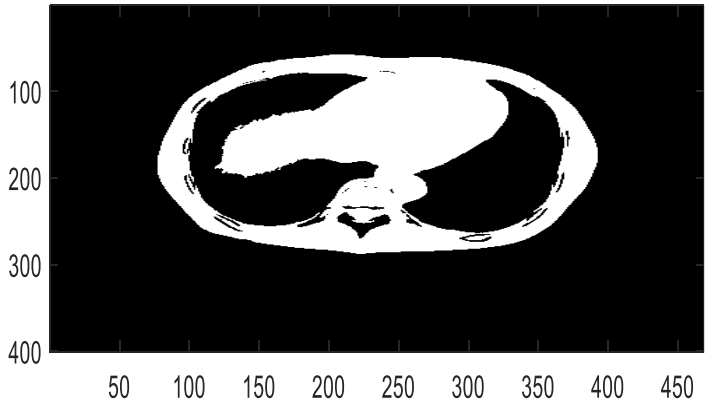
```
% This function performs "region growing" in an image from a specified
% seedpoint (x,y)
%
% J = regiongrowing(I,x,y,t)
%
% I : input image
% J : logical output image of region
% x,y : the position of the seedpoint (if not given uses function getpts)
% t : maximum intensity distance (defaults to 0.2)
```

Universidad Veracruzana

Region Growing

```
I = im2double(imread('medtest.png'));

imagesc(I)
colormap(gray)

[y,x]=ginput(1);

x=round(x);
y=round(y);

J = regiongrowing(I,x,y,0.2);
figure, imshow(I+J);
```





Universidad Veracruzana

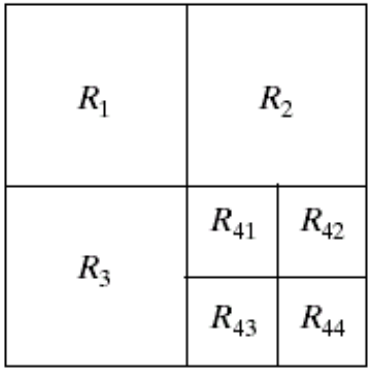
Region Splitting and Merging

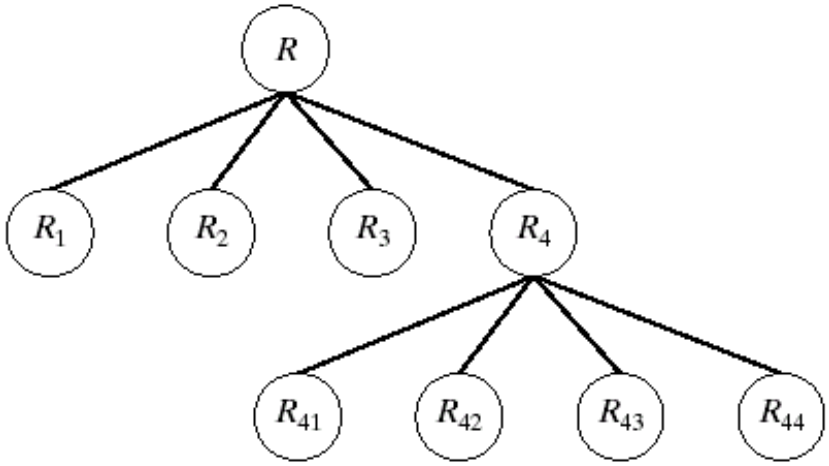
Let R represent the entire image region and select a predicate P. One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_1, $P(R_i)$ = TRUE. We start with the entire region. If $P(R)$ = FALSE, we divide the image into quadrants (*quadtree*).

Region Splitting and Merging

a b

FIGURE 10.42
(a) Partitioned image.
(b) Corresponding quadtree.





Region Splitting and Merging

If only splitting were used, the final partition likely would contain adjacent regions with identical properties. To solve this, two adjacent regions R_j and R_k are merged only if $P(R_j \cup R_k)$ = TRUE.

1. Split into four disjoint quadrants any region R_i for which $P(R_i)$ = FALSE.
2. Merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k)$ = TRUE.
3. Stop when no further merging or splitting is possible.

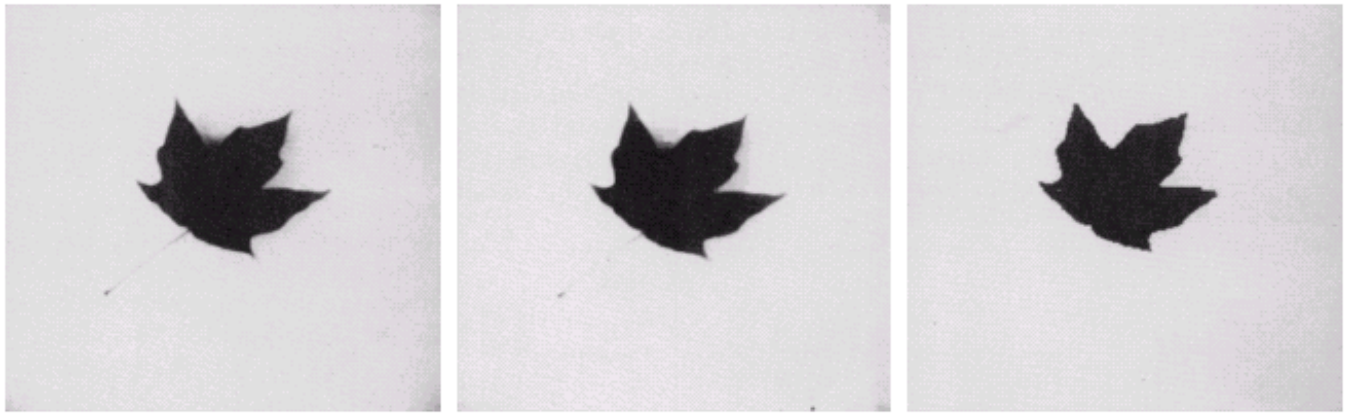
Several variations of the preceding basic theme are possible.

Region Splitting and Merging

Figure 10.43 (a) shows a simple image. We define $P(R_i)$ = TRUE if at least 80% of the pixels in R_i have the property $|z_j - m_i| \leq 2\sigma_i$, where z_j is the gray level of the jth pixel in R_i, m_i is the mean gray level of that region, and σ_i is the standard deviation of the gray levels in R_i.

a b c

FIGURE 10.43
(a) Original image. (b) Result of split and merge procedure. (c) Result of thresholding (a).



Region Splitting and Merging

S = qtdecomp(I,threshold)

S = qtdecomp(I) performs a quadtree decomposition on the grayscale image I and returns the quadtree structure in the sparse matrix S. By default, qtdecomp splits a block unless all elements in the block are equal.

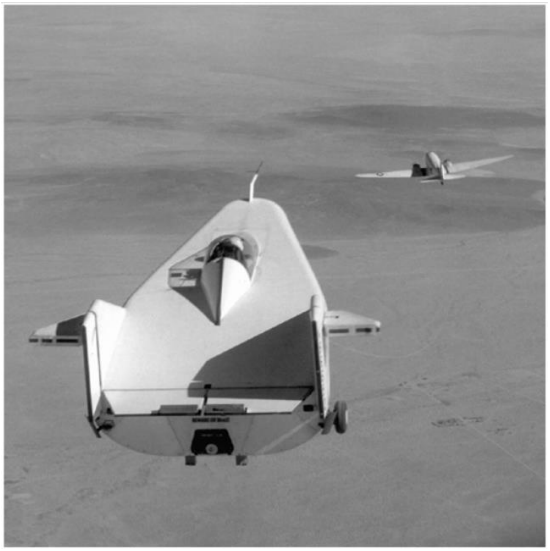
S = qtdecomp(I,threshold) splits a block if the maximum value of the block elements minus the minimum value of the block elements is greater than threshold.

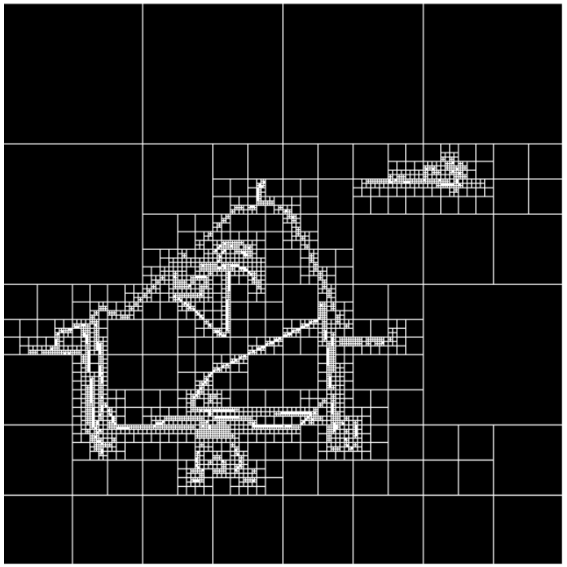
Region Splitting and Merging

I = imread('liftingbody.png');

Perform the quadtree decomposition and display the block representation in a figure.

S = qtdecomp(I,.27);

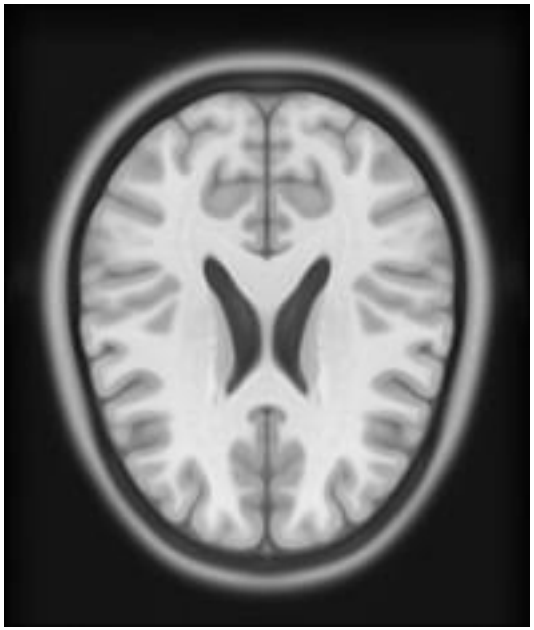


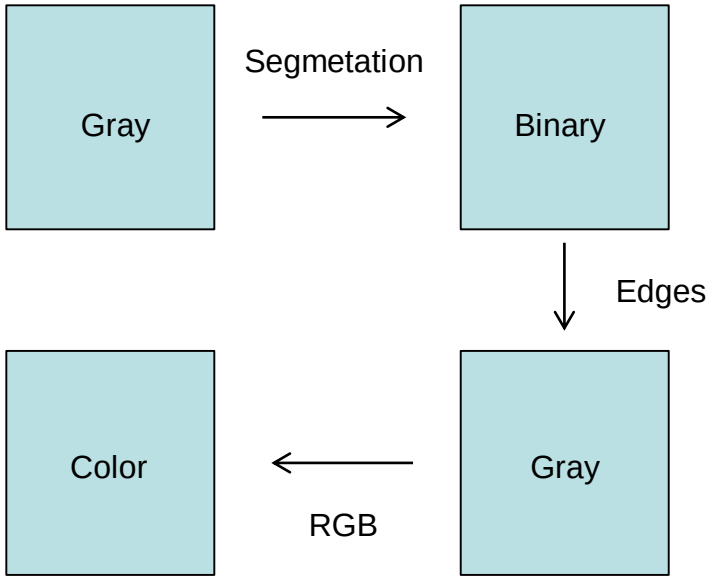


Check: https://la.mathworks.com/help/images/ref/qtdecomp.html

Homework 8

- Segment the gray matter in the following image.
- Use the method of your preference.





Region Splitting and Merging

The concept of *texture segmentation* is based on using measures of texture for the predicates $P(R_i)$, the same applys to other criteria.

From edge detection to contour specification

Non parametric

Edge linking

Active contours (snakes)

Parametric

Hough transform

Active shape models (smart snakes)

Momentos geométricos?

Edge Linking and Boundary Detection

Local Processing

One of the simplest approaches for linking edge points is to analyze the characteristics of pixels in a small neighborhood. All points that are similar according to a set of predefined criteria are linked, forming a boundary of pixels that share those criteria.

Two principal properties: (1) the strength of gradient (2) the direction of the gradient

Edge Linking and Boundary Detection

$$\left|\nabla f(x,y) - \nabla f(x_0,y_0)\right| \le E \qquad (10.2\text{-}1)$$

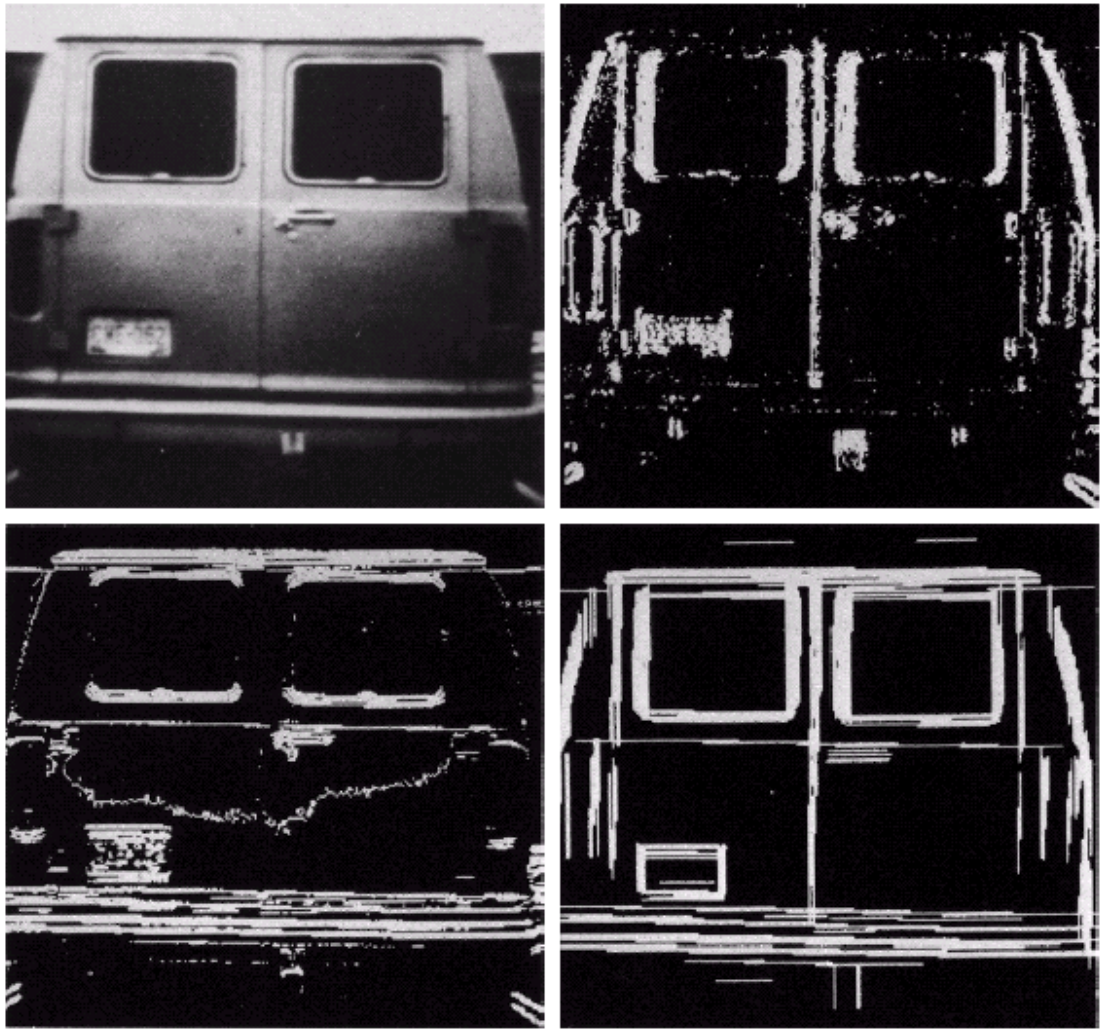
$$\left|\alpha(x,y) - \alpha(x_0,y_0)\right| < A \qquad (10.2\text{-}2)$$

A point in the predefined neighborhood of (x,y) is linked to the pixel at (x,y) if both magnitude and direction criteria are satisfied.

Edge Linking and Boundary Detection

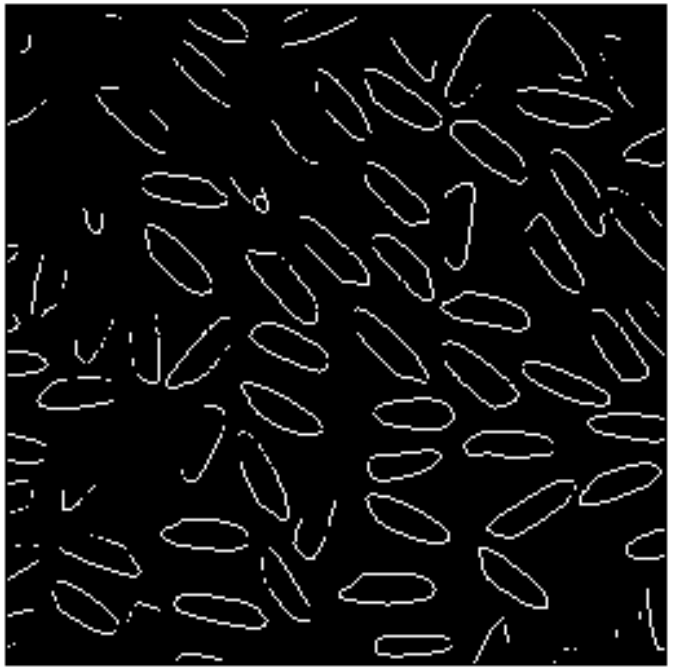
a b
c d

FIGURE 10.16
(a) Input image.
(b) G_y component of the gradient.
(c) G_x component of the gradient.
(d) Result of edge linking. (Courtesy of Perceptics Corporation.)

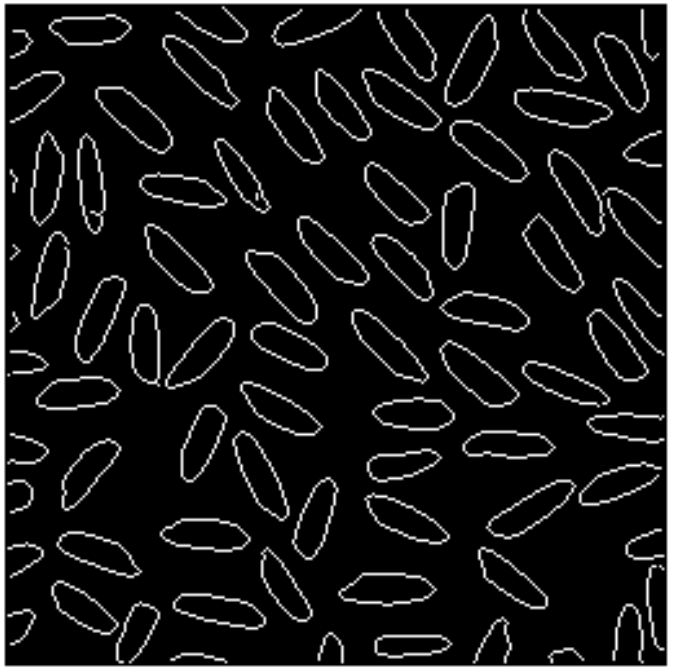


Edge Linking and Boundary Detection

```
>> I = imread('rice.png');
>> I=edge(f,’sobel’);
>> I=edge(f,’canny’);
```



Sobel Filter



Canny Filter

Graph-Theoretic Techniques

Global approach for edge detection and linking based on representing edge segments in the form of a graph and searching the graph for low-cost paths.

Basic definitions. A *graph* $G = (N, U)$ is a finite, nonempty set of nodes N, together with a set U of unordered pairs of distinct elements of N. Each pair (n_i, n_j) of U is called an *arc*.

Graph-Theoretic Techniques

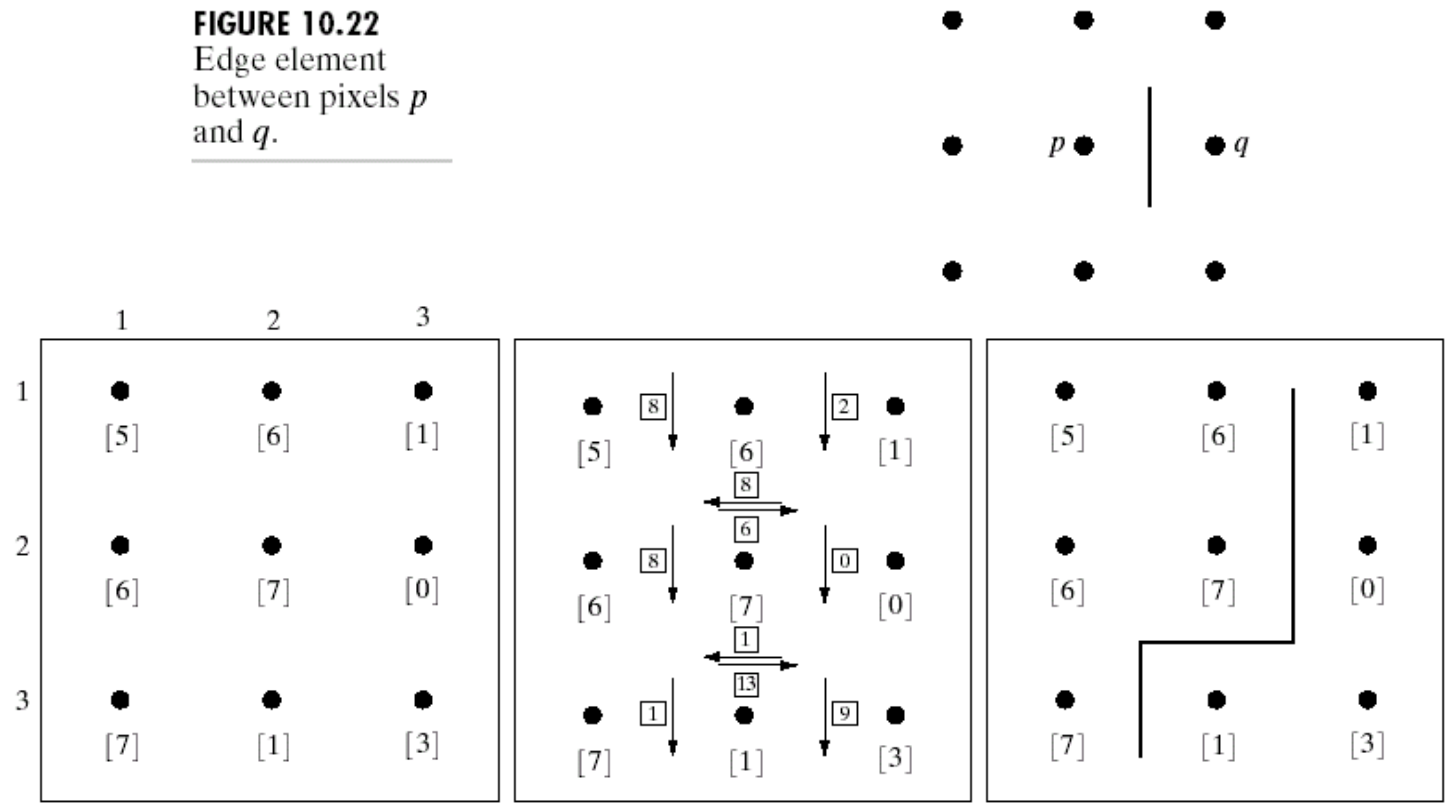
A *cost* $c(n_i,n_j)$ can be associated with every arc(n_i,n_j). A sequence of nodes $n_1,n_2,\ldots,n_k$, with each node n_i, being a successor of node n_{i-1}, is called a *path* from n_i to n_k. The cost of the entire path is

$$c = \sum_{i=2}^{k} c(n_{i-1}, n_i) \qquad (10.2\text{-}5)$$

If we define an *edge element* as the boundary between two pixels p and q, such that p and q are 4-neighbors, as figure 10.22. The concepts just discussed apply to edge detection using the 3 × 3 image shown in figure 10.23.

Graph-Theoretic Techniques

FIGURE 10.22
Edge element between pixels p and q.



a b c

FIGURE 10.23 (a) A 3 × 3 image region. (b) Edge segments and their costs. (c) Edge corresponding to the lowest-cost path in the graph shown in Fig. 10.24.

The numbers in brackets represent gray-level values

Graph-Theoretic Techniques

Each edge element, defined by pixels p and q, has an associated cost, defined as

$$c(p,q) = H - [f(p) - f(q)] \qquad (10.2\text{-}6)$$

where H is the highest gray-level value in the image (7 in this case), and $f(p)$ and $f(q)$ are the gray-level values of p and q, respectively.

Graph-Theoretic Techniques

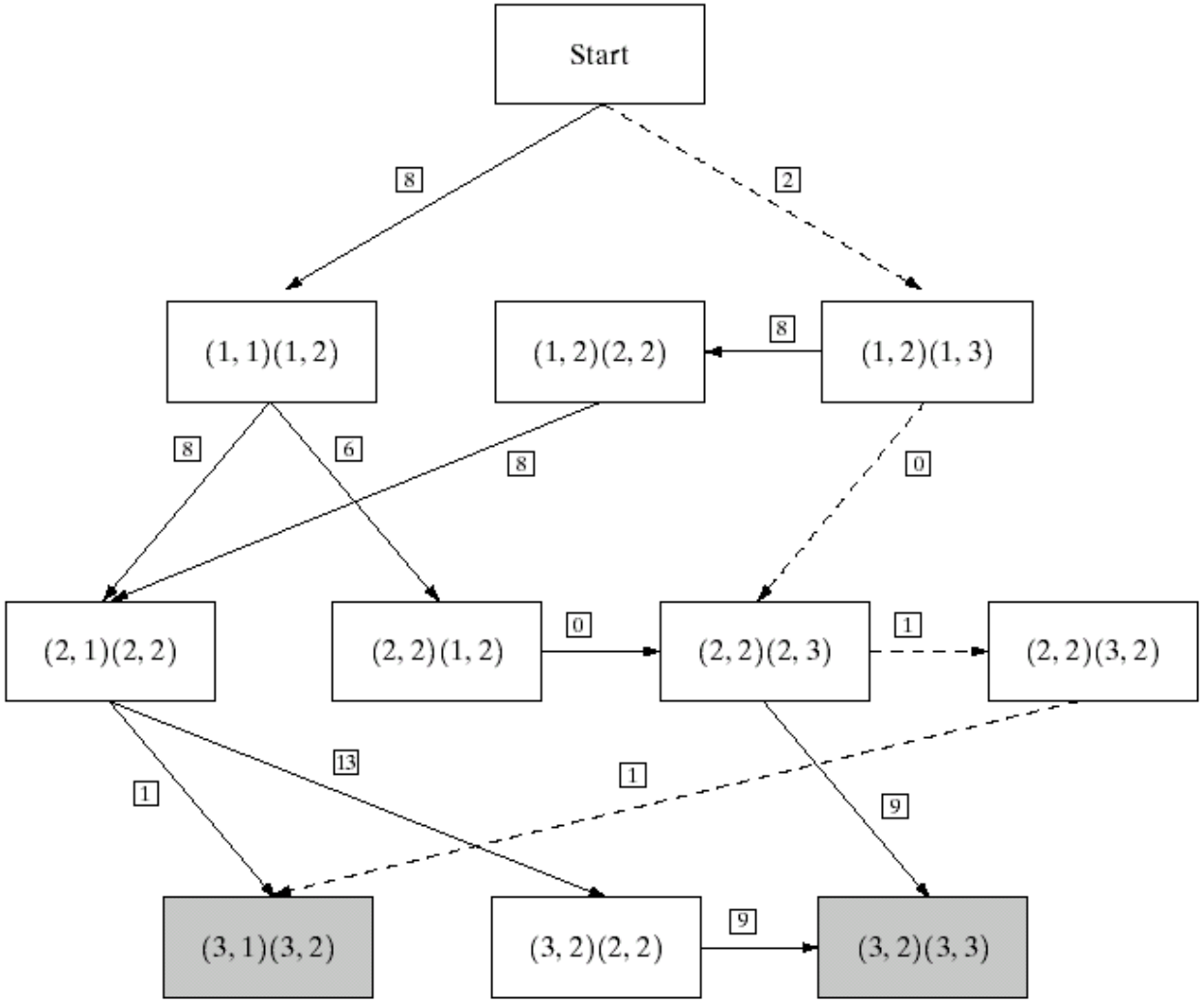


FIGURE 10.24
Graph for the image in Fig. 10.23(a). The lowest-cost path is shown dashed.

Graph-Theoretic Techniques

Use heuristics in order to reduce the search effort

$$r(n) = g(n) + h(n) \qquad (10.2\text{-}7)$$

An algorithm

http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/index.html

Graph-Theoretic Techniques

Step 1: Mark the start node OPEN and set $g(s) = 0$.
Step 2: If no node is OPEN exit with failure; otherwise, continue.
Step 3: Mark CLOSED the OPEN node n whose estimate $r(n)$ computed from Eq. (10.2-7) is smallest. (Ties for minimum r values are resolved arbitrarily, but always in favor of a goal node.)
Step 4: If n is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise, continue.
Step 5: Expand node n, generating all of its successors. (If there are no successors go to step 2.)
Step 6: If a successor n_i is not marked, set

$$r(n_i) = g(n) + c(n, n_i),$$

mark it OPEN, and direct pointers from it back to n.
Step 7: If a successor n_i is marked CLOSED or OPEN, update its value by letting

$$g'(n_i) = \min\left[g(n_i), g(n) + c(n, n_i)\right].$$

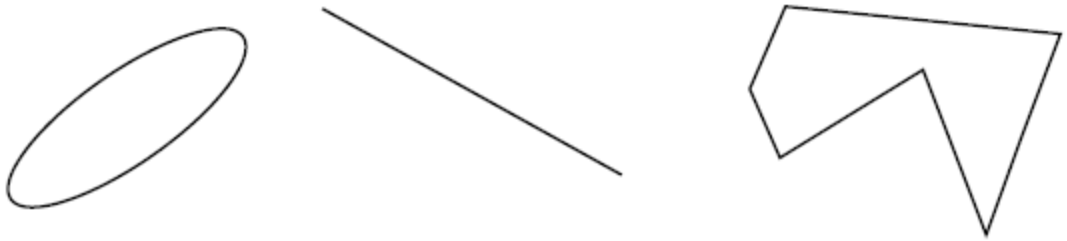
Mark OPEN those CLOSED successors whose g' values were thus lowered and redirect to n the pointers from all nodes whose g' values were lowered. Go to step 2.

Graph-Theoretic Techniques

FIGURE 10.25
Image of noisy chromosome silhouette and edge boundary (in white) determined by graph search.

Active contours, deformable contours, or snakes.

Hough transforms find evidence for well-defined parametrised geometrical shapes.



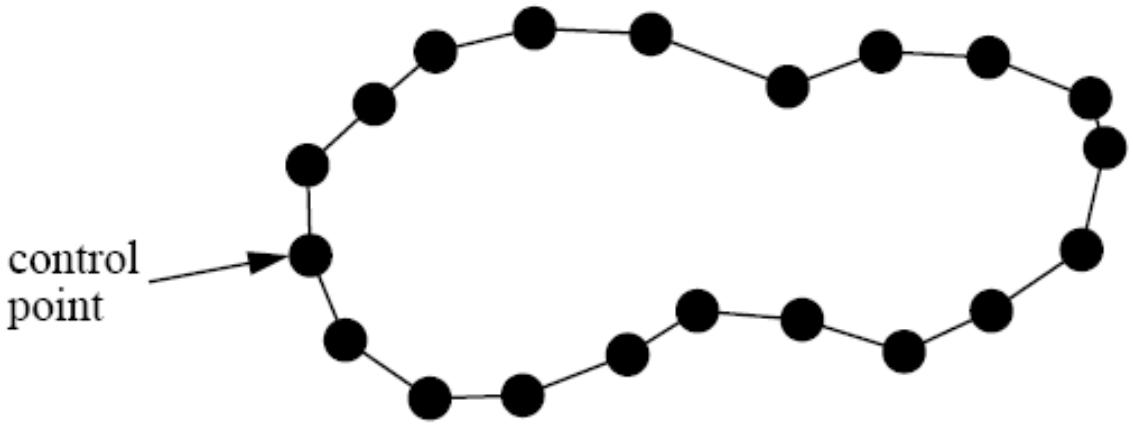
But often what is needed is to find extended shapes that do not have a concise geometrical description, but which have certain *properties* that we can specify. **Snakes** are a way of doing this.

Active contours

Basic snake properties

A snake is a *contour* in the image plane, defined by a set of *control points*.

We will use closed contours, and treat the snake as being a curve that passes through the control points. (This need not always be the case.)



Active contours

The snake's position and shape is made to *evolve* to satisfy

- the *intrinsic* properties that we want it to have,
- the *extrinsic* or image-related properties that we want it to have,
- and any *constraints* that we want to impose.

The algorithm is iterative: snakes are initialised (somehow!) and then some quantity is optimised in steps. Nonetheless, they are computationally very cheap.

The properties we want to specify can be thought of as if they were physical properties of a physical shape. E.g. if the snake is to shrink, think of an elastic band; if to be smooth, a strip of springy metal.

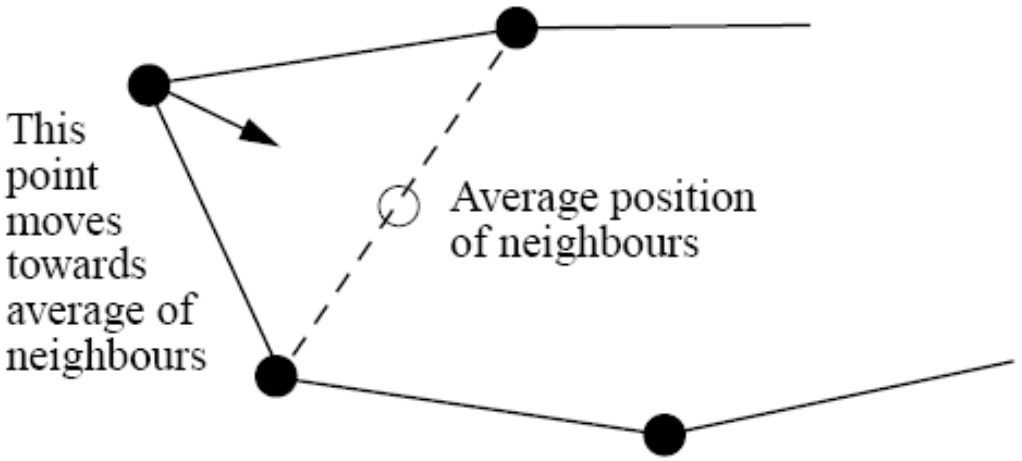
We implement snakes computationally by simulating the physical model of the desired properties.

Active contours

Shrinking and sticking

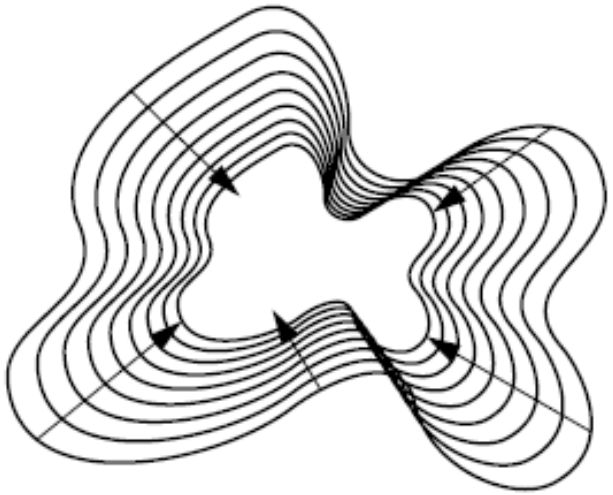
We can use a snake to "shrink-wrap" image structures.

Internal property: the snake should be like a rubber band, which pulls itself together. At each time step, each control point moves closer to its neighbours:



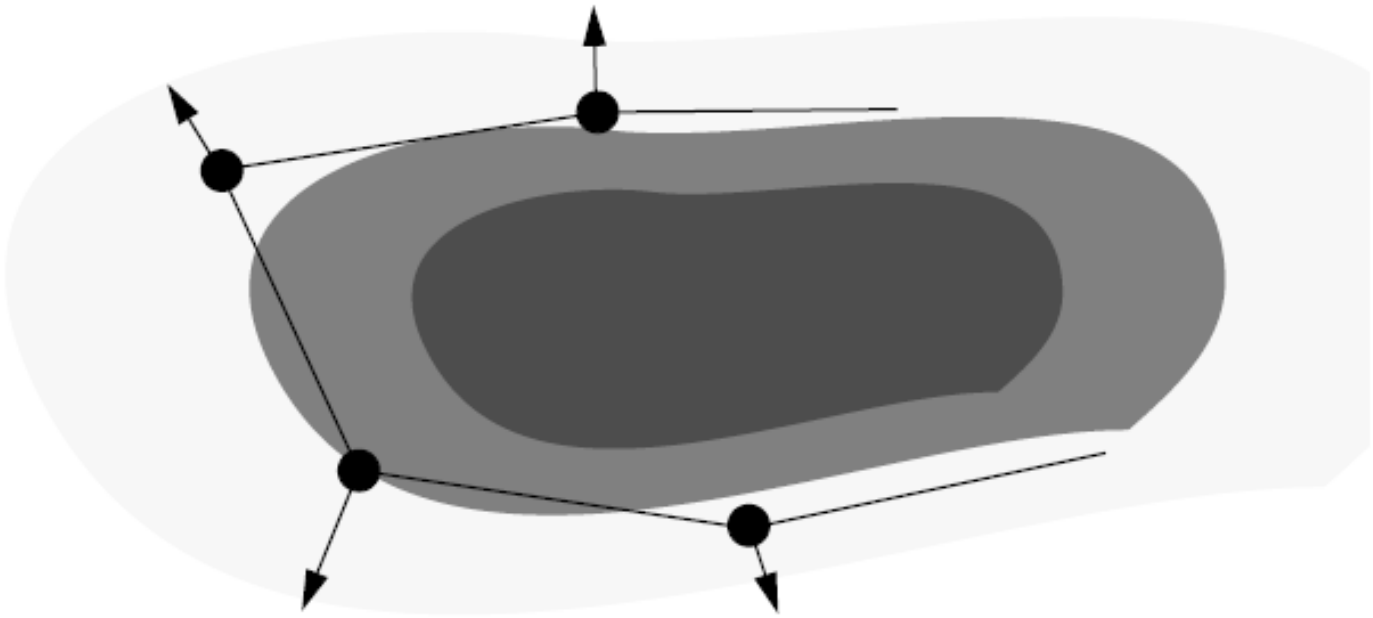
Active contours

Easy to implement: to find the coords of the average of the neighbours, just average their x and y coords separately. Move the point by some fraction of the vector from its current position to the average position. (Usually calculate all the moves before actually doing any of them.)



Active contours

External property: the snake should be repelled by dark parts of the image, if we want it to shrink-wrap a dark structure.



Each control point is pushed towards brighter pixels in its neighbourhood.

Active contours

We can do this by calculating the local *gradient* of image intensity, using x and y differences. Each point is moved in x by some constant times the local x gradient, and likewise for y.

Some local smoothing might be used — but it is not necessary to preprocess the whole image.

For the x coordinate of a control point, a simple formula (no smoothing) is

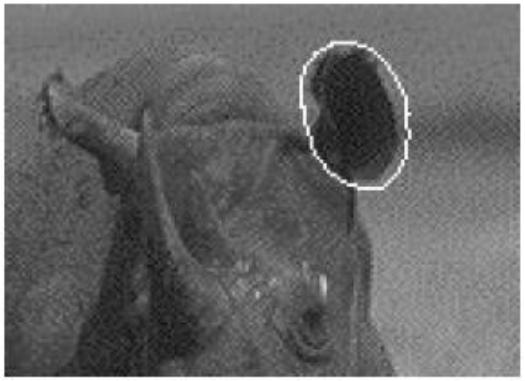
$$x_{\text{new}} = x + \alpha\left(\frac{1}{2}(x_{\text{left}} + x_{\text{right}}) - x\right) + \beta[I(x+1, y) - I(x-1, y)]$$

where x_{left} refers to the control point to the left along the contour, $I(x\text{-}1, y)$ means the grey-level one pixel to the left along the row, and α and β are constants chosen to give the right balance and amount of movement. The formula for updating the y coordinate is similar.

Active contours

A snake governed by these equations is applied to the smoothed image at the top right. It shrinks until it sticks near the edge of the dark region.

The images below show the starting position of the snake, and the position after some iterations, when the elastic inward force is balanced by the outward force from the grey-level gradient.



Active contours

Snake energy

We need a general way of specifying the properties we want the snake to have.

This is done using a quantity called ***energy***, by analogy with physical systems.

We *define* the energy for a snake so that states we want it to be in have low energy.

The energy is made up of internal and external parts added together:

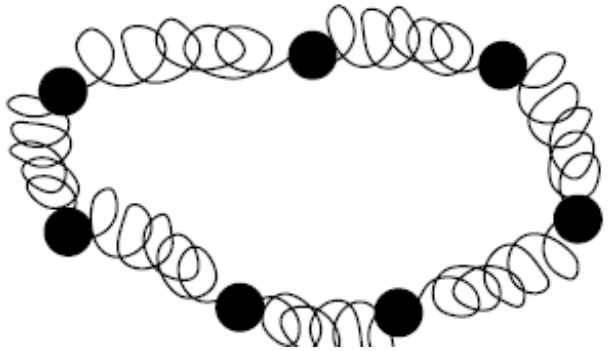
$$E = E_{\text{internal}} + E_{\text{external}}$$

Active contours

To make the snake shrink, the internal energy increases with the length of the snake. Rather than make the energy proportional to the length, we use the sum of the square of the distances between the control points:

$$E_{\text{internal}} = A \sum_{\text{control points}} (\text{distance between control point and neighbour to left})^2$$

This makes the control points tend to spread out evenly along the snake — big gaps contribute a disproportionately high energy. It also models (roughly) a physical system in which the control points are joined by springs.



Active contours

To make the snake avoid dark parts of the image, we make the external energy proportional to minus the sum of the grey-levels lying under the snake. For simplicity, we might just use the grey-levels under the control points.

$$E_{\text{external}} = -B \sum_{\text{control points}} I(\text{location of control point})$$

where I is the grey-level value in the image. A and B are constants, related to α and β.

Converting energy to force

The snake must evolve so that at each step its energy decreases.

Mathematically we differentiate the energy with respect to control point position. This gives a formula showing how to move the control points to reduce the energy. This process is called *gradient descent* on the energy surface.

Active contours

In the physical analogy, this amounts to calculating the *forces* acting on the control points. At each time step, each point is moved a distance proportional to the force on it. This is as if the snake was moving in a viscous fluid.

Differentiating the energy formulae above gives rise to the update rules used for the example:

- to reduce E_{internal} take a small step towards a point half-way between your neighbours;
- to reduce E_{external} take a small step up the grey-level gradient.

Why use an energy formulation?

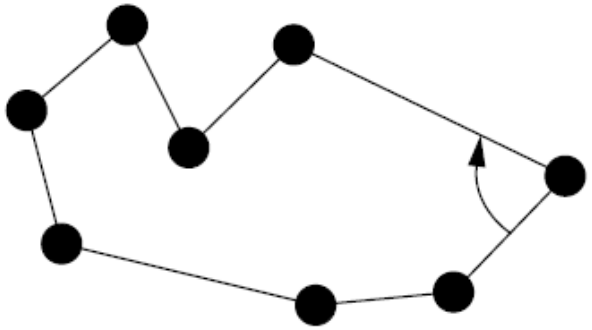
Because we can state the requirements for a "good" snake directly, then generate an algorithm that implements it.

Active contours

Smoothing

Suppose we want the final snake to be smooth, but not necessarily to shrink.

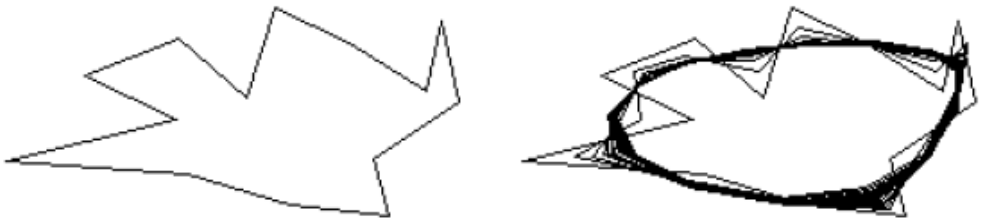
We define an internal energy that increases depending on how rough the snake is.



A sharp angle at a control point means a high curvature. We then use

$$E_{\text{internal}} = A \sum_{\text{control points}} (\text{curvature at control point})^2$$

which results in a force which moves each control point towards a smooth curve through its 4 nearest neighbours.



Active contours

Defining the external energy

It is similarly possible to define the external energy to match particular requirements.

Here the external energy is defined so that the snake wants to lie on regions of high grey-level gradient.

The gradients are shown at the top; but they never actually have to be computed for the whole image.

The snake contracts to lie on the boundaries round the butterfly (except where the elastic energy pulls it tight) even though at some points no clear boundary is defined in the image.

Active contours

$$E_{Snake} = \int_0^1 E_{Snake}(v(s))ds = \int_0^1 E_{Int}(v(s)) + E_{\mathrm{Im}age}(v(s)) + E_{Con}(v(s))ds =$$

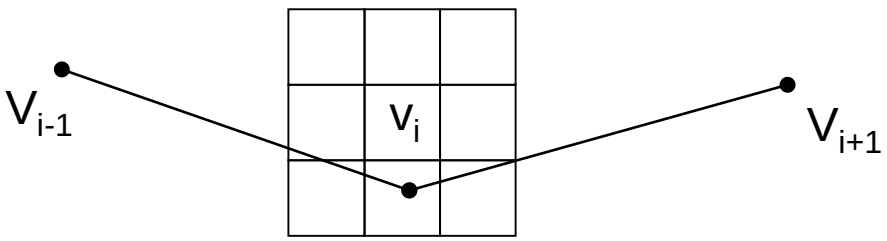
$$E_{Int} = \alpha \left|V_s(s)\right|^2 + \beta \left|V_{ss}(s)\right|^2 / 2$$

$$\left|\frac{dv_i}{ds}\right|^2 \approx \left|V_i - V_{i-1}\right|^2 = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2$$

Active contours

$$\left|\frac{d^2 v_i}{ds^2}\right|^2 \approx \left|V_i - 2V_i + V_{i+1}\right|^2 = (x_{i-1} - 2x_i + x_{i+1})^2 + (y_{i-1} - 2y_i + y_{i+1})^2$$

$V_{Image}=\Delta I(x,y)$



Active contours

Algorithm:

The input is an intensity image I containing the target contour and points $\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_N$ defining the initial position and shape of the snake.

Step.1. For each $\overline{p}_i$, $1 \le i \le N$, search its $M \times M$ neighborhood (search is done in a square window whose length is M) to find the location that minimizes the energy function; move $\overline{p}_i$ to that location.

Step.2. Estimate the curvature of the snake at each point and look for local maxima (i.e., corners); Set β_j to zero for each $\overline{p}_j$ at which the curvature is a local maximum and exceeds a threshold.

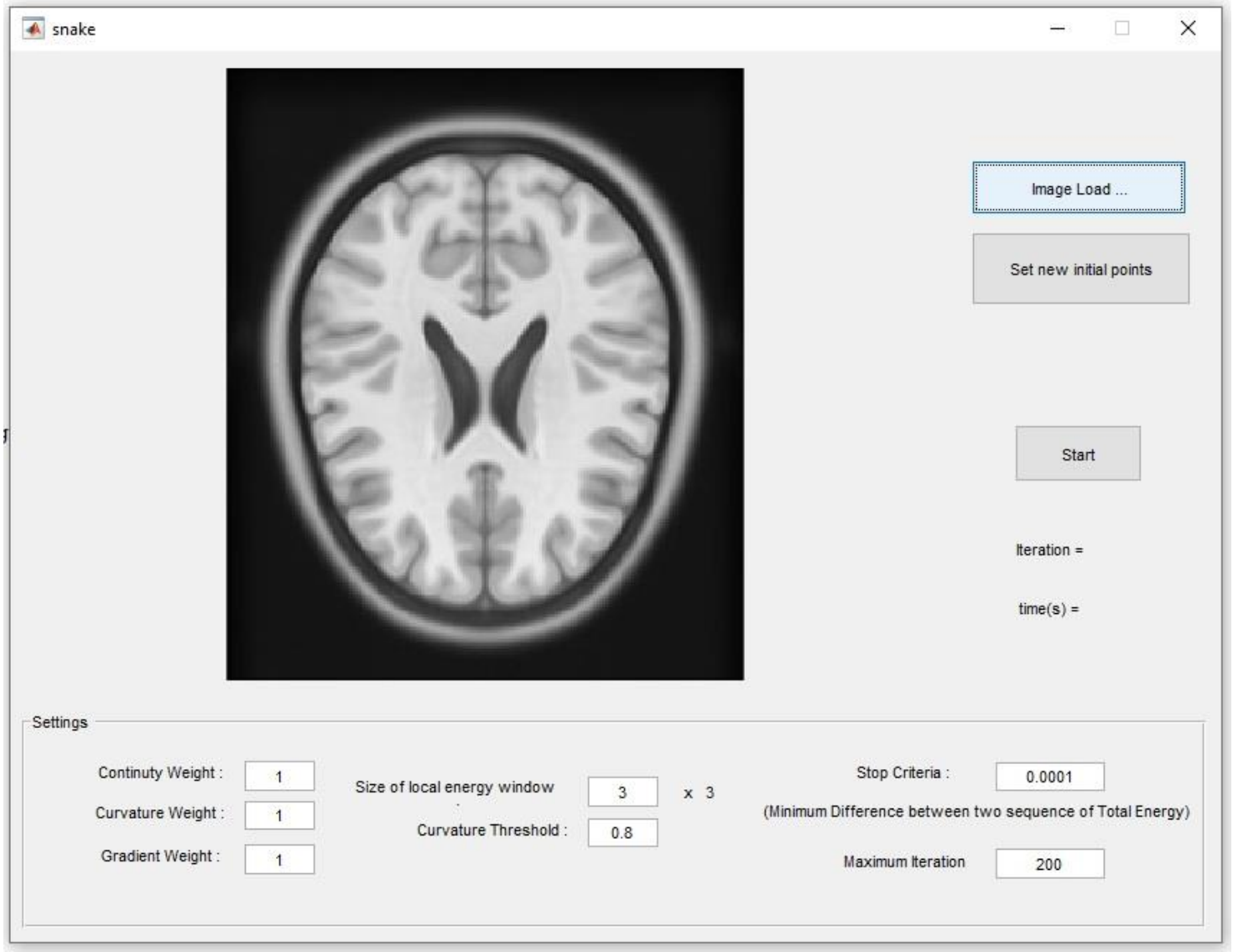
Step.3. Update the value of $\overline{d}$.

Repeat steps 1-3 until only a very small fraction of snake points move in an iteration.

https://www.youtube.com/watch?v=SdqBNFx-uNc

See code:
https://www.mathworks.com/matlabcentral/fileexchange/51220-snake-algorithm

Example

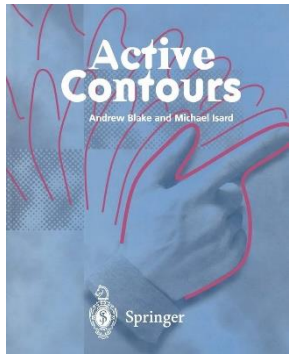

snake
Image Load ...
Set new initial points
Start
Iteration =
time(s) =
Settings
Continuty Weight :
1
Curvature Weight :
1
Gradient Weight :
1
Size of local energy window
3
x 3
Curvature Threshold :
0.8
Stop Criteria :
0.0001
(Minimum Difference between two sequence of Total Energy)
Maximum Iteration
200

Suggested readings

• **Snakes: Active Contour Models.** Michael Kass, Andrew Witkin and Demetri Terzopoulos. International Journal of Computer Vision. Kluwer Academic Publishers. pp. 321-331. 1988.

• **A fast algorithm for active contours and curvature estimation**. Donna j. Williams and Mubarak Sharah. Image understanding. Vol. 55, No. 1, January, pp. 14-26, 1992

• **Active Contours**: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion. Andrew Blake. Springer.



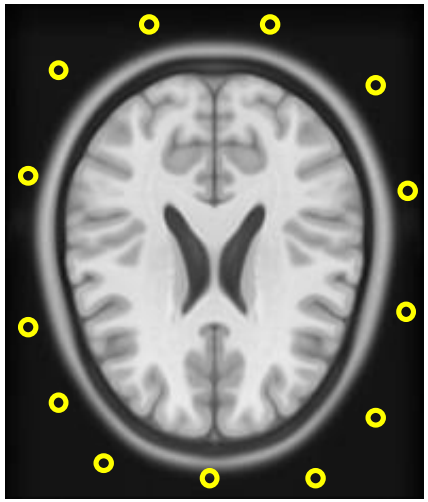
• **Cervical image segmentation using active contours and evolutionary programming over temporary acetowhite patterns.** Aldo Marquez-Grajales, Héctor-Gabriel Acosta-Mesa, Efrén Mezura-Montes, Rodolfo Hernández-Jiménez,. CEC 2016: 3863-3870

Video

Homework 12

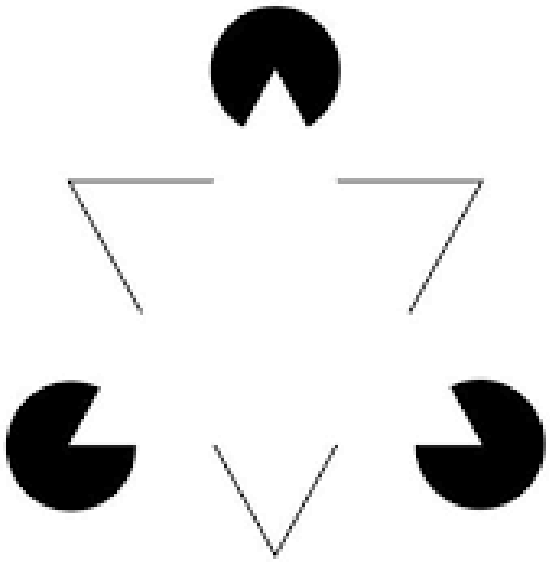
- Implement the basic active contour algorithm.

- Use your algorithm to segment a sintetic the gray matter of the following image.



- Use as many control points as you need arround and outside de skull.

Model Based Vision



Kanizsa triangle

Global Processing via the Hough Transform

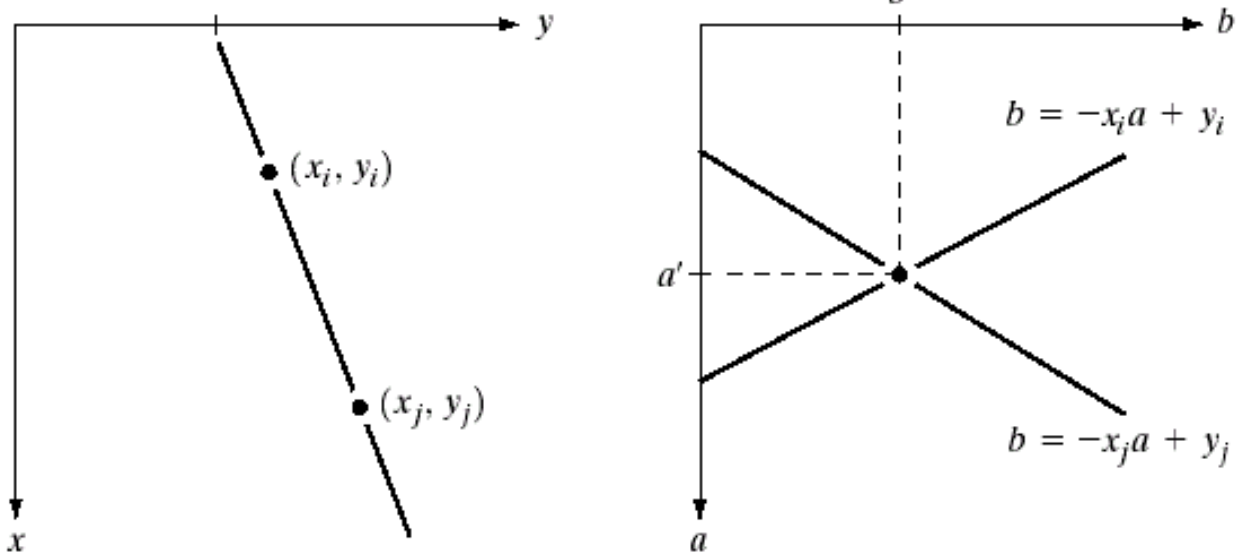
In this section, points are linked by determining first if they lie on a curve of specified shape. Unlike the local analysis method discussed in Section 10.2.1, we now consider global relationships between pixels.

Given n points in an image, suppose that we want to find subsets of these points that lie on straight lines. One possible solution is to first find all lines determined by every pair of points and then find all subsets of points that are close to particular lines. The problem with this procedure is that it involves finding $n(n-1)/2 \sim n^2$ lines and then performing $(n)(n(n-1))/2 \sim n^3$ comparisons of every point to all lines. This approach is computationally prohibitive in all but the most trivial applications.

General equation of a straight line

Hough [1962] proposed an alternative approach, commonly referred to as the *Hough transform*. Consider a point (x_i, y_i) and the general equation of a straight line in slope-intercept form, $y_i = ax_i + b$. Infinitely many lines pass through (x_i, y_i), but they all satisfy the equation $y_i = ax_i + b$ for varying values of a and b. However, writing this equation as $b = -x_i a + y_i$ and considering the ab-plane (also called *parameter space*) yields the equation of a *single* line for a fixed pair (x_i, y_i). Furthermore, a second point (x_j, y_j) also has a line in parameter space associated with it, and this line intersects the line associated with (x_i, y_i) at (a', b'), where a' is the slope and b' the intercept of the line containing both (x_i, y_i) and (x_j, y_j) in the xy-plane. In fact, all points contained on this line have lines in parameter space that intersect at (a', b'). Figure 10.17 illustrates these concepts.

Global Processing via the Hough Transform



a b

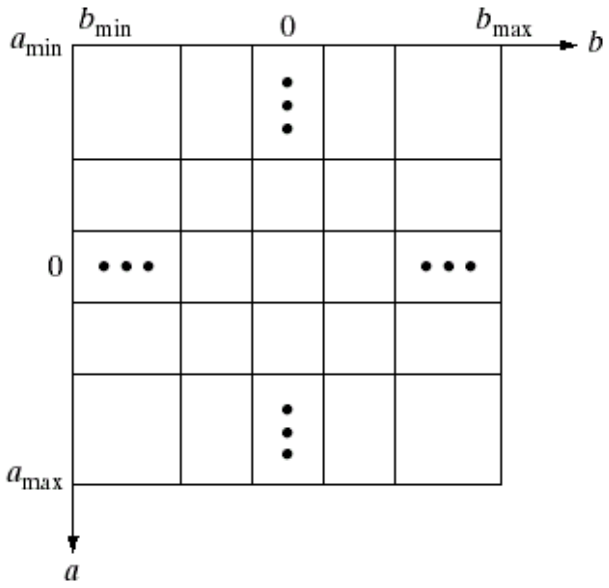
FIGURE 10.17
(a) xy-plane. (b) Parameter space.

Global Processing via the Hough Transform

The computational attractiveness of the Hough transform arises from subdividing the parameter space into so-called *accumulator cells*, as illustrated in Fig. 10.18, where (a_{max}, a_{min}) and (b_{max}, b_{min}) are the expected ranges of slope and intercept values. The cell at coordinates (i, j), with accumulator value $A(i, j)$, corresponds to the square associated with parameter space coordinates (a_i, b_j). Initially, these cells are set to zero. Then, for every point (x_k, y_k) in the image plane, we let the parameter a equal each of the allowed subdivision values on the a-axis and solve for the corresponding b using the equation $b = -x_k a + y_k$. The resulting b's are then rounded off to the nearest allowed value in the b-axis. If a choice of a_p results in solution b_q, we let $A(p, q) = A(p, q) + 1$. At the end of this procedure, a value of Q in $A(i, j)$ corresponds to Q points in the xy-plane lying on the line $y = a_i x + b_j$. The number of subdivisions in the ab-plane determines the accuracy of the colinearity of these points.

Global Processing via the Hough Transform

FIGURE 10.18
Subdivision of the parameter plane for use in the Hough transform.



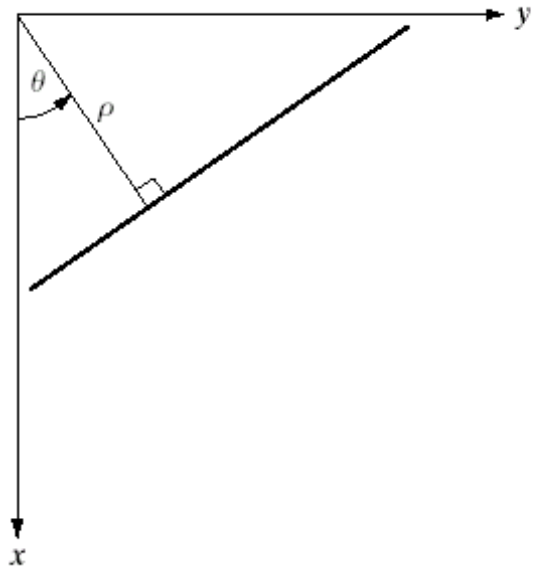
Normal representation of a straight line

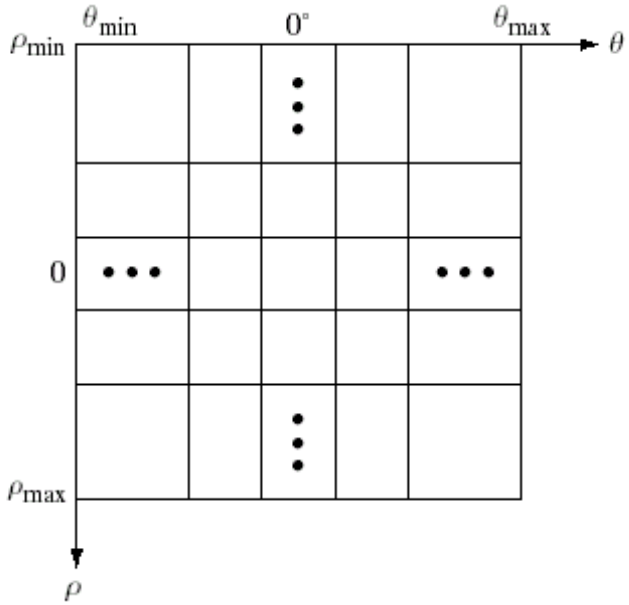
Note that subdividing the a axis into K increments gives, for every point (x_k, y_k), K values of b corresponding to the K possible values of a. With n image points, this method involves nK computations. Thus the procedure just discussed is *linear* in n, and the product nK does not approach the number of computations discussed at the beginning of this section unless K approaches or exceeds n.

A problem with using the equation $y = ax + b$ to represent a line is that the slope approaches infinity as the line approaches the vertical. One way around this difficulty is to use the normal representation of a line:

$$x \cos\theta + y \sin\theta = \rho. \qquad (10.2\text{-}3)$$

Global Processing via the Hough Transform





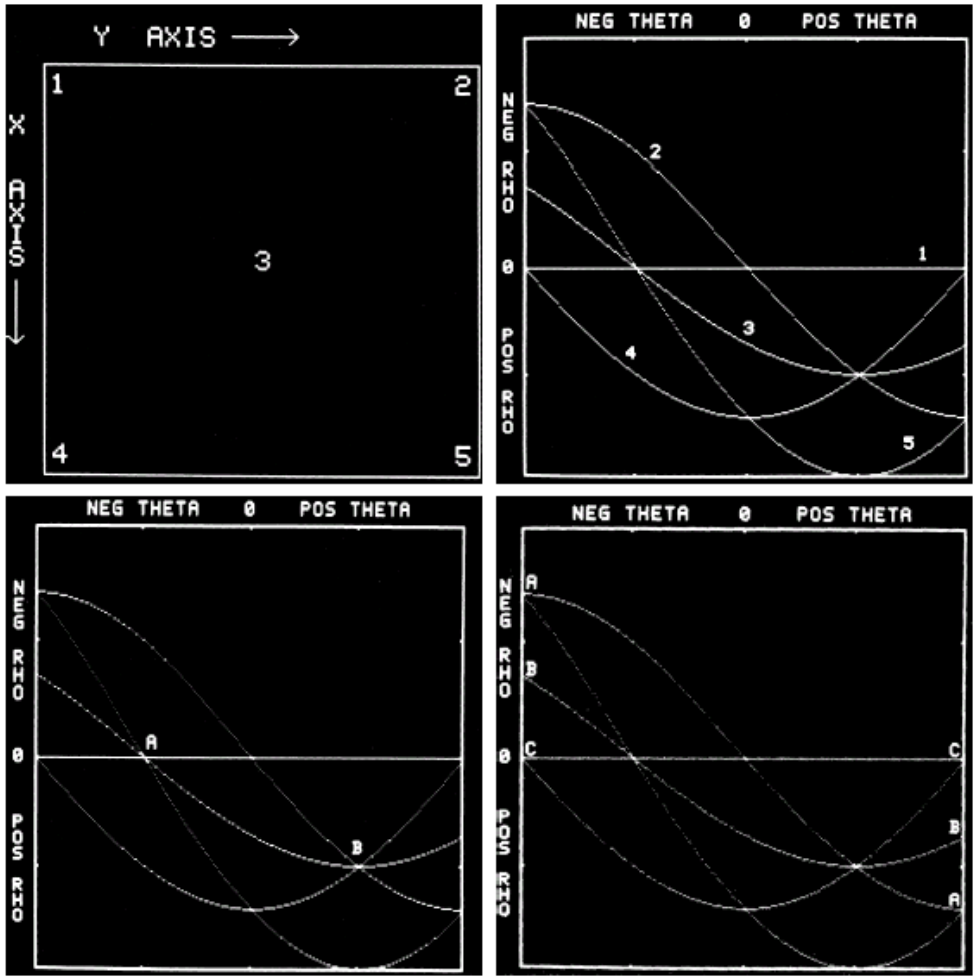
a b

FIGURE 10.19
(a) Normal representation of a line.
(b) Subdivision of the $\rho\theta$-plane into cells.

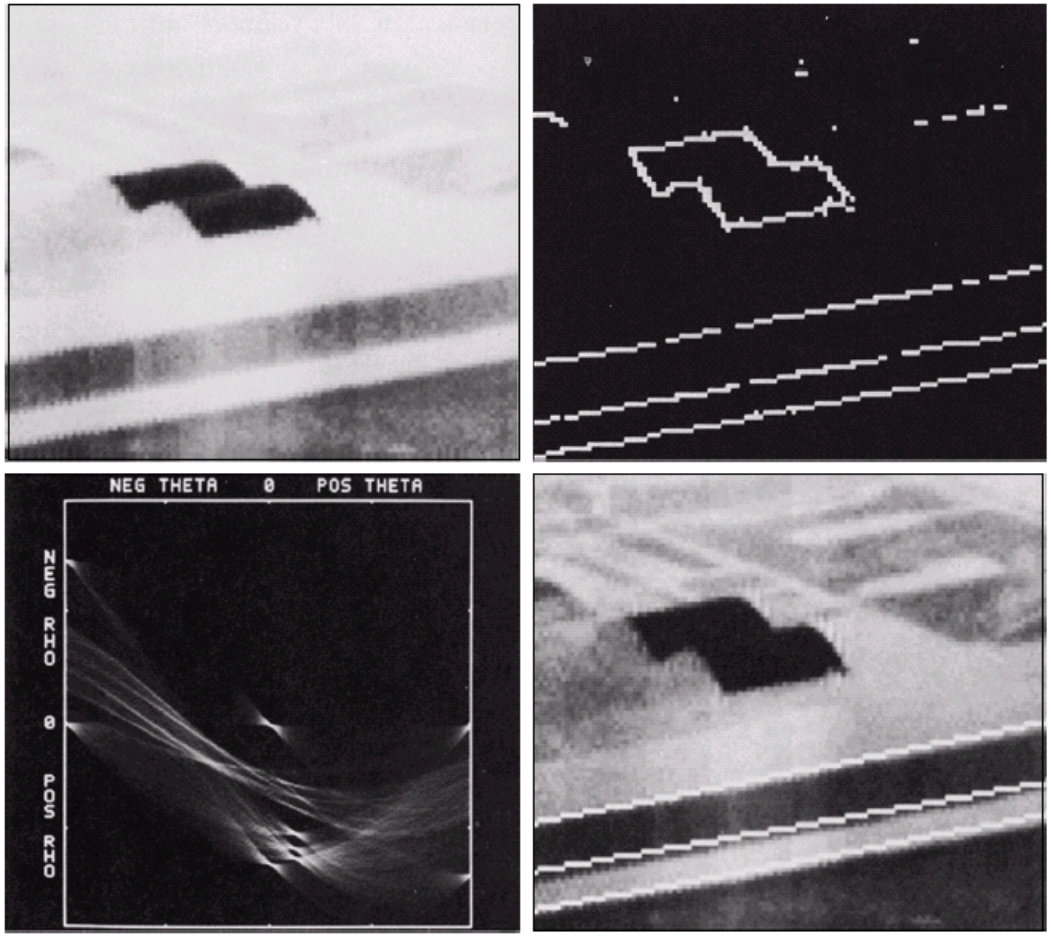
Global Processing via the Hough Transform

a b
c d

FIGURE 10.20
Illustration of the Hough transform. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)



Global Processing via the Hough Transform



a b
c d

FIGURE 10.21
(a) Infrared image.
(b) Thresholded gradient image.
(c) Hough transform.
(d) Linked pixels.
(Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

Hough lines in Matlab

MathWorks® Products Solutions Academia Support Community Events

MATLAB Examples

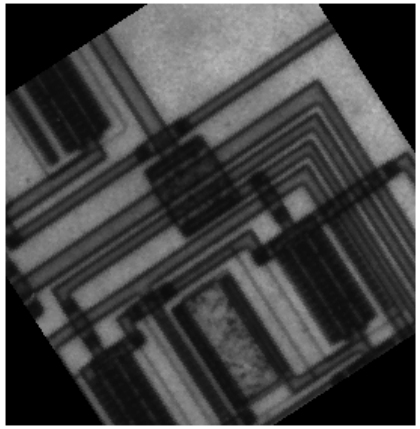
Examples Home › MATLAB Family › Image Processing and Computer Vision › Image Processing Toolbox › Image Segmentation and Analysis

Detect Lines in Images Using Hough

This example shows how to detect lines in an image using the Hough transform.

Read an image into the workspace and, to make this example more illustrative, rotate the image. Display the image.

```
I = imread('circuit.tif');
rotI = imrotate(I,33,'crop');
imshow(rotI)
```



https://la.mathworks.com/help/images/ref/houghlines.html

Global Processing via the Hough Transform

Although the focus so far has been on straight lines, the Hough transform is applicable to any function of the form $g(\mathbf{v},\mathbf{c}) = 0$, where $\mathbf{v}$ is a vector of coordinates and $\mathbf{c}$ is a vector of coefficients. For example, the points lying on the circle

$$(x-c_1)^2+(y-c_2)^2=c_3^2 \qquad (10.2\text{-}4)$$

can be detected by using the approach just discussed.

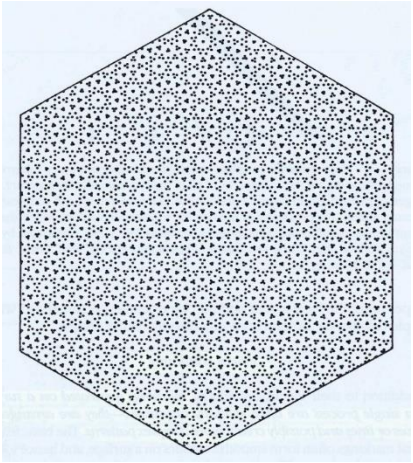
Ecuación de la circunferencia

Una circunferencia queda determinada por un centro $C = (a, b)$ y un radio r, por tanto, su ecuación queda determinada al imponer que la distancia de sus puntos, $P = (x, y)$, al centro sea constante, es decir, $\|P - C\| = r$ dando la siguiente ecuación:[6] [7]

$$(x - a)^2 + (y - b)^2 = r^2.$$

Su representación en un sistema de coordenadas viene dada por cada punto de la forma (x, y) que satisfacen la ecuación.

https://es.wikipedia.org/wiki/Circunferencia



Global Processing via the Hough Transform

The Hough transform can be used to determine the parameters of a circle when a number of points that fall on the perimeter are known. A circle with radius R and center (a, b) can be described with the parametric equations

$$\begin{aligned} x &= a + R\cos(\theta) \\ y &= b + R\sin(\theta) \end{aligned}$$

When the angle θ sweeps through the full 360 degree range the points (x, y) trace the perimeter of a circle.

If an image contains many points, some of which fall on perimeters of circles, then the job of the search program is to find parameter triplets (a, b, R) to describe each circle. The fact that the parameter space is 3D makes a direct implementation of the Hough technique more expensive in computer memory and time.

Global Processing via the Hough Transform

An approach based on the Hough transform is as follows:

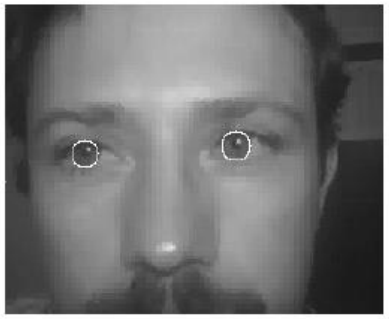
1. Compute the gradient of an image and threshold it to obtain a binary image.
2. Specify subdivisions in the pθ-plane
3. Examine the counts of the accumulator cells for high pixel concentrations.
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

   The concept of continuity in this case usually is based on computing the distance between disconnected pixels identified during traversal of the set pixels corresponding to a given accumulator cell.

Global Processing via the Hough Transform

Detección de movimientos oculares. Dr. Homero V. Ríos Figueroa.

Figura 9: Detección de iris. Aquí se empleó la TH con la variante que los centros potenciales se localizaron mediante la intersección de los vectores gradientes.



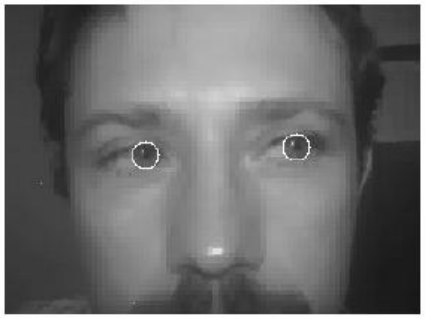


Figura 11: Detección de ojos mirando a la derecha.

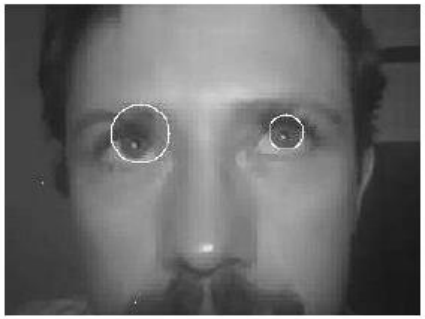


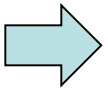
Figura 12: Detección de ojos mirando hacia arriba.

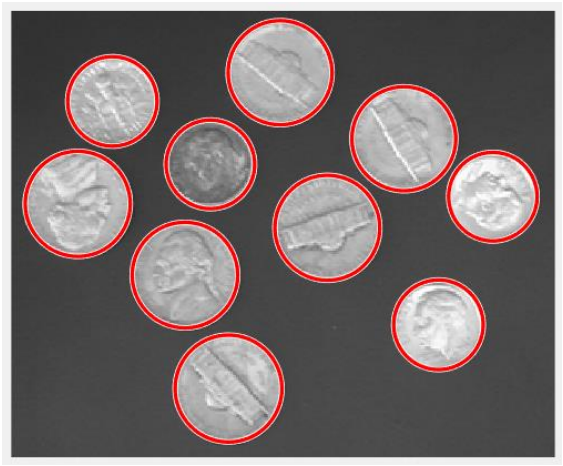
Homework 12

- Implement the Hough transform for circle detection.

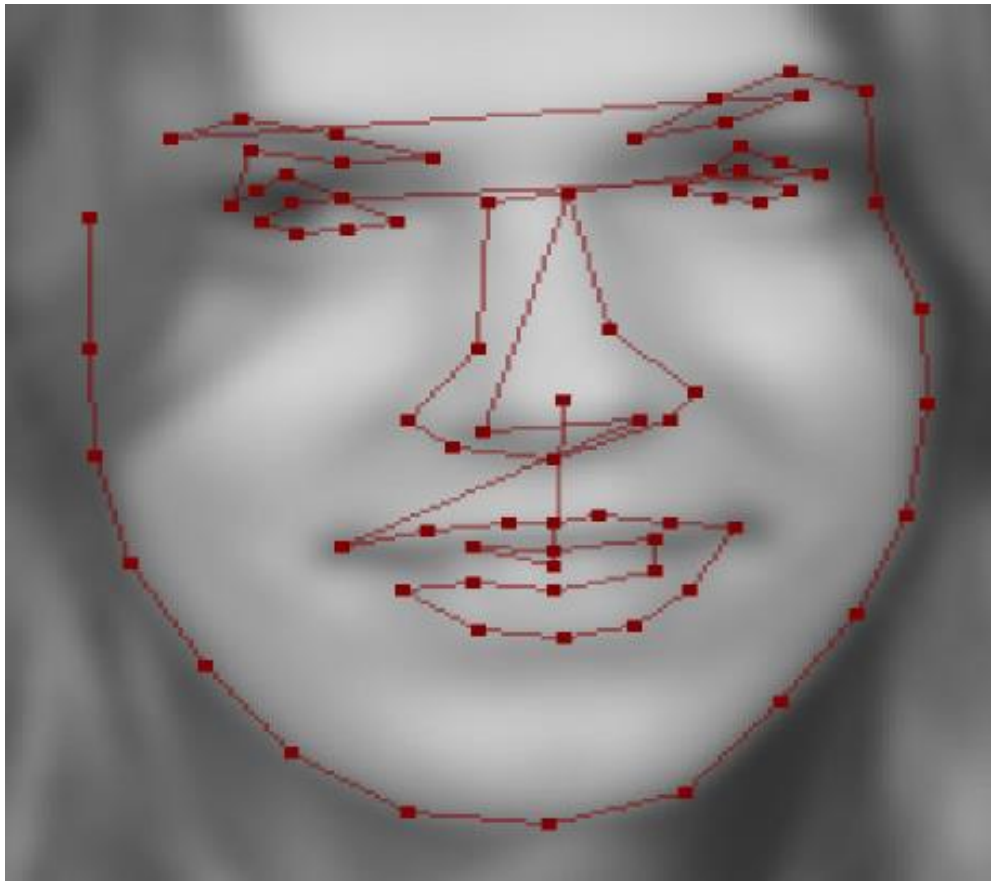
- Use your algorithm to segment the coins in the following image:

- I = imread('coins.png');





Active Shape Models



Consult PDF file

(Active Shape Models.ppt)

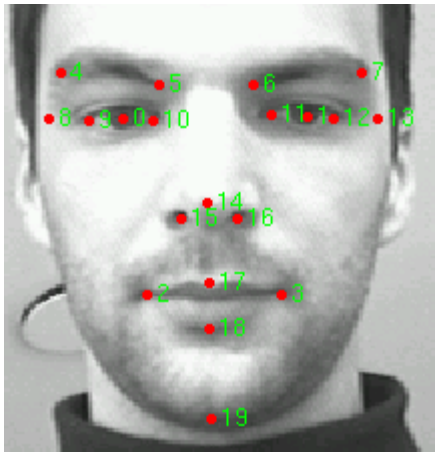
Active Shape Models

Tim Cootes: Software

I write software predominantly in C++ using the excellent VXL computer vision libraries, to which I and my colleagues regularly contribute.
You can download VXL from Sourceforge.
I've written a VXL library for constructing and manipulating 2D shape models: contrib/mul/msm
I have created a set of graphics classes and widgets to display VXL objects using Qt: UoMqVXL
This includes tools to annotate images with points, and display the modes of a shape model. See: qmsm/tools
I have also released source code for an Active Shape Model library, building on VXL.

Tools

- am_tools : A set of tools to build and play with Appearance Models and AAMs.
- qsnake_demo : Basic tool to play with snakes (active contour models).
- Brain Segmentation Tool written by Kola Babalola using 3D AAMs

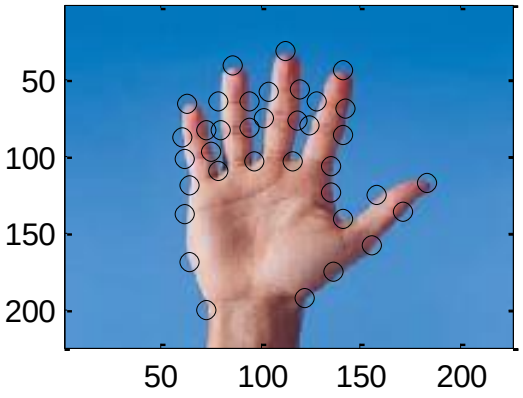


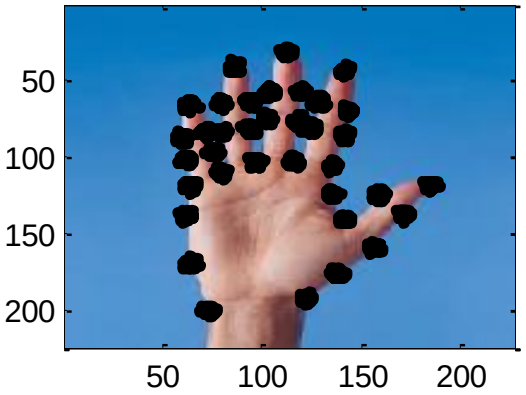
Datasets

You can find a range of useful annotated datasets on the FGNET Website.
Manual annotation of the BioID face images.
Manual annotation of the XM2VTS face images.
Manual annotation of the AR Face Database face images.
Images with annotation of a talking face.
A set of hand points.

http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/tfc_software.html

Matlab example





Matlab example

```
% Ejemplo de creación del modelo estadístico de forma usando  PCA
% y calculo de parametros de regresion lineal sobre CP de la base de datos.

% Vector de posiciones predefinidas.
% Secuencia de tuplas (x,y) por punto.
puntos=[73,199,64,168,62,137,64,118,61,101,60,… % Coordenadas
figure % Viendo los datos
subplot(2,2,1)
title('Base points')
img=imread('Mano.jpg');
imagesc(img)
hold on
for i=1:2:72
        plot(puntos(i),puntos(i+1),'ok')
end

aux=repmat(puntos,100,1); % Generación de eventos

noise=randn(100,72)*1.5; % Ruido
DB=aux+noise;
```

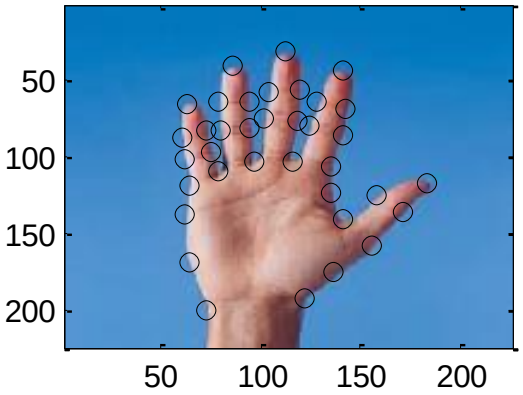
Matlab example

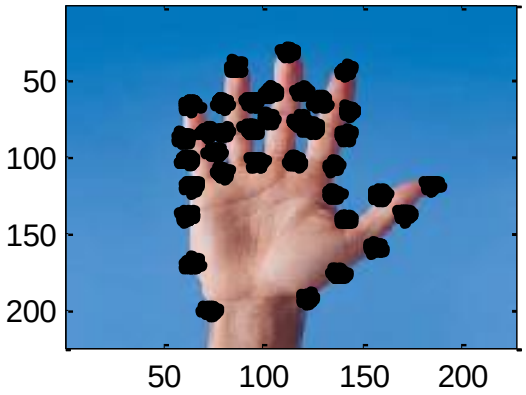
```
subplot(2,2,2) % Viendo los datos
title('Data')
imagesc(img)
hold on
for i=1:2:72
    for j=1:100
        plot(DB(j,i),DB(j,i+1),'.k')
    end
end

m=mean(DB')';
DBm=DB-repmat(m,1,72); % Haciendo la base de datos con media cero

% Calculando los componentes principales usando PCA

PcI=mypca(DBm',4)'; % Número de componentes requerido

PcIm=PcI-repmat(mean(PcI')',1,72);
```

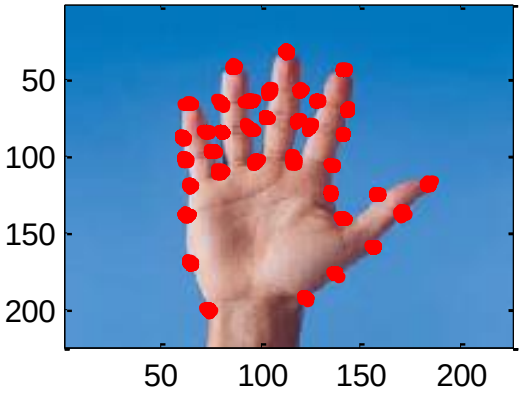
Matlab example

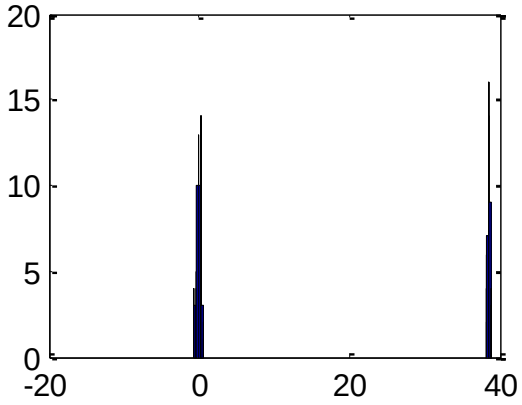
```
% Regresion lineal de los datos en los componentes

A=inv(PcIm*PcIm');
B=PcIm*DB';
alphas=A*B; % Parametros de regresion
models=alphas'*PcIm+repmat(m,1,72); % Reconstruyendo los datos

DB=DB+repmat(m,1,72);

% Comparando resultados.
subplot(2,2,3) % Viendo los datos
title('Models')
imagesc(img)
hold on
for i=1:2:72
    for j=1:100
        plot(models(j,i),models(j,i+1),'.r')
    end
end
```

Matlab example

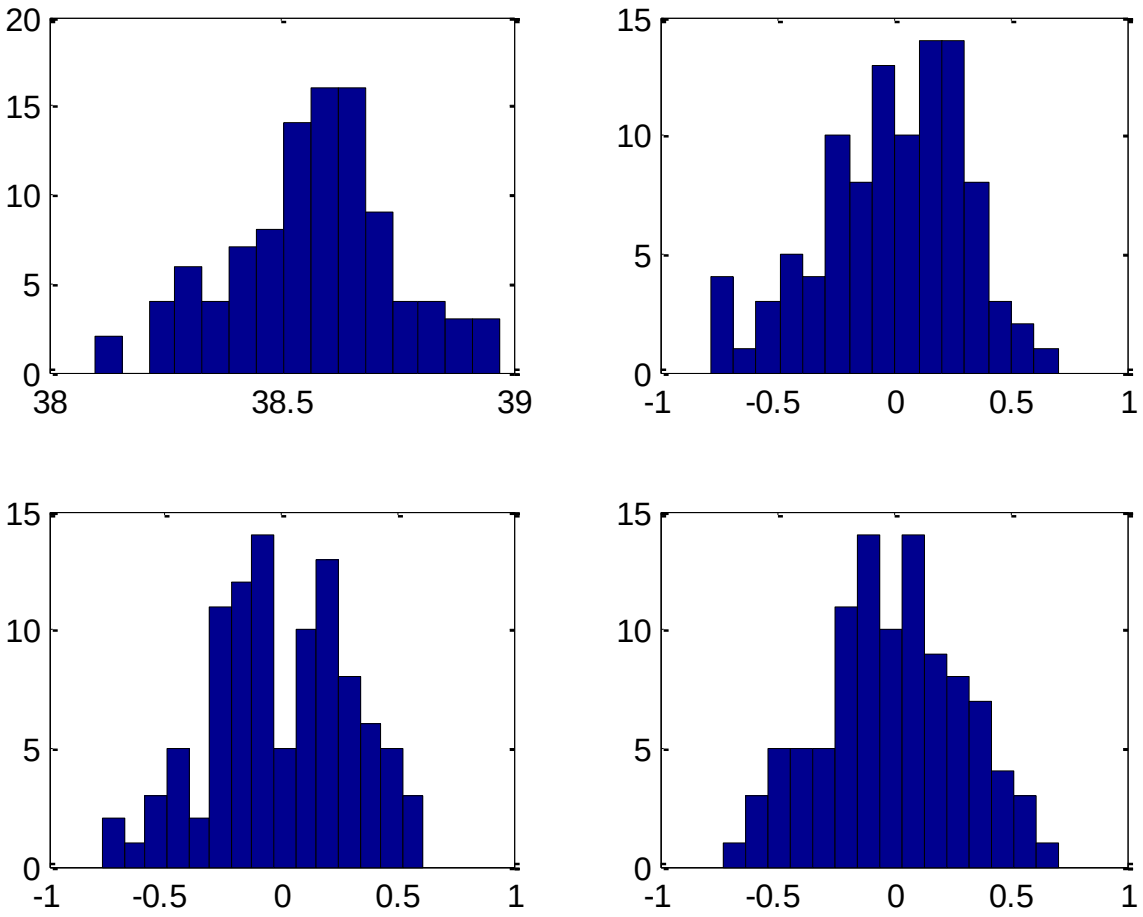








Matlab example



Suggested readings

- **Active Shape Models: Their Training and Application.** T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Computer Vision and Image Understanding. Vol. 61, No. 1, pp. 38-59, 1995.

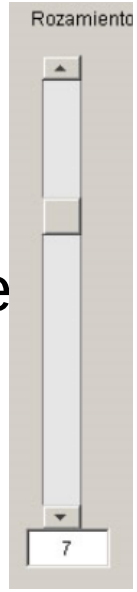
- **The Use of Active Shape Models for Locating Structures in Medical Images.** T.F. Cootes, A. Hill, C.J. Taylor and J. Haslam. Image and vision computing, Vol. 12, No. 6. pp.355-366. 1994

- **Improved edge detection for object segmentation in ultrasound images using Active Shape Models.** F. Arámbula Cosío, Héctor G. Acosta, Edgar Conde, Proc. SPIE 9287, 10th International Symposium on Medical Information Processing and Analysis.

Tarea 13

• Utilizar el mapa de distribución de puntos que registraron de la base de datos de caras.

• Aplicar PCA y crear su modelo estadístico.



• Crear una interfaz con botones deslizables donde se puedan manipular los parámetros de forma y se visualice su efecto.

• Los valores máximos y mínimos para cada parámetro deberán estar dados por los valores de los extremos de los histogramas respectivos.

• Encontrar los parámetros de pose y forma para ajustar un par de rostros (optimización).

Momentos geométricos

- ✓ **Los momentos son propiedades numéricas que se pueden obtener de una determinada imagen.**
- ✓ **Tienen en cuenta todos los píxeles de la imagen, no solo los bordes.**
- ✓ **Clasificación de los momentos:**
  - **Momentos Simples**
  - **Momentos Centrales**
  - **Momentos Centrales Normalizados**
  - **Invariantes**

Momentos Simples

- **Se emplean para obtener otros momentos, pero también dan información por sí mismos.**

- **Para una función** continua **f(x,y), el** momento de orden (p+q) **se define como:**

$$m_{pq} = \int\int x^p y^q f(x,y) dxdy$$

En discreto **obtenemos lo siguiente:**

$$M(p,q) = \sum_x \sum_y x^p y^q f(x,y)$$

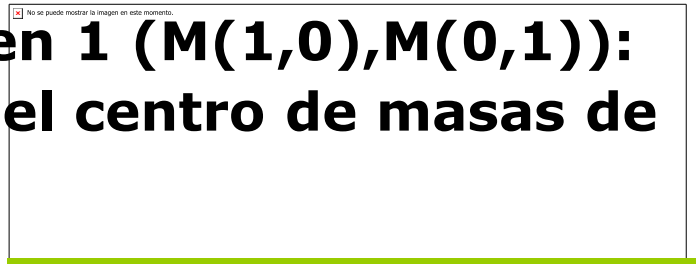
Momento de Orden 0

- **El momento simple de orden 0 representa el área de la figura en imágenes binarias y la superficie en imágenes en escala de grises. Es la suma de los valores de todos los píxeles.**

$$M(0,0) = \sum_x \sum_y f(x,y)$$

Momentos Orden 1. [(p+q) = 1]

- **Momentos simples de Orden 1 (M(1,0),M(0,1)):**
  - **Se emplean para hallar el centro de masas de una figura.**



$$M(1,0) = \sum_x \sum_y x f(x,y)$$

$$M(0,1) = \sum_x \sum_y y f(x,y)$$

- **Momentos Centrales de Orden 1 (MC(1,0),MC(0,1)):**
  - **Estos momentos son 0 por definición.**

$$U(1,0) = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^0 f(x,y)$$

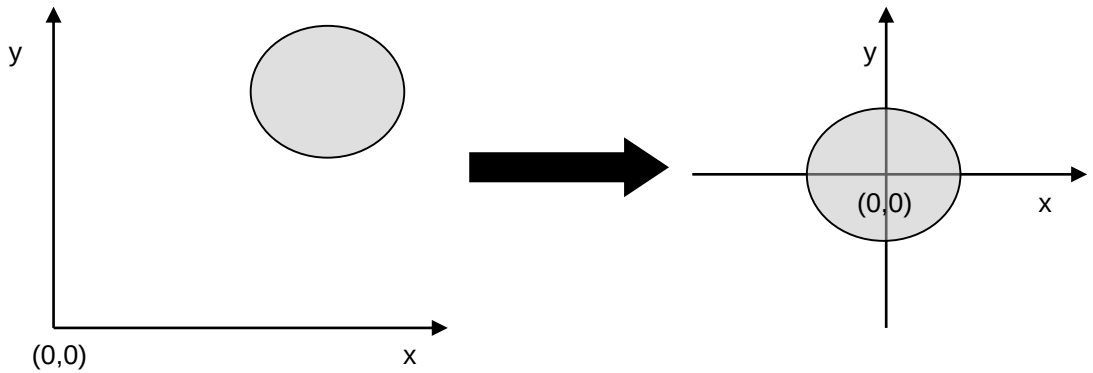
$$= M(1,0) - \frac{M(1,0)}{M(0,0)} M(0,0)$$

$$U(0,1) = \sum_x \sum_y (x-\bar{x})^0 (y-\bar{y})^1 f(x,y)$$

$$= M(0,1) - \frac{M(0,1)}{M(0,0)} M(0,0)$$

Momentos Centrales

- Permiten reconocer figuras dentro de una imagen independientemente de su posición.



Continuo

$$\mu_{pq} = \int\int (x-X)^p (y-Y)^q f(x,y)$$

Discreto

$$MC_{pq} = \sum\sum (x-X)^p (y-Y)^q f(x,y)$$

Centroide

- Centro de masas de la figura.
- Viene representado por los momentos de orden 0 y 1

$$X = M(1,0) / M(0,0) \qquad Y = M(0,1) / M(0,0)$$

- El área de la figura que queda a la derecha e izquierda del punto X es la misma.
- Igual para el área que queda por encima y por debajo del punto Y

Momentos Centrales Normalizados

- **Permiten reconocer figuras dentro de una imagen independientemente de su tamaño.**
- **Se normalizan los** momentos centrales **con el** momento de orden 0**, obteniendo así figuras independientes de la escala.**

$$MCN(p,q) = MC(p,q) / MC^{\beta}(0,0)$$

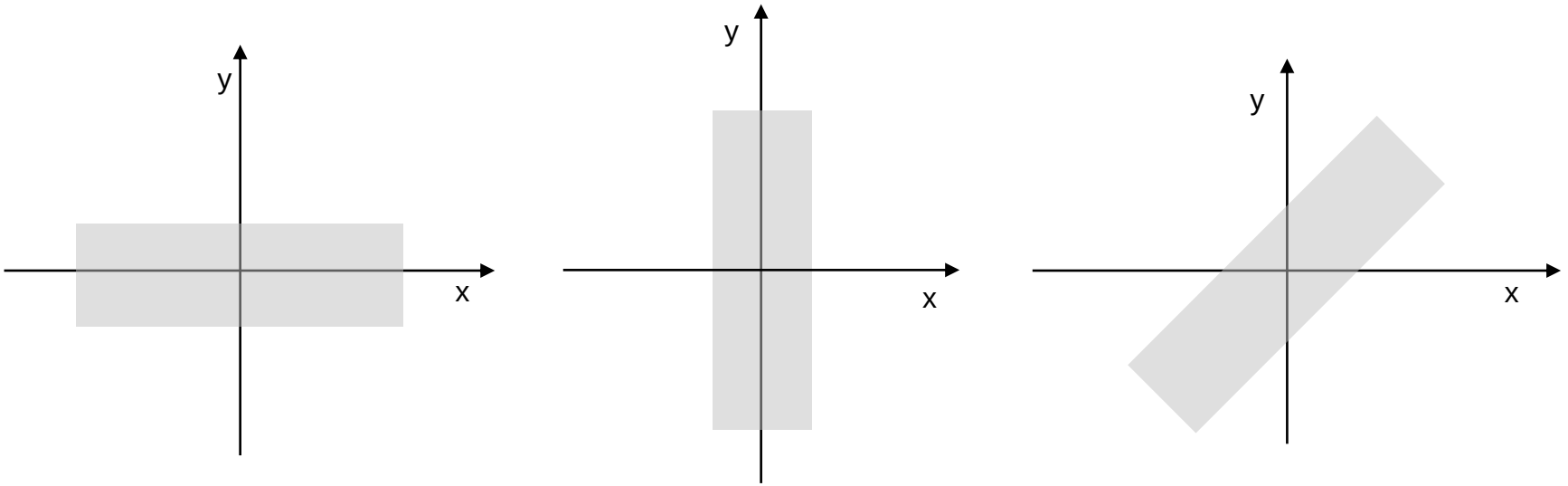
$$\beta = \frac{p+q}{2} + 1$$

Momentos Orden 1. [(p+q) = 1]

- **Momentos Centrales Normalizados de Orden 1 (MCN(1,0),MCN(0,1)):**
  - **Estos momentos son 0 por definición.**

Momentos Orden 2. [(p+q) = 2]

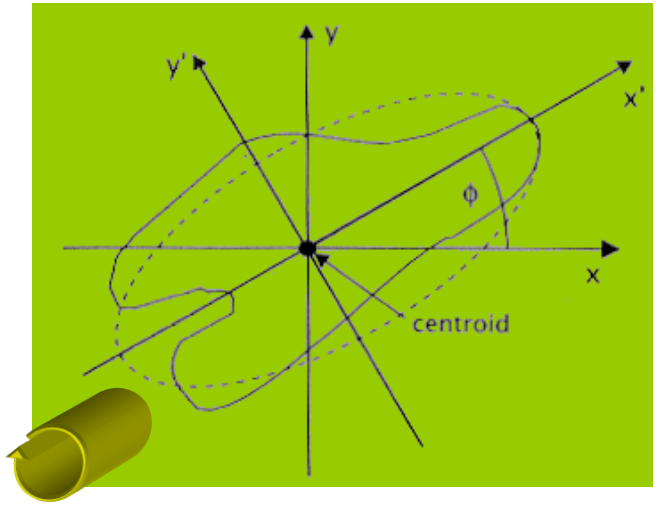
- **Son importantes para el calculo de los momentos centrales.**
- **La densidad de la figura se multiplica por distancias al cuadrado desde el centro de masas o centroide (Inercia).**



$$U(p,q) = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

Angulo rotación

- **La** orientación **o** ángulo de rotación **de la figura se define como el** ángulo **entre el eje de abscisas y el eje alrededor del cual la figura puede rotar con** mínima inercia**.**



$$\phi = \frac{1}{2}\arctan\frac{2U(1,1)}{U(2,0)-U(0,2)}$$

Momentos Orden 3. [(p+q) = 3]

- **Sirven para calcular los momentos invariantes.**

$$U(2,1) = M(2,1) - 2\bar{x}M(1,1) - \bar{y}M(2,0) + 2\bar{x}^2 M(0,1)$$

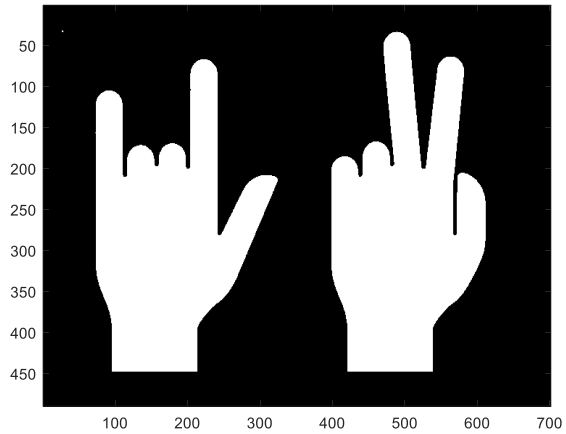
$$U(1,2) = M(1,2) - 2\bar{y}M(1,1) - \bar{x}M(0,2) + 2\bar{y}^2 M(1,0)$$

$$U(3,0) = M(3,0) - 3\bar{x}M(2,0) + 2\bar{x}^2 M(1,0)$$

$$U(0,3) = M(0,3) - 3\bar{y}M(0,2) + 2\bar{y}^2 M(0,1)$$

Moments in Matlab

```
I = imread('AB.jpg');
I=rgb2gray(I);

bw = I < 180;

imagesc(bw)
colormap(gray)
```



Area	Centroid		MajorAxisLength	MinorAxisLength
4	28.5	33.5	2.3094	2.3094
53301	172.89	279.94	353.79	232.28
55409	500.88	270.25	392.91	202.97

```
stats = regionprops('table',bw,'Area','Centroid',...
    'MajorAxisLength','MinorAxisLength')
```

Momentos Invariantes

- **A partir de los momentos centrales normalizados de orden 2 y 3 se obtienen los siete momentos invariantes de Hu.**
- **Estos conjunto de momentos es** invariante **a la** traslación **y** cambio de escala **de una figura.**

Momentos Invariantes de Hu

- A partir de los momentos centrales normalizados, Hu derivó un conjunto de 7 invariantes a rotaciones, traslaciones y cambios de escala, los cuales son:

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

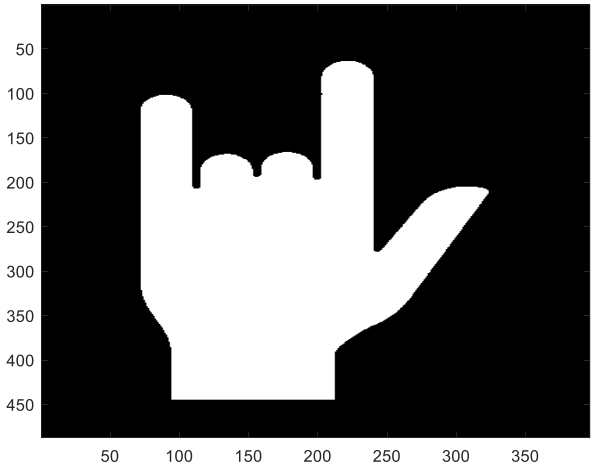
$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Hu invariant moments in Matlab

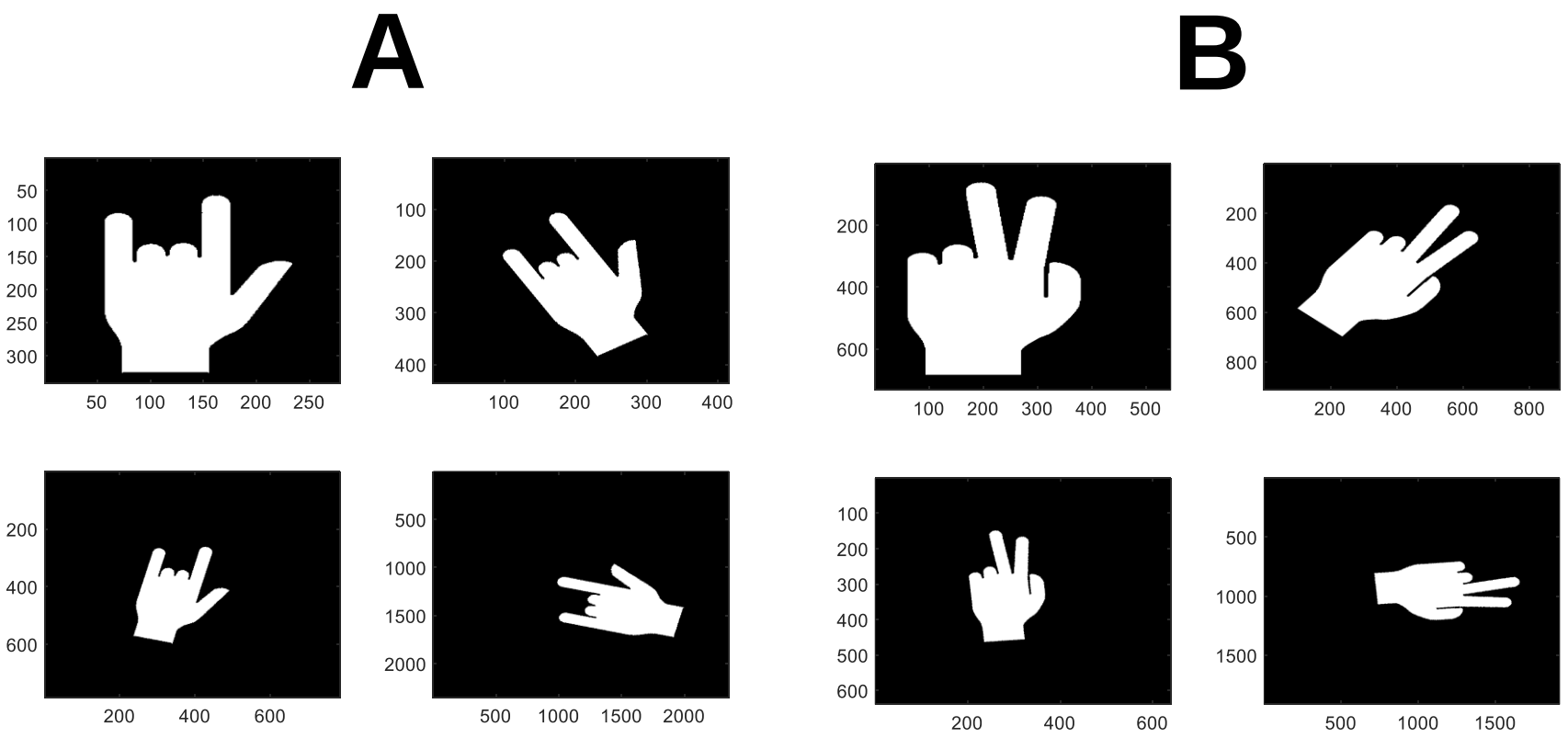
```
I = imread('A.jpg');
I=rgb2gray(I);

bw1 = I < 180;
bw1 = imtranslate(bw1,[10 20],'FillValues',0);
bw1 = imresize(bw1,0.7);
bw2 = imrotate(bw1,31);
bw3 = imresize(bw2,1.3);
bw3 = imtranslate(bw3,[-10 20],'FillValues',0);
bw3 = imrotate(bw3,-45);
bw4 = imresize(bw3,3);
bw4 = imrotate(bw4,90);
bw4 = imtranslate(bw4,[200 100],'FillValues',0);
```

```
phi(1,:) = invmoments(bw1);
phi(2,:) = invmoments(bw2);
phi(3,:) = invmoments(bw3);
phi(4,:) = invmoments(bw4);
```



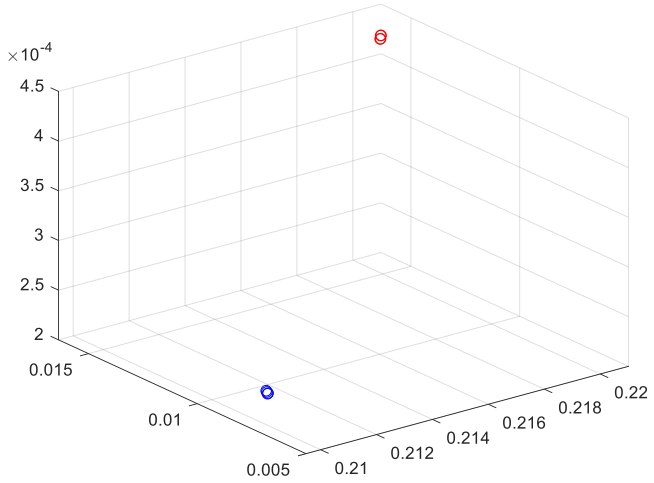
Hu_moments

Hu invariant moments in Matlab



Hu invariant moments in Matlab

```
phi1 =

    0.2095    0.0069    0.0002    0.0002   -0.0000    0.0000   -0.0000
    0.2095    0.0068    0.0002    0.0001   -0.0000    0.0000   -0.0000
    0.2095    0.0068    0.0002    0.0002   -0.0000    0.0000   -0.0000
    0.2096    0.0068    0.0002    0.0002   -0.0000    0.0000   -0.0000

>> phi2

phi2 =

    0.2210    0.0164    0.0004    0.0004    0.0000    0.0000   -0.0000
    0.2210    0.0164    0.0004    0.0004    0.0000    0.0000   -0.0000
    0.2210    0.0164    0.0004    0.0004    0.0000    0.0000   -0.0000
    0.2210    0.0164    0.0004    0.0004    0.0000    0.0000   -0.0000
```



Suggested readings

- **Applied Machine Learning to Identify Alzheimer's Disease through the Analysis of Magnetic Resonance Imaging**. E. M. Novoa-Del-Toro, H. G. Acosta-Mesa, J. Fernández-Ruiz and N. Cruz-Ramírez, 2015 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, 2015, pp. 577-582. doi: 10.1109/CSCI.2015.143