Taif University
College of Computers and Information Technology

Image Processing & Computer Visi§on

(AI:501837-3) - (DS:501838-3)

Programs:
Master of AI
Master of Data Science

Department:
Computer Science

Year:
Spring 2025

Lecture 5:
Image Segmentation

Prepared by: Maha A. Thafar

Lecture Outline

Chapter 10:

Image Segmentation

Part-1: Line/Boundary Detection

- Edge Linking and Boundary Detection

Part-2: Image Segmentation using:

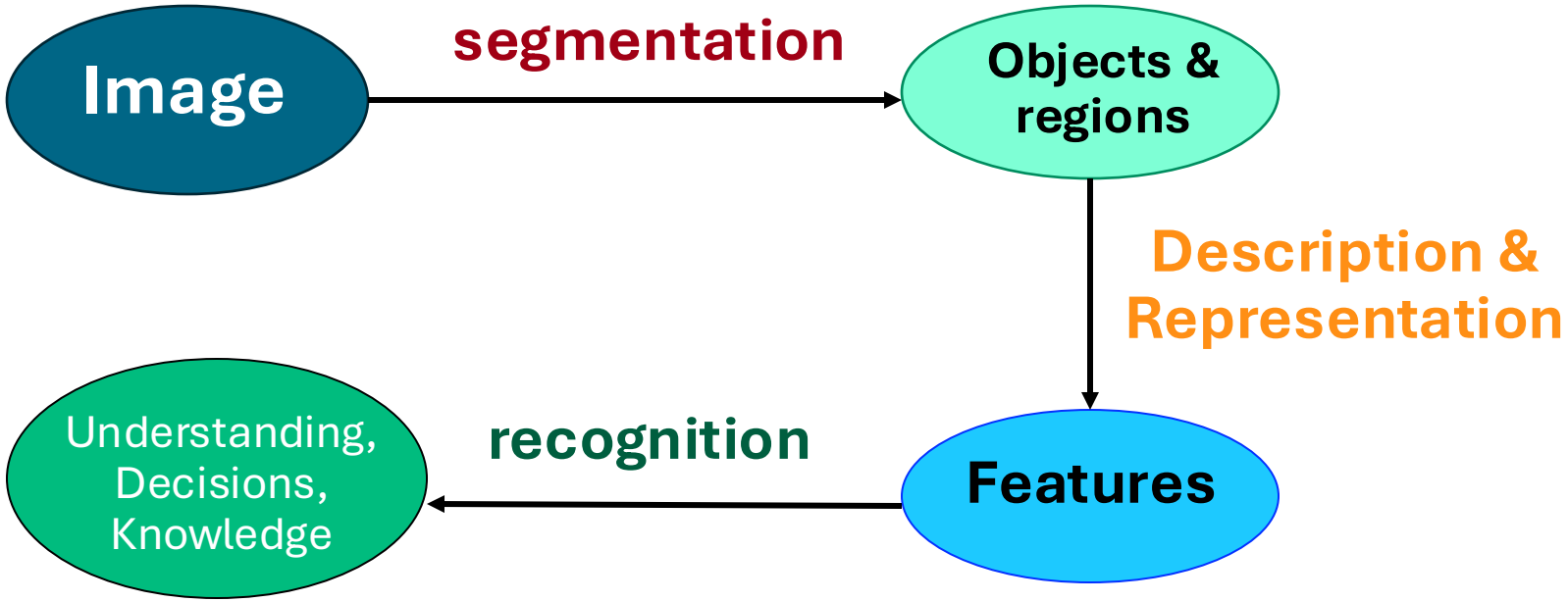
- Thresholding
- Region-Based Segmentation

Part-3: What is Next?

A Gateway to ML Clustering

- K-means Clustering

High Level Image Processing (HLIP)



- **Image analysis:**
  - Extracting information from an image.
- **First step** → To segment the image →

- Dividing image into its constituent regions or objects

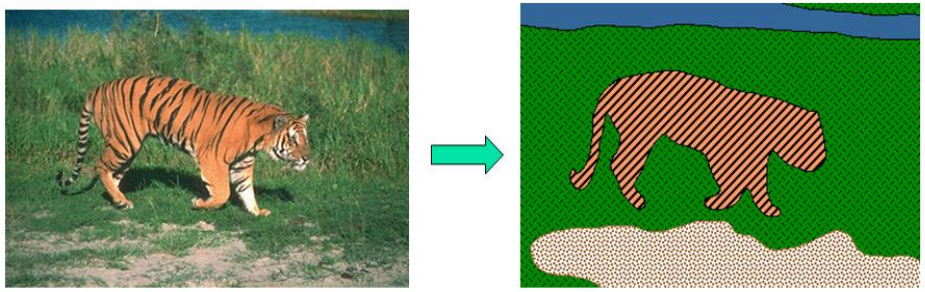


Image Segmentation

★ Based on two properties of gray-level image values

❑ **Discontinuity**

- **3 basic types of gray-level discontinuities: points, lines, and edges**. (*refer to lecture 4: edge detection*)
- Edge detection

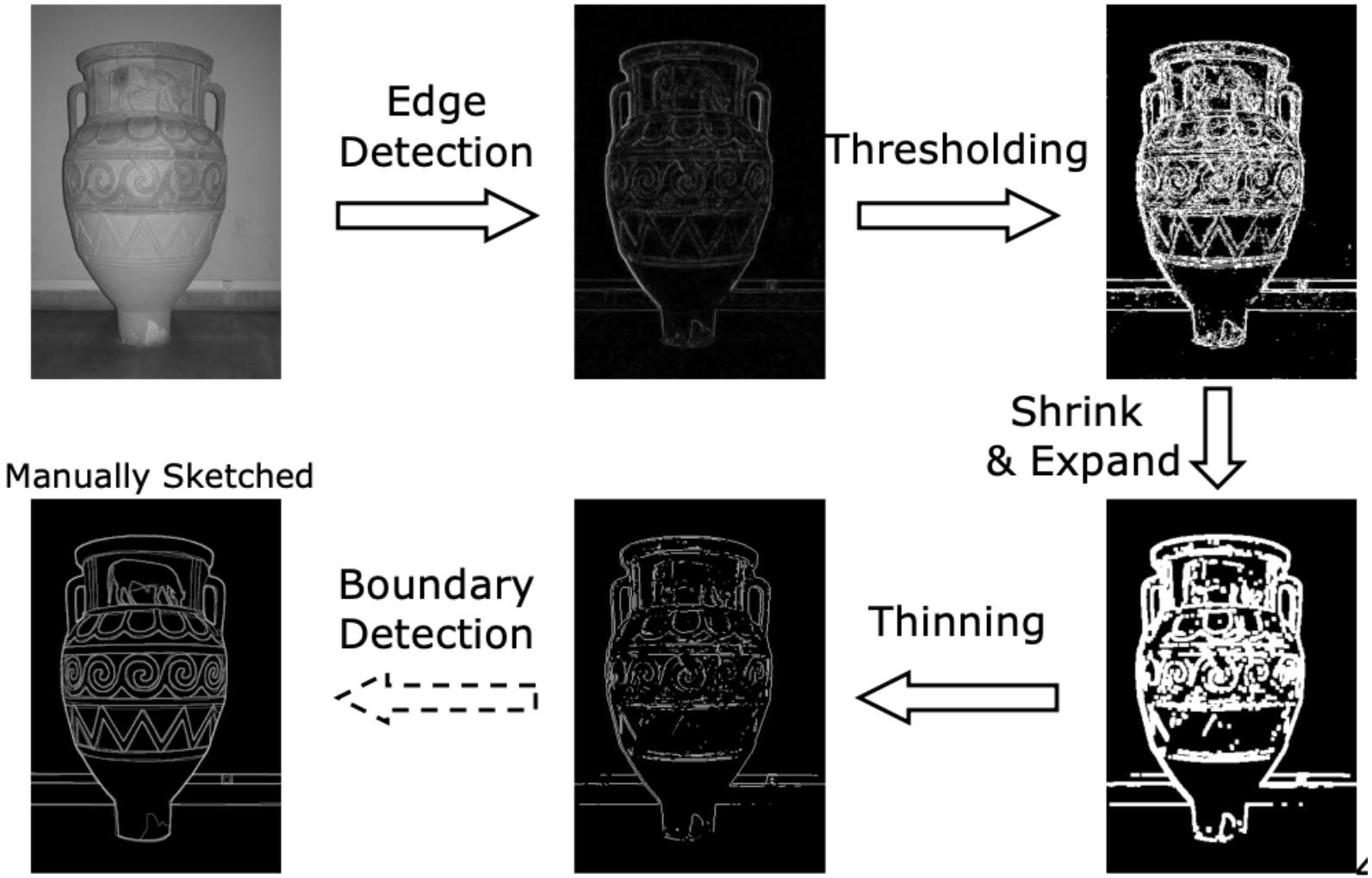
❑ **Similarity**

- Thresholding
- Region growing/ splitting/ merging

- **Image segmentation** is typically used to locate objects and boundaries (lines, curves, etc.) in images.
- Image segmentation aims to change an image's representation into something more meaningful and easier to analyze.

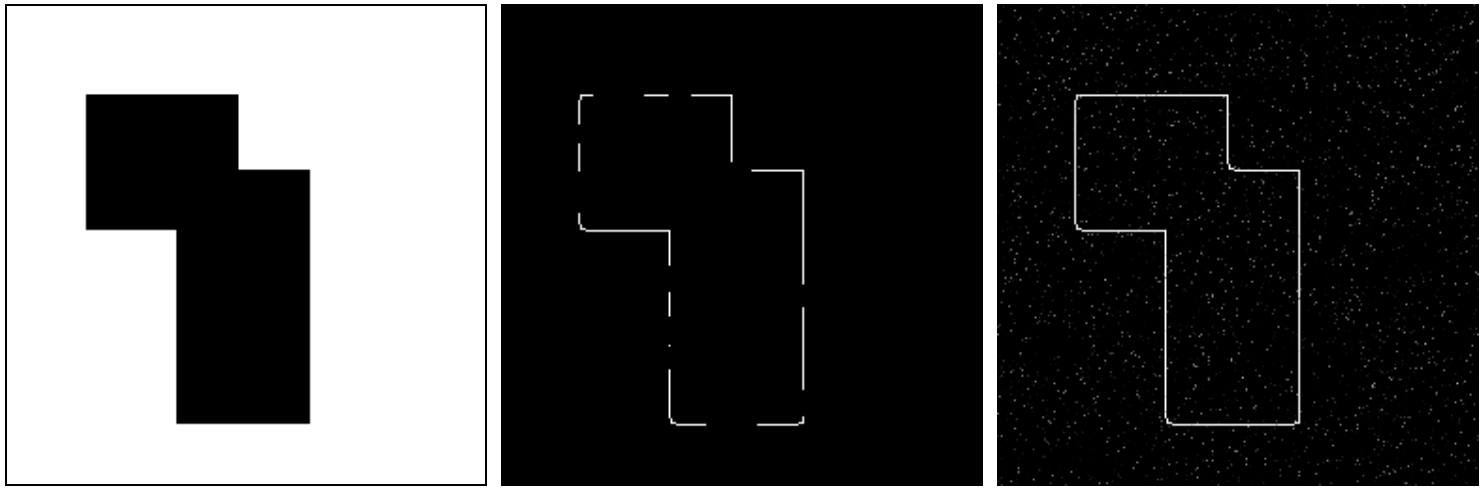
Preprocessing Edge Images

Segmentation by edge detecting method



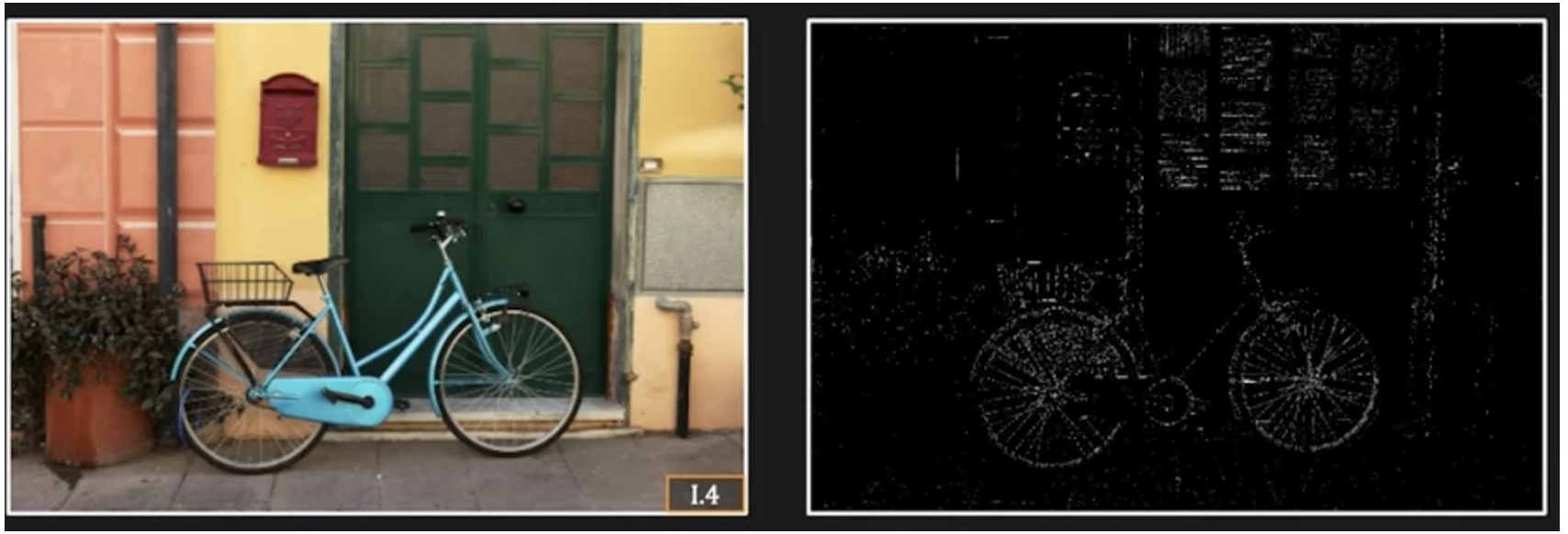
Edge Linking

- How to deal with gaps in edges?
- How to deal with noise in edges?
- Linking points by determining whether they lie on a curve of a specified shape



Edge detection is typically followed by **linking algorithms** designed to **assemble** edge pixels into meaningful edges and/or region boundaries.

Difficulties of some of the Boundary Detection Approaches



Issues:

- Extraneous Data: Which points to fit to?
- Incomplete Data: Only part of the model is visible.
- Noise

Solution: Hough Transform

✶ Can tolerate noise and gaps in edge image

✶ Look for solutions in a **parameter space**

✶ **Classical Hough transform**

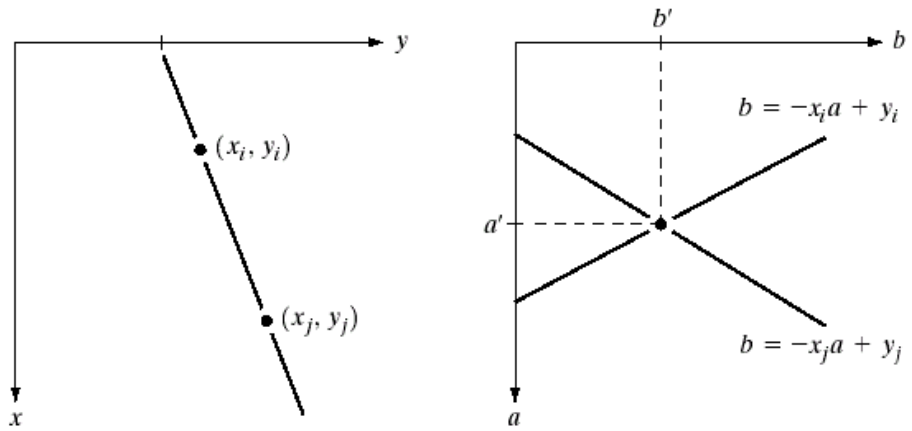
- Detect simple shapes → (Line or circle detection)

✶ **Generalized Hough transform**

- Detect complicate shapes

Edge Linking and Boundary Detection: Global Processing

- **Hough transform** is a feature extraction method for detecting simple shapes, such as circles, lines, etc., in an image.
- A "simple" shape is one that can be represented by only a few **parameters**.
- For example,
  - A **line** can be represented by **two parameters:** *the **xy-plane** vs. **ab-plane*** (slope and intercept in parameter space)
  - A **circle** has **three parameters** — the coordinates of the center and the radius (x, y, r).



a b

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

$$y_i = ax_i + b$$

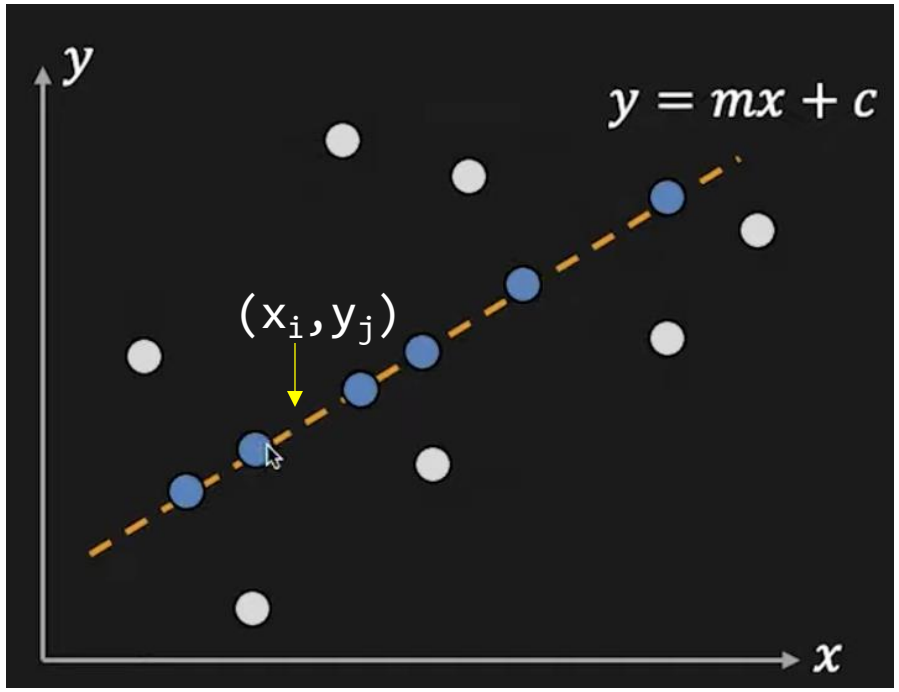
Hough Transform: Line Detection

Given a set of edge points, we want to find as many lines that connect these points in image space.

Given: Edge Points (x_i, y_j)

Goal: Detect line $y=mx+c$

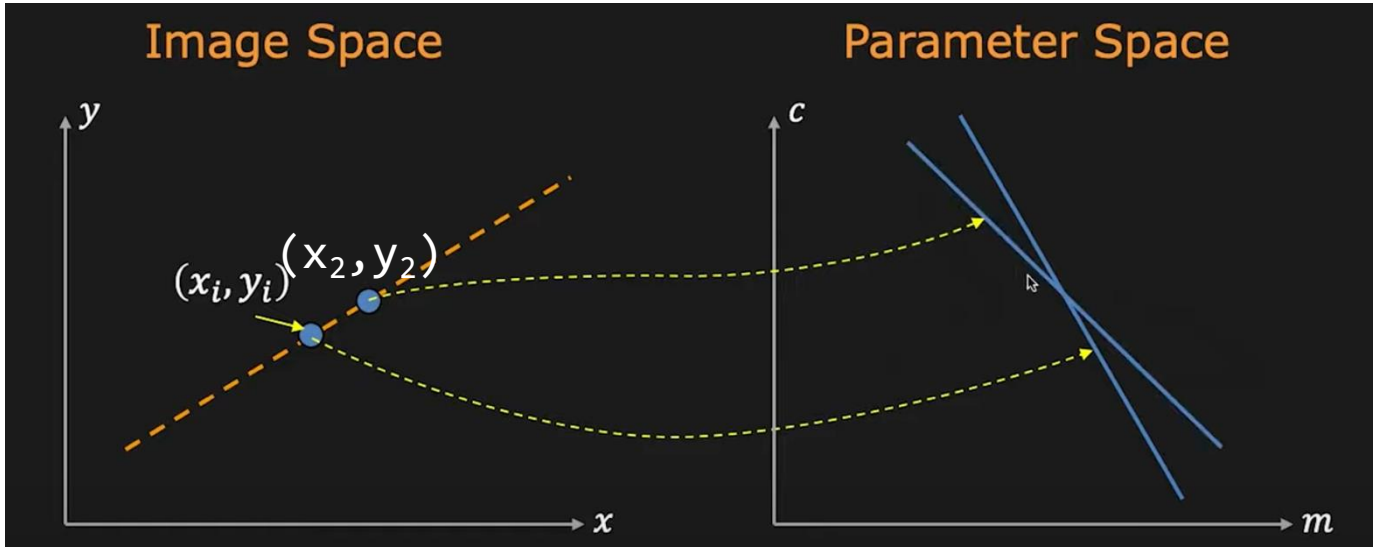
- **m** = gradient or slop of the line (rise/run)
- **c** = y -intercept (the point where the line crosses the y-axis).
- **rise** = the vertical change between two points, **run** = the horizontal change.



consider: one point on the line: (x_i, y_j)

$$y_i = mx_i + c \quad \text{rewrite} \quad c = -mx_i + y_i$$

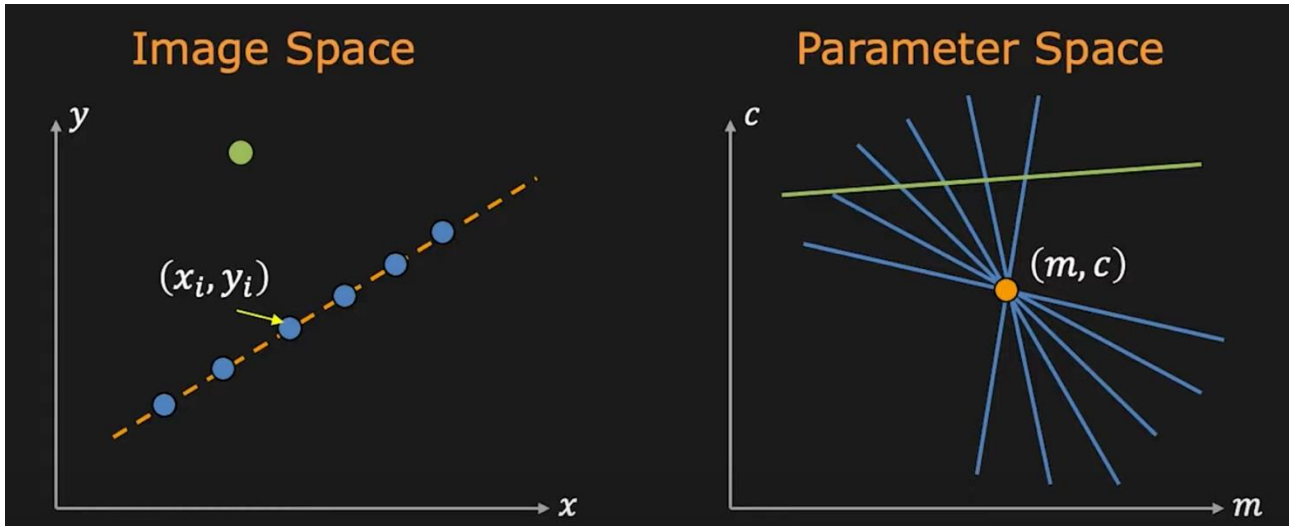
Hough Transform Concept



$y_i = mx_i + b$ $\qquad$ $c = -mx_i + y_i$

- A **line** (shown as dotted) in image space and one point (x_i, y_i) that lies exactly on the line.
- This point corresponds to the straight-line $c = -mx_i + y_i$ in parameter space.
- This line in parameter space corresponds to all the lines that pass through the point (xi, yi) in image space.
- If we take another point (x_2, y_2) in image space, it will give us another line in parameter space.
- **Note:** the two lines in parameter space intersect at a single point (m, c) corresponding to the slope and intercept of the dotted line that passes through the two points in image space.

Hough Transform Concept



$y_i = mx_i + b$ $\qquad$ $c = -mx_i + y_i$

Point ⟷ Line

Line ⟷ Point

- All points on a line in image space intersect at a common point in parameter space. This common point (m, b) represents the line in image space.

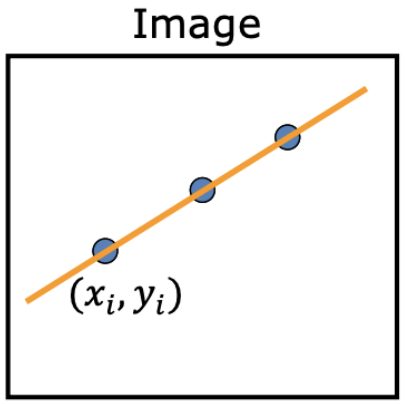
- If we take a point that does not lie on the dotted line in image space, it will also create a line in parameter space. However, that line will not pass through (m, c) because the image point does not lie on the straight line with parameters (m, c).

Line Detection Algorithm by Hough Transform

1) Quantize Parameter Space (m, c)

2) Create **Accumulator Array** $A(m, c)$

we define a 2D array of memory, called the **accumulator array**, to represent the discrete parameter space. The accumulator array is initialized with zeros



3) Set $A(m, c) = 0 \quad \forall m, c$

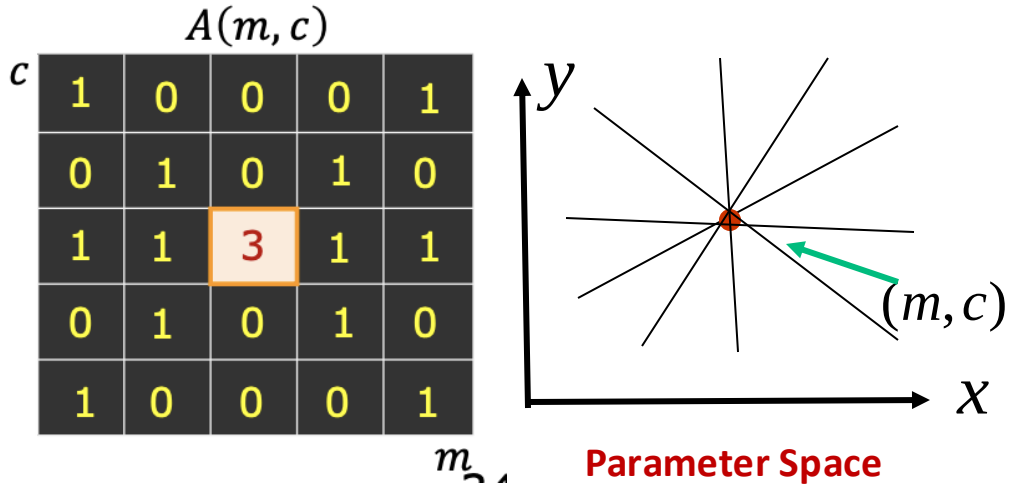
4) For each image edge (x_i, y_i) increment:

$$A(m, c) = A(m, c) + 1$$

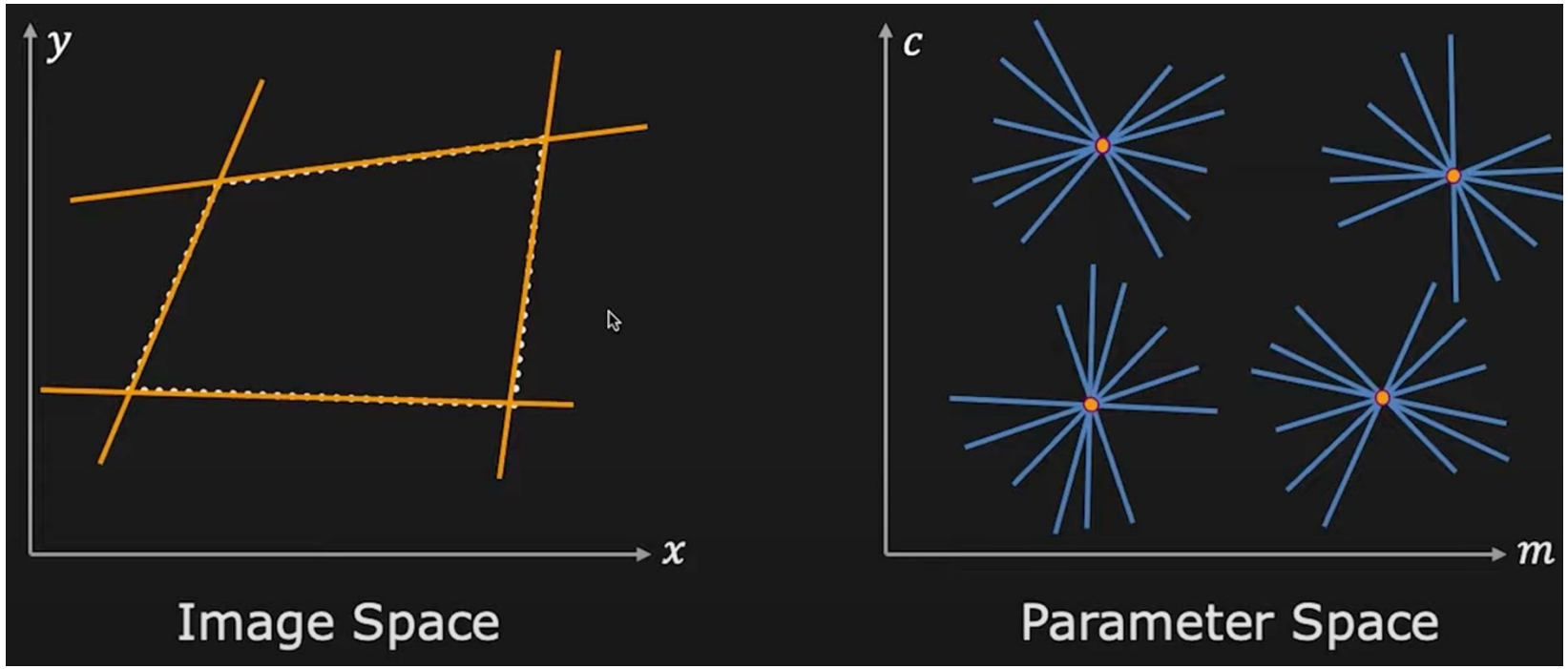
If (m, c) lies on the line:

$$c = -x_i m + y_i$$

5) Find local maxima in $A(m, c)$



Multiple Line Detection



- **Example**: we have 4-line segments, which results in 4 intersections in parameter space.
- **Note**: Hough transform allows to find these 4 lines without giving care to which point in image space belongs to which line. We simply repeat our voting process for each point in image space and if a strong maximum emerges in parameter space, we know there are a significant number of edges in the image that lie on a line. This demonstrates the power of the Hough transform.

Better Parametrization

Issue: slope of the line :

$$-\infty \le m \le \infty$$

- Large Accumulator
- More memory and computations
- slope, m, is undefined when the line is vertical (division by 0!).

Solution: (Finite Accumulator Array Size)

Line equation:

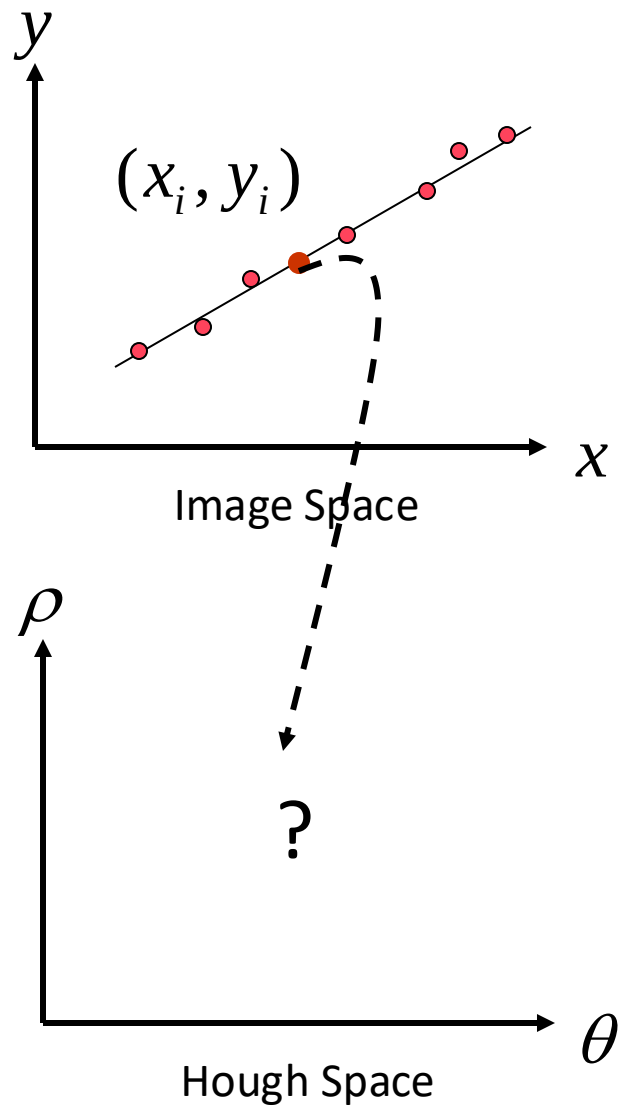
$$\rho = -x\cos\theta + y\sin\theta$$

Here **Orientation (angle)** is finite $0 \le \theta \le 2\pi$

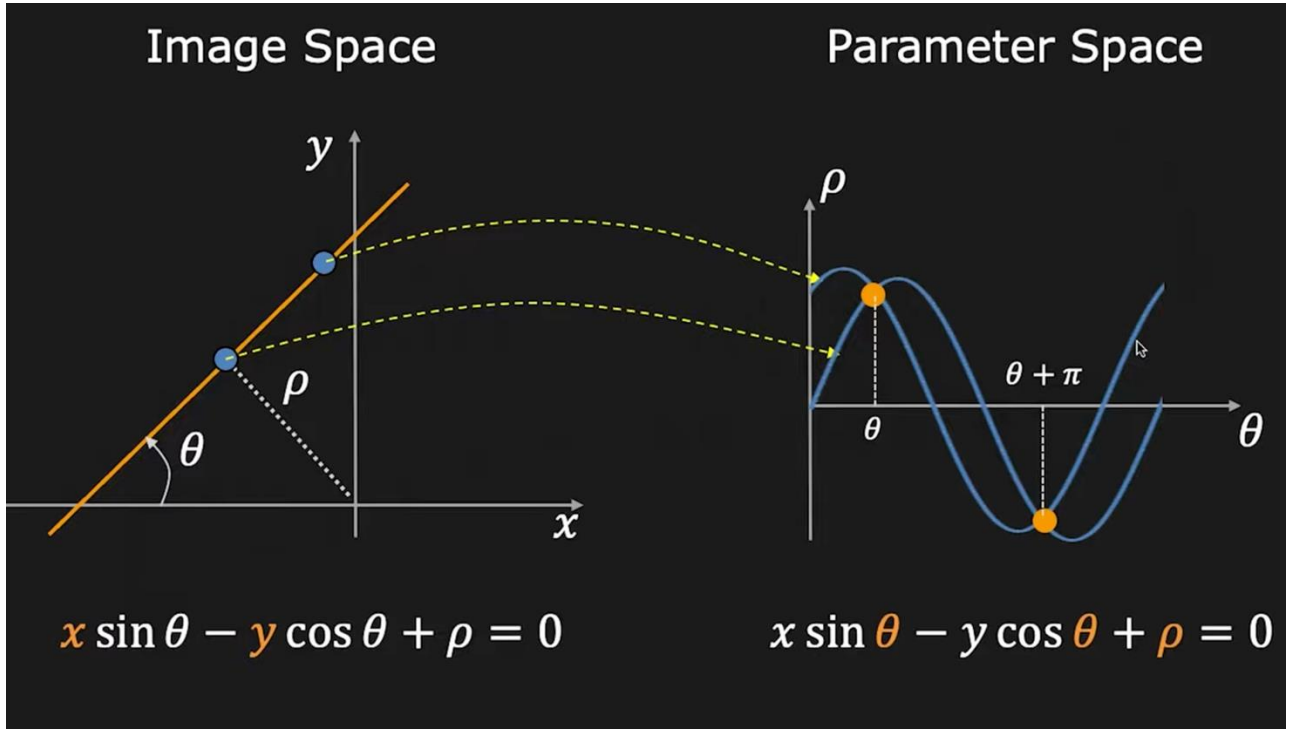
Distance (length) p is finite $0 \le \rho \le \rho_{\max}$

Given points (x_i, y_i) **find** (ρ, θ)

Hough Space Sinusoid



Angle-distance parameter space : (Hough space)

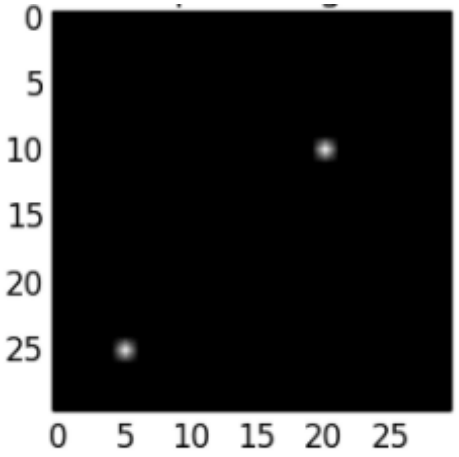


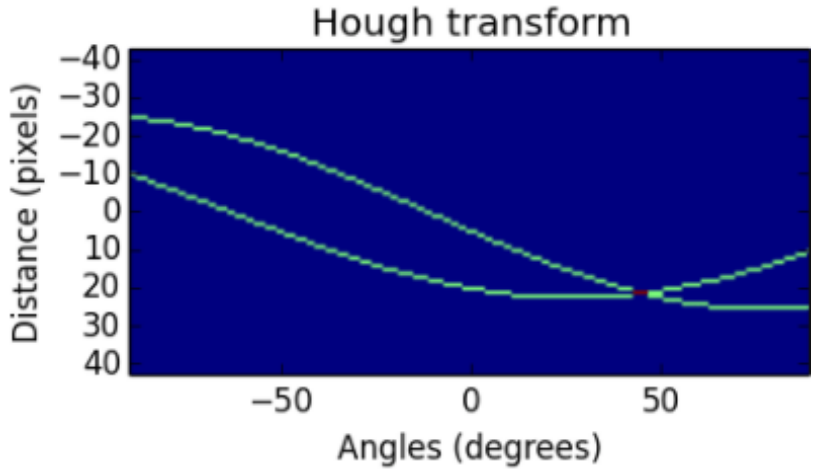
- Instead of representing the Hough Space with the slope *m* and intercept *c*, it is now represented with ρ and θ where the horizontal axis are for the θ values, and the vertical axis are for the ρ values.
- The mapping of edge points onto the Hough Space works in a similar manner except that an edge point (x_i, y_i) now generates a cosine curve in the Hough Space instead of a straight line.
- This normal representation of a line eliminates the issue of unbounded value of *m* that arises when dealing with vertical lines.

Hough Transform Example

- We've created an input binary image of size 30x30 pixels with points at (5,25) and (20,10)
- The image is transformed to the Hough space by calculating ρ with a point at each angle from −90◦ to 90◦ (negative angles are anti-clockwise starting horizontally from the origin and positive angles are clockwise).
- The points in Hough space make a sinusoidal curve.

Input Image





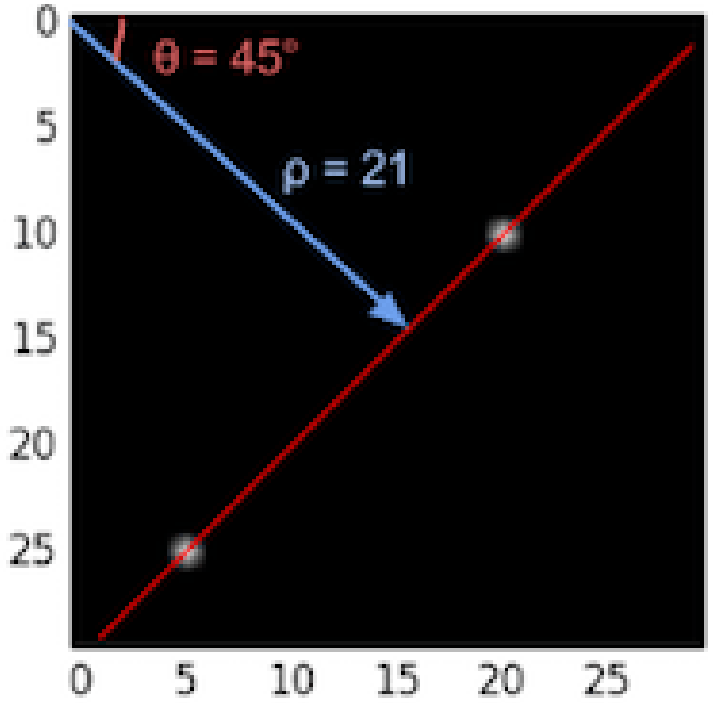
The value of ρ at various angles for the 2 edge points:

point	$-90°$	$-45°$	$0°$	$45°$	$90°$
(5, 25)	-25	-14	5	21	25
(20, 10)	-10	7	20	21	10

16

An Example (cont'd)

- We see that the curves in Hough space intersect at 45∘ with ρ=21.
- Curves generated by collinear points in the image space intersect in peaks (ρ,θ) in the Hough transform space.
- The more curves intersect at a point, the more "votes" a line in image space will receive.



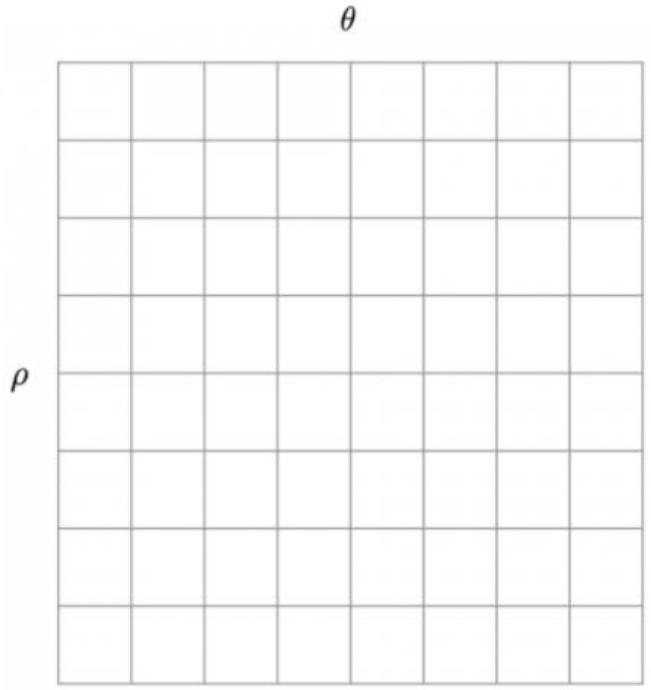
17

Algorithm of the Hough Transform

Algorithm

Step 1:Initialize Accumulator

- First, we need to create an accumulator array.
- The number of cells you choose to have is a design decision.
- Let's say you chose a 10×10 accumulator. It means that ρ can take only 10 distinct values and the θ can take 10 distinct values, and therefore you will be able to detect 100 different kinds of lines.
- The **size** of the **accumulator** will also depend on the image's resolution.



❖ We will see in a moment that the value inside the bin (ρ, θ) will increase as more evidence is gathered about the presence of a line with parameters ρ and θ.

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Algorithm of the Hough Transform

Algorithm

Step 2: Detect Edges

- Now that we have set up the accumulator, we want to collect evidence for every cell because each cell corresponds to one line.

How do we collect evidence?

- The idea is that if there is a visible line in the image, an edge detector should fire at the boundaries of the line.
- These edge pixels provide evidence for the presence of a line.

The output of edge detection is an array of edge pixels

$$[(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)]$$

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

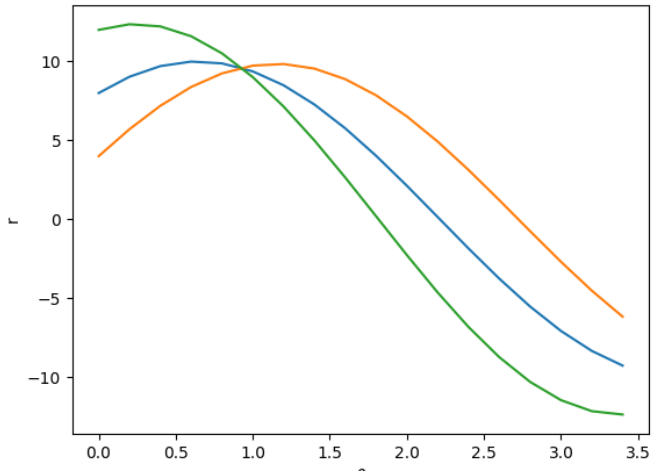
Algorithm of the Hough Transform

Algorithm

Step 3: Voting by Edge Pixels

- For every edge pixel (x, y) in the above array, we vary the values of θ and plug it in equation 1 to obtain a value for ρ.
- In the Figure below we vary the θ for three pixels (represented by the three colored curves) and obtain the values for ρ using equation 1.
- As you can see, these curves intersect at a point indicating that a line with parameters θ = 1 and ρ = 9.5 is passing through them.

- Typically, we have 100s of edge pixels and the accumulator is used to find the intersection of all the curves generated by the edge pixels.
- We do this for every edge pixel and now we have an accumulator that has all the evidence about all possible lines in the image.



We can simply select the bins in the accumulator above a certain threshold to find the lines in the image. If the threshold is higher, you will find fewer strong lines, and if it is lower, you will find a large number of lines including some weak ones.

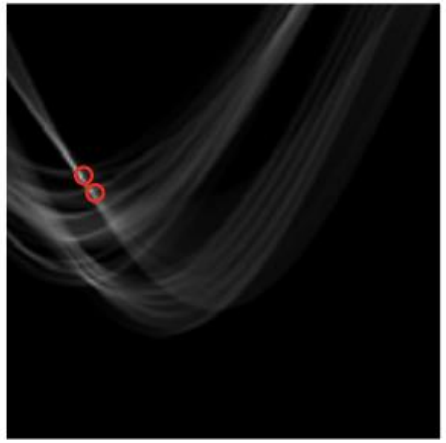
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Line Detection Results

Original Image

Gradient

Edge (Threshold)

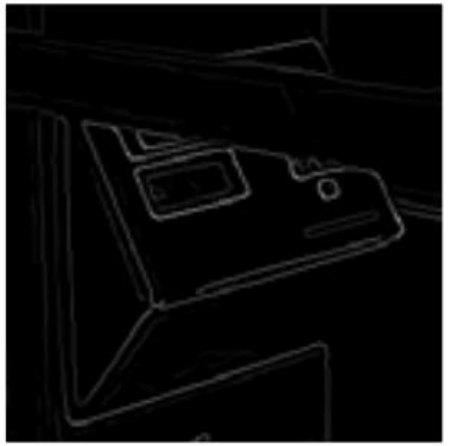


Hough Transform $A(\rho, \theta)$

Detected Lines

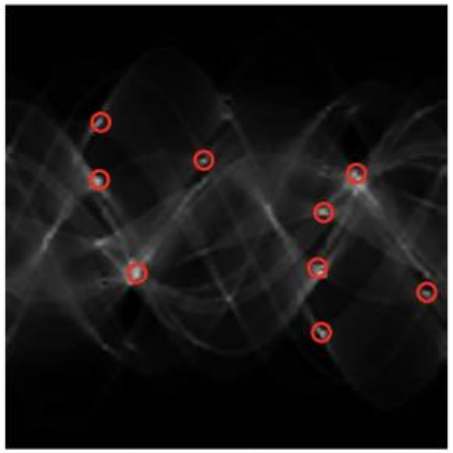
Line Detection Results

Original Image

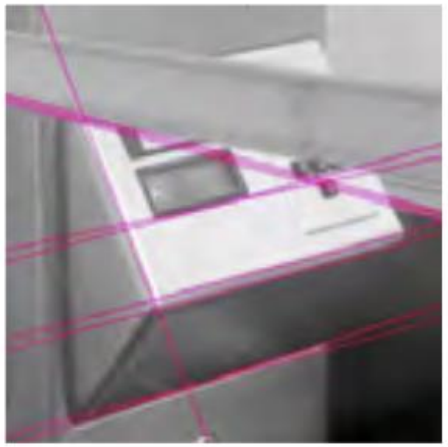


Gradient

Edge (Threshold)



Hough Transform $A(\rho, \theta)$



Detected Lines

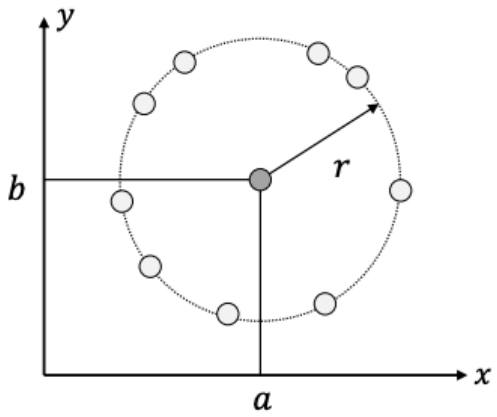
Hough Transform: Circles Detection

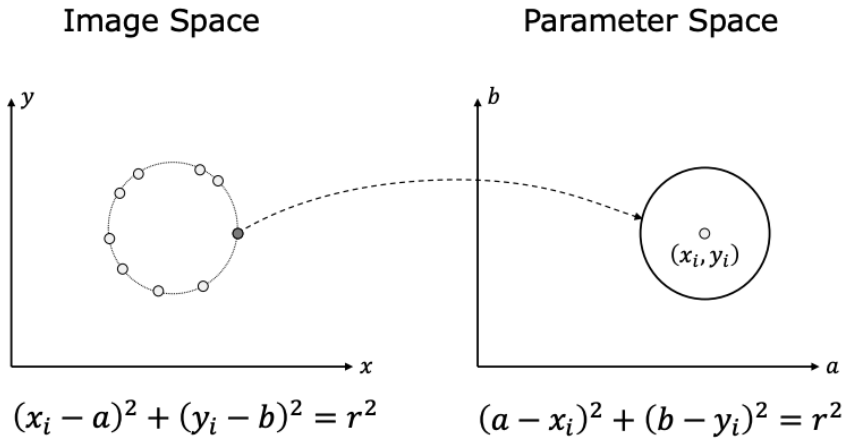
Equation of Circle:

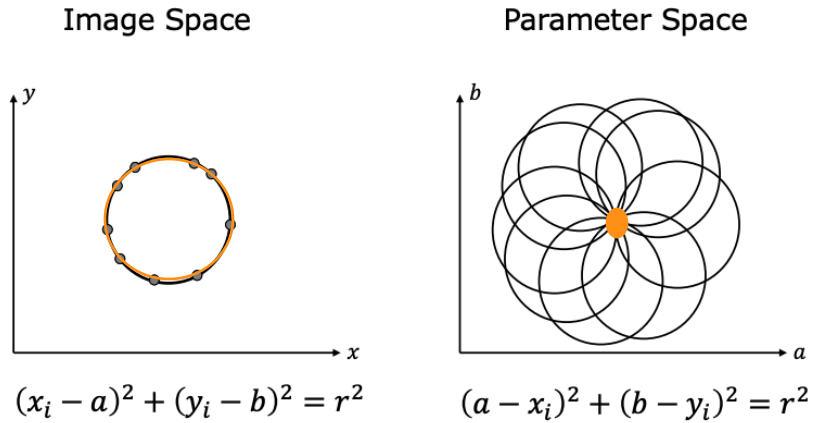
$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

If radius is known: (2D Hough Space)

Accumulator Array $A(a, b)$



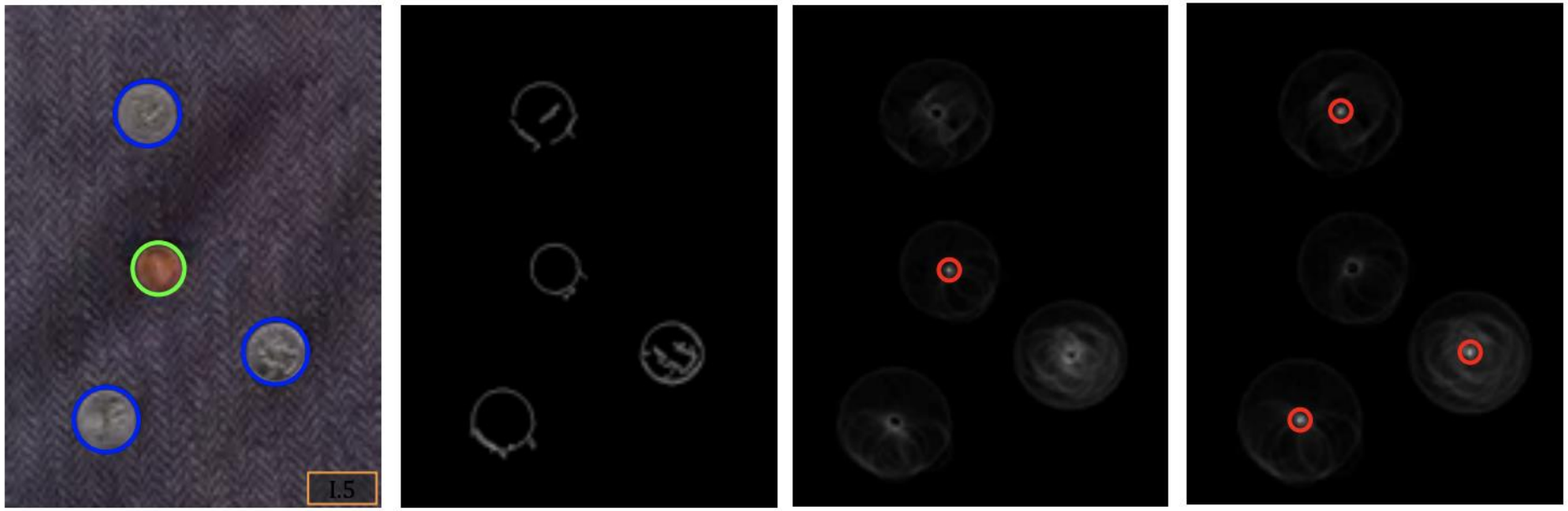




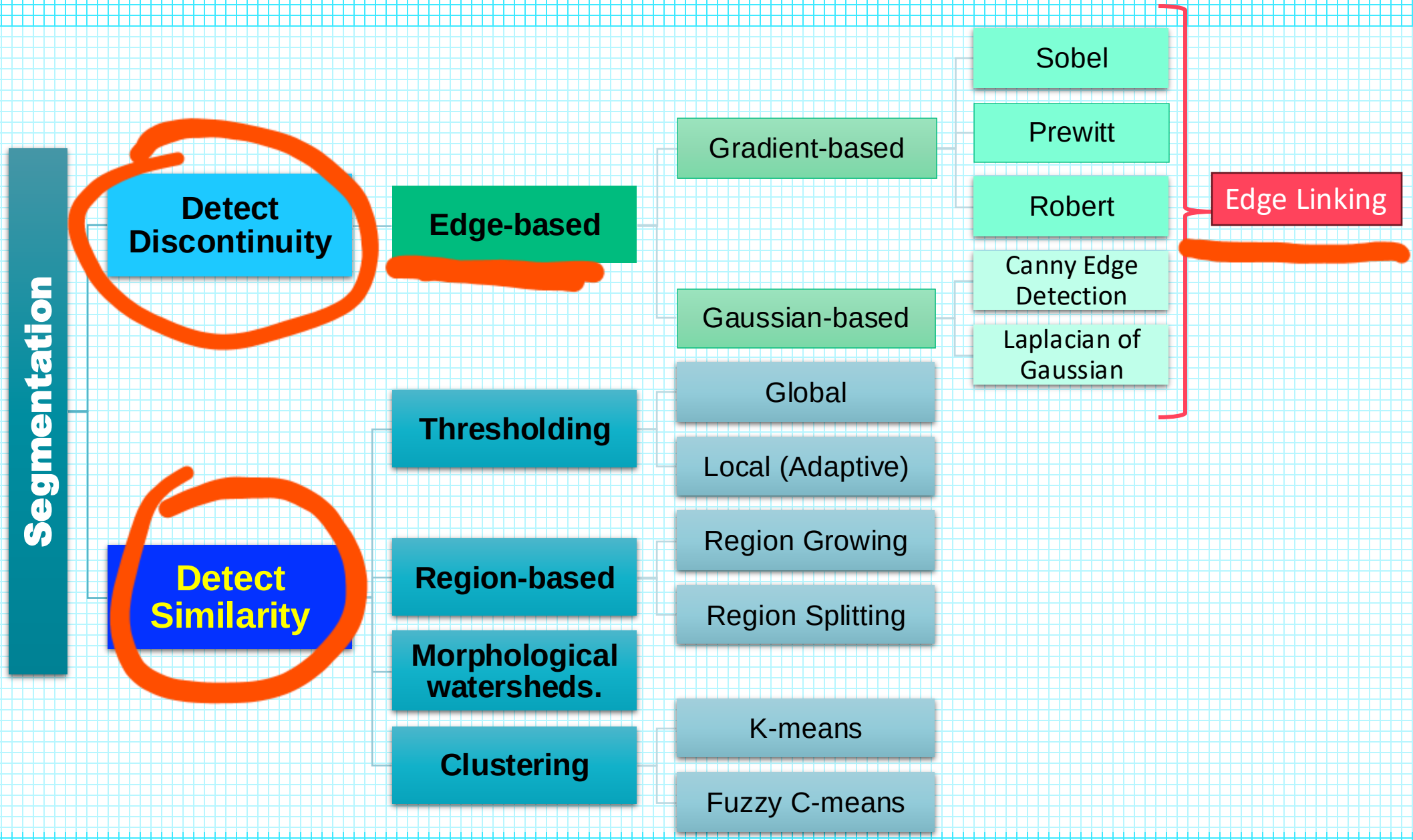
23

Hough Transform: Circles Detection

Penny $(r = r_1)$ Quarter $(r = r_2)$



Original Image | Edge (Threshold) | Hough Transform $A_1(a, b)$ | Hough Transform $A_2(a, b)$


Segmentation
Detect Discontinuity
Edge-based
Gradient-based
Sobel
Prewitt
Robert
Gaussian-based
Canny Edge Detection
Laplacian of Gaussian
Edge Linking
Detect Similarity
Thresholding
Global
Local (Adaptive)
Region-based
Region Growing
Region Splitting
Morphological watersheds.
Clustering
K-means
Fuzzy C-means

Thresholding

- **Thresholding** is one of the easiest methods of image segmentation, where a threshold is set to divide pixels into two classes.
- **Image Thresholding** is a technique used with various applications, including image segmentation, object detection, and character recognition. By separating **objects** from their **background** in an image, image thresholding makes it easier **to analyze and extract relevant information**.
- **Assumption**: the range of intensity levels covered by objects of interest is different from the background.

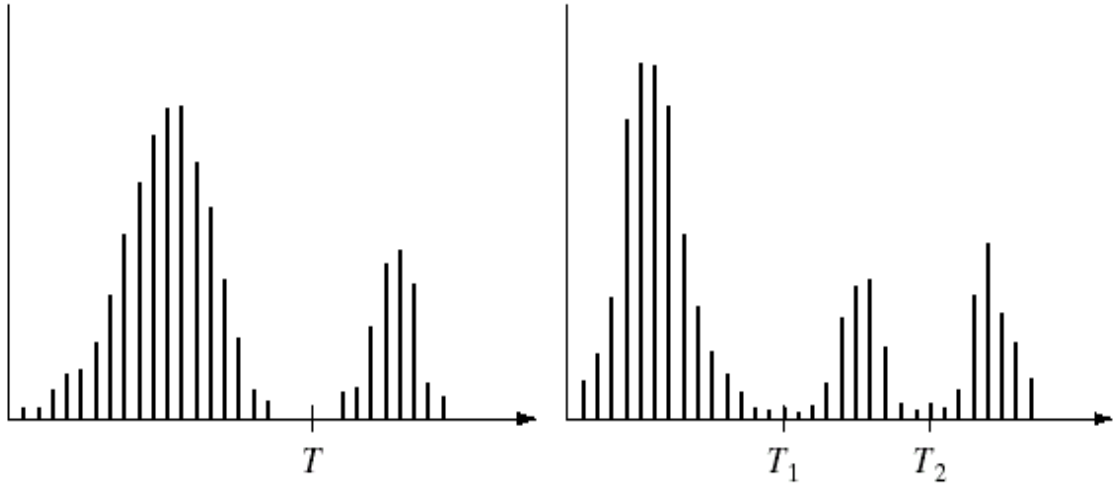
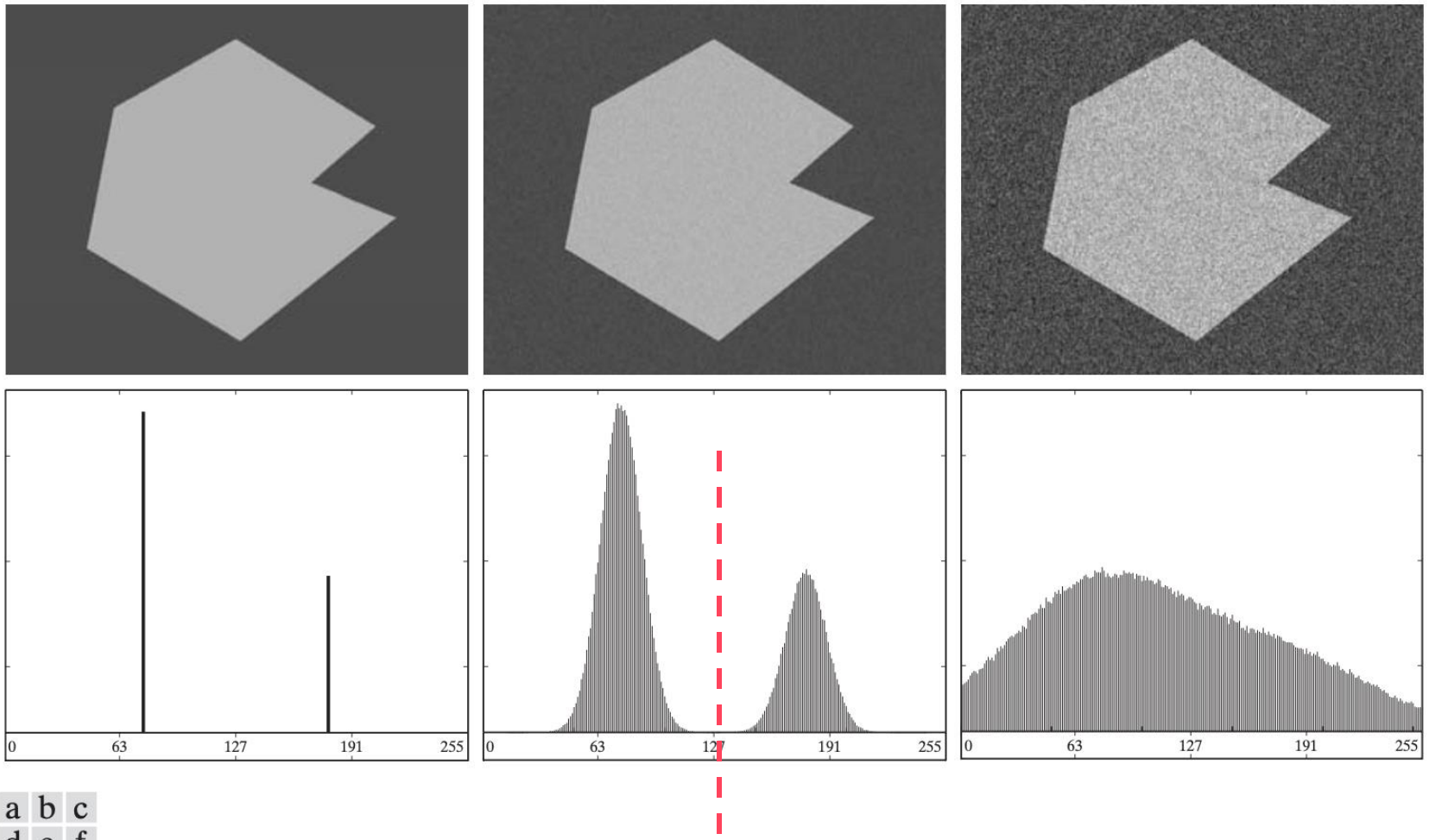


FIGURE 10.26 (a) Gray-level histograms that can be partitioned by (a) a single threshold, and (b) multiple thresholds.

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

$$g(x,y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \le T_2 \\ c & \text{if } f(x,y) \le T_1 \end{cases}$$

Thresholding: The Role of Noise



a b c
d e f

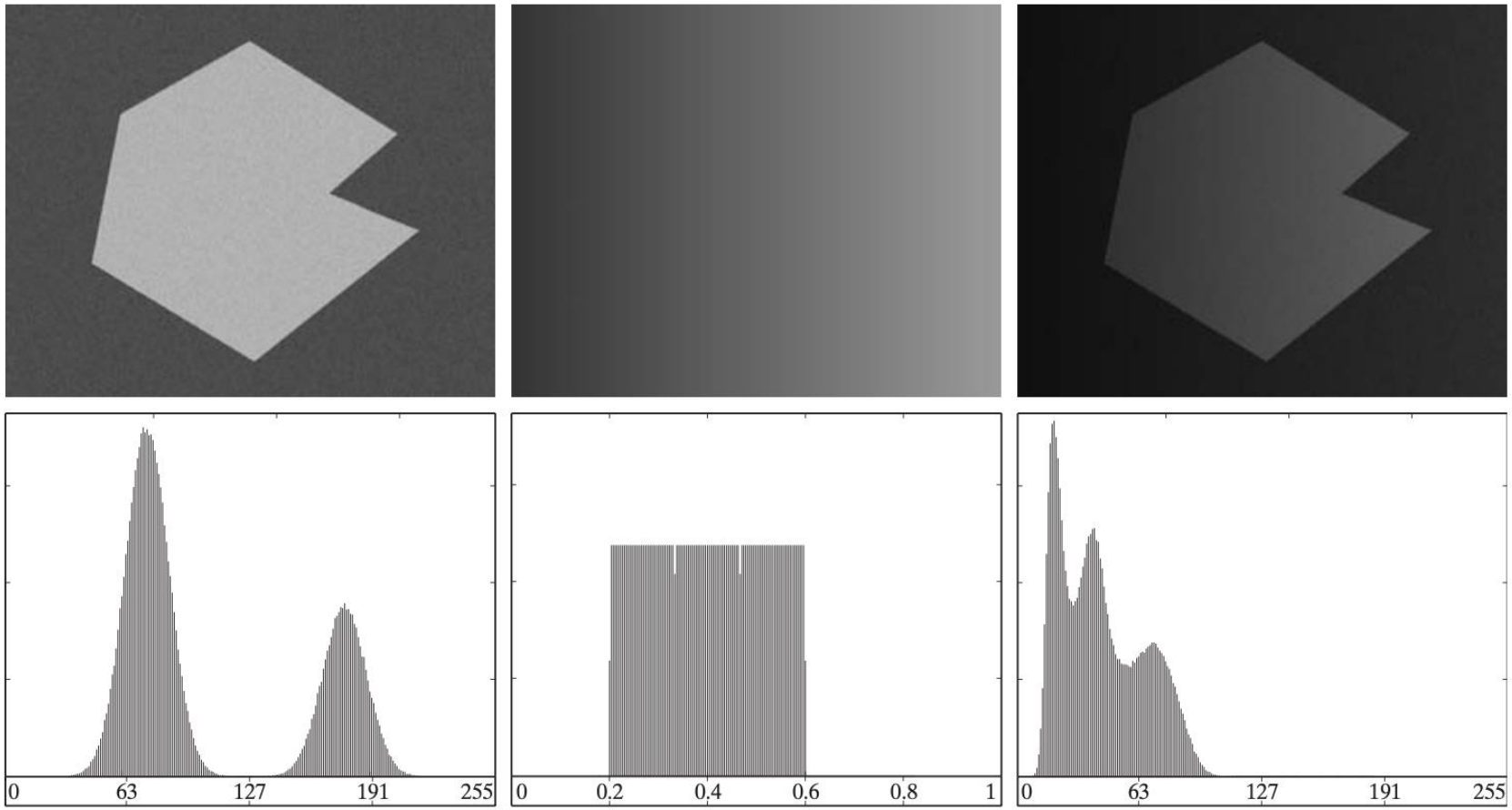
FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

Although the corresponding histogram modes are now broader [10.36(e)], their separation is large enough that the depth of the valley between them makes the modes easy to separate. A **threshold** placed midway between the two **peaks** would do a nice job of **segmenting** the image.

As the histogram in [10.36(f)] shows, the situation is much more serious now, as **there is no way to differentiate** between the two modes.

Thresholding: The Role of Illumination

$r(x, y)$ $i(x, y)$ $f(x, y) = i(x, y)r(x, y)$



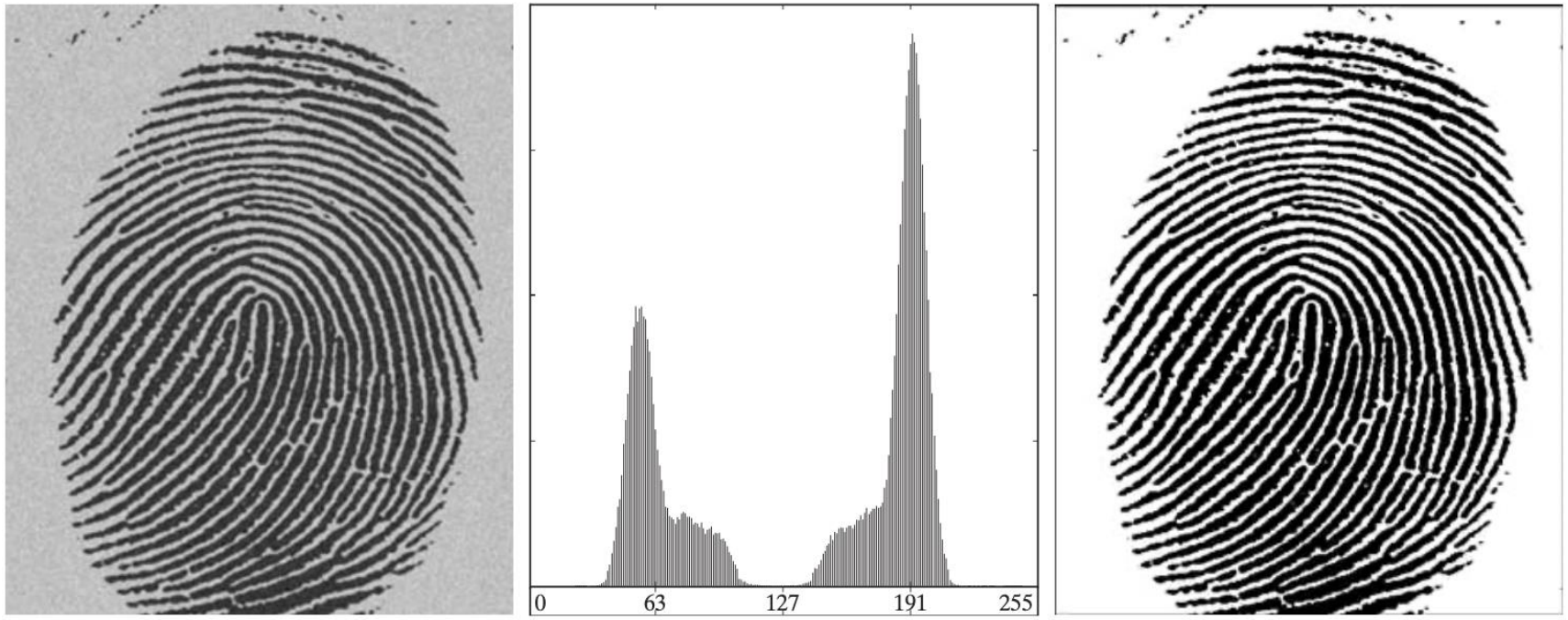
a b c
d e f

FIGURE 10.37 (a) Noisy image. (b) Intensity ramp in the range [0.2, 0.6]. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.

Illumination and reflectance play a **central** role in the success of image segmentation using thresholding. Therefore, controlling these factors when it is possible should be the first step in solving a segmentation problem.

Thresholding: Basic Global Thresholding

- When the **intensity distributions of objects and background pixels are sufficiently distinct**, it is possible to use **a single (global) threshold** applicable over the entire image.
- The initial threshold must be chosen as greater than the minimum and less than the maximum intensity level in the image.
- The average intensity of the image is a good initial choice for T.



a b c

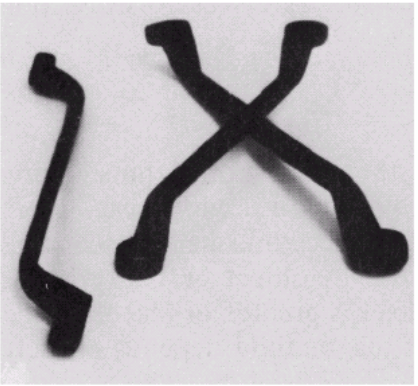
FIGURE 10.38 (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (the border was added for clarity). (Original courtesy of the National Institute of Standards and Technology.)

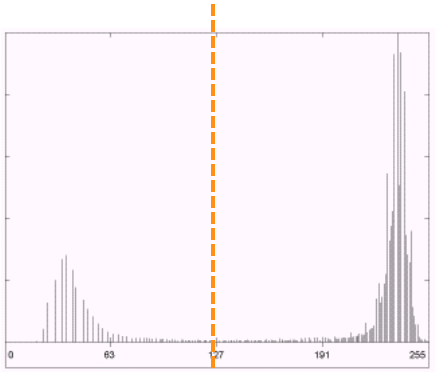
Thresholding: Basic Global Thresholding

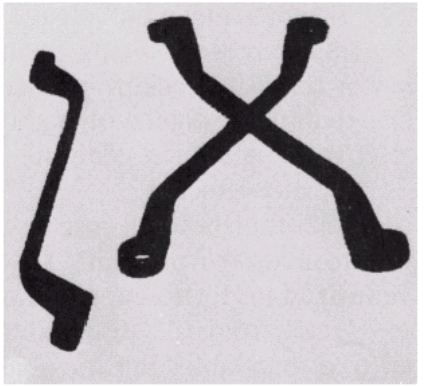
1. Select an initial estimate for the global threshold, T.
2. Segment the image using T in Eq. (10.3-1). This will produce two groups of pixels: G_1 consisting of all pixels with intensity values $> T$, and G_2 consisting of pixels with values $\leq T$.
3. Compute the average (mean) intensity values m_1 and m_2 for the pixels in G_1 and G_2, respectively.
4. Compute a new threshold value:

$$T = \frac{1}{2}(m_1 + m_2)$$

5. Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than a predefined parameter ΔT.







a
b c

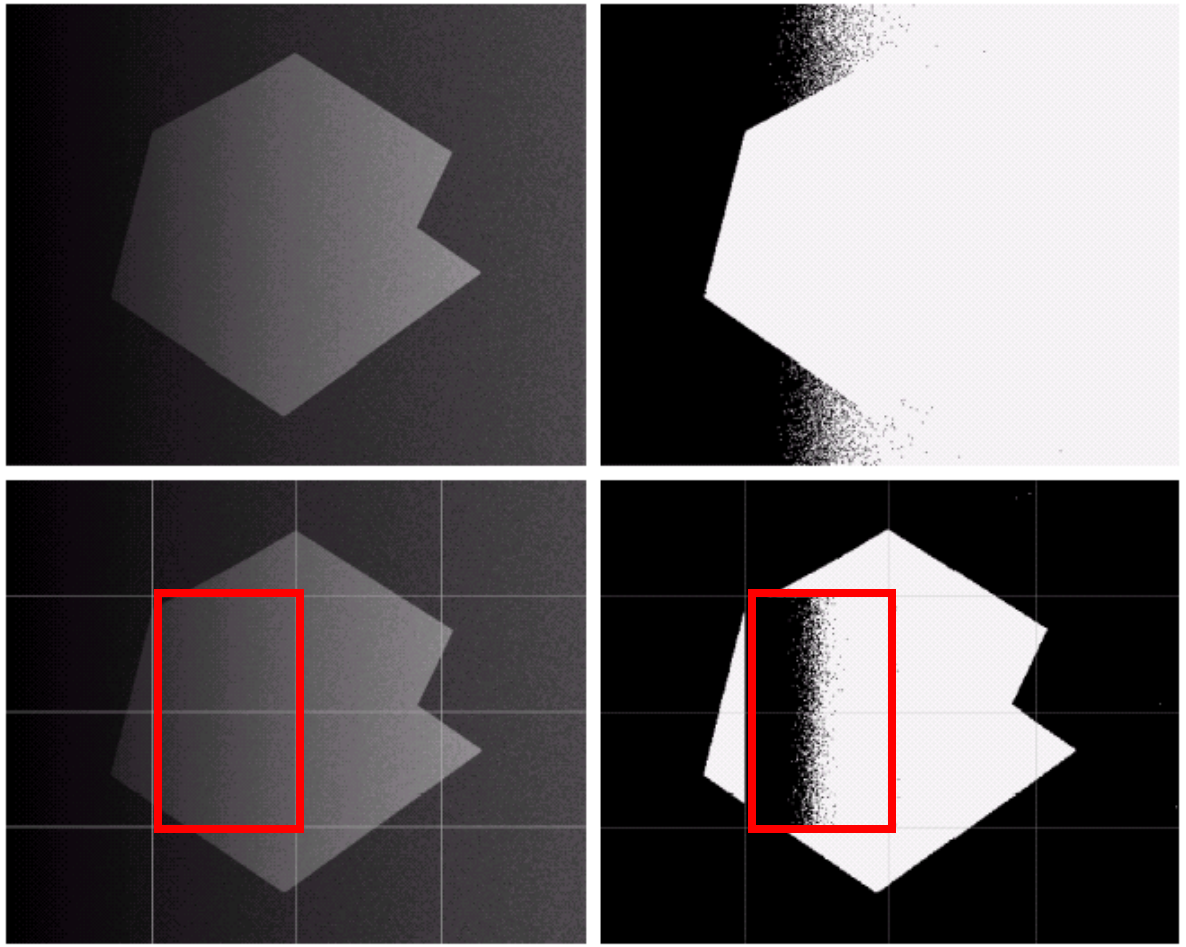
FIGURE 10.28 (a) Original image. (b) Image histogram. (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

Thresholding: Basic Adaptive Thresholding

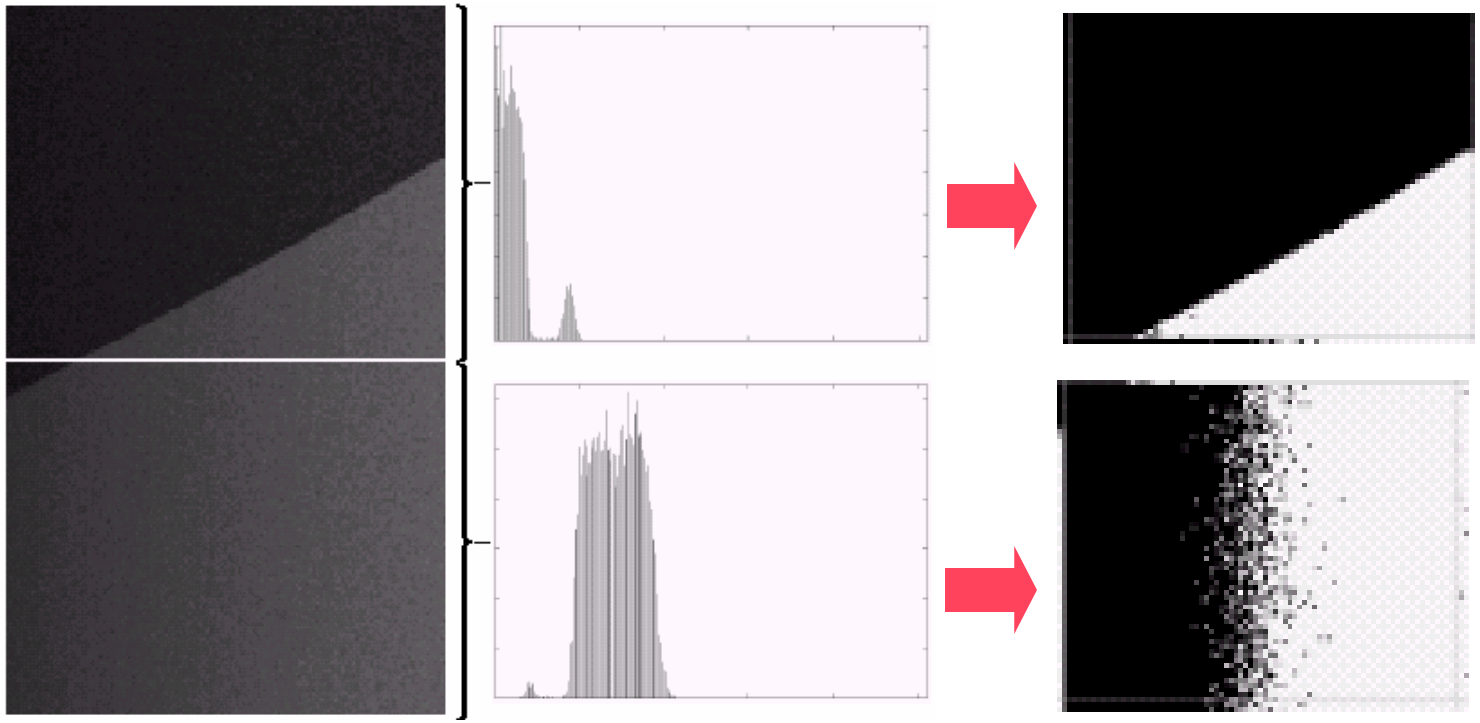
- Uneven Illumination can upset the single value thresholding (b).
- The image shows an example of using adaptive thresholding with the image shown previously (d).
- As can be seen, success is mixed.
- However, we can further subdivide the troublesome sub-images for more success.

a b
c d

FIGURE 10.30
(a) Original image. (b) Result of global thresholding. (c) Image subdivided into individual subimages. (d) Result of adaptive thresholding.

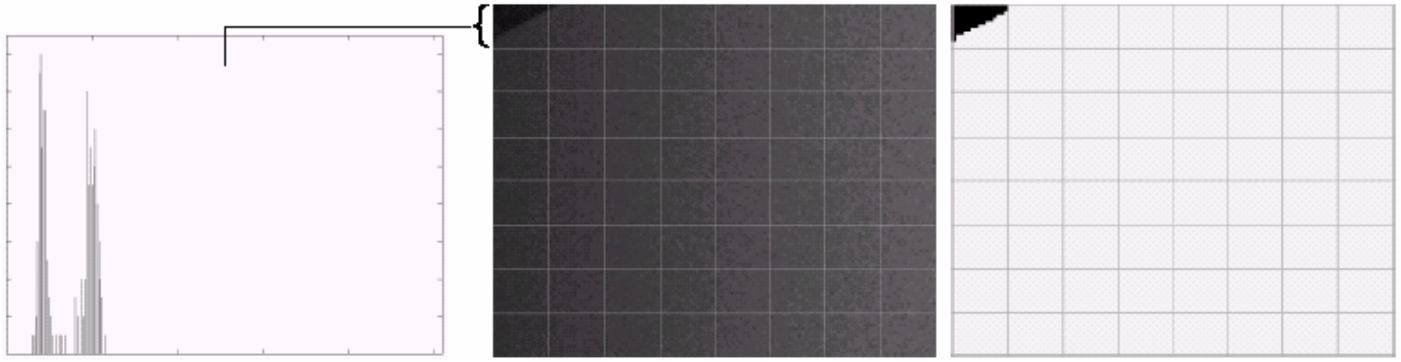


Thresholding: Basic Adaptive Thresholding



How to solve this problem?

Thresholding: Basic Adaptive Thresholding



Answer: subdivision

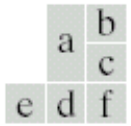


FIGURE 10.31 (a) Properly and improperly segmented subimages from Fig. 10.30. (b)–(c) Corresponding histograms. (d) Further subdivision of the improperly segmented subimage. (e) Histogram of small subimage at top, left. (f) Result of adaptively segmenting (d).

Another Example

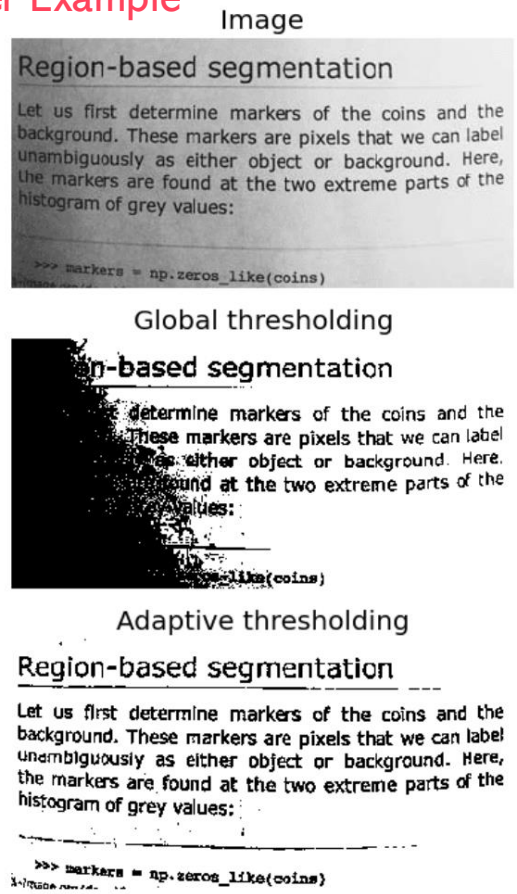
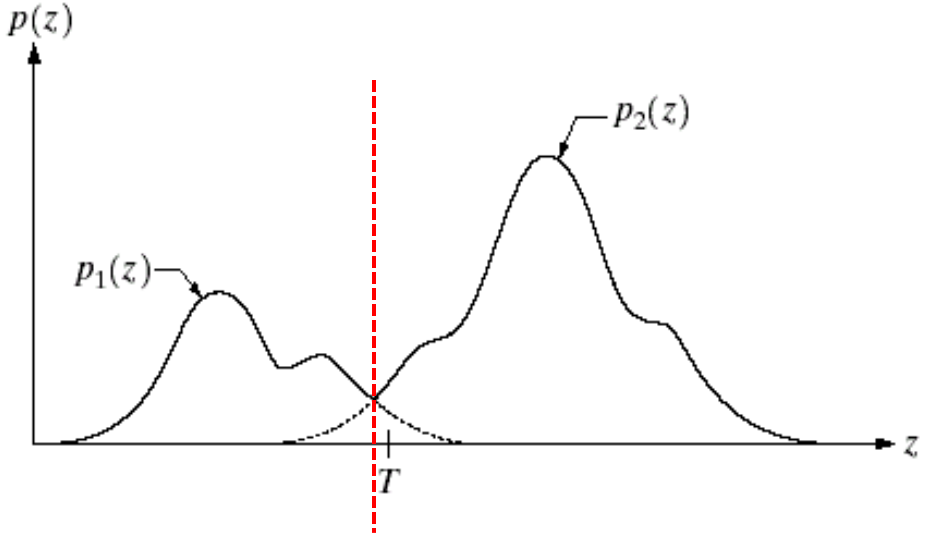


Figure 1: *Top:* Original input image. *Middle:* Applying global thresholding leads to a poor segmentation result. *Bottom:* Using adaptive thresholding creates a much cleaner segmentation (image source).

Thresholding: Optimal Global & Adaptive Thresholding

- This method treats pixel values as probability density functions (pdf).
- The goal of this method is to minimize the probability of misclassifying pixels as either objects or backgrounds.
- There are two kinds of errors:
  - mislabeling an object pixel as background, and
  - mislabeling a background pixel as an object.

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



Segmentation error

Region-Based Segmentation

- Edges and thresholds sometimes do not give good segmentation results.
- **Region-based segmentation** is based on the **connectivity of similar** pixels in a region.
  a. Each region must be uniform.
  b. The connectivity of the pixels within the region is very important.
- Region-based segmentation has two main approaches:
  1. Region growing.
  2. Region splitting.

Region-Based Segmentation: Basic Formulation

- Let ***R* represent** the **entire image region**.
- Segmentation is a process that **partitions *R* into subregions, $R_1, R_2, \ldots, R_n$,** such that:

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

Every pixel must be in a region

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

Pixels in a region must be connected in some predefined sense

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

Region must be distinct

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

(e) $P(R_i \cup R_j) = \text{FALSE}$ for any adjacent regions R_i and R_j

Region R_i and R_j are different w.r.t. P

where $P(R_k)$ is a logical predicate defined over the points in set Rk

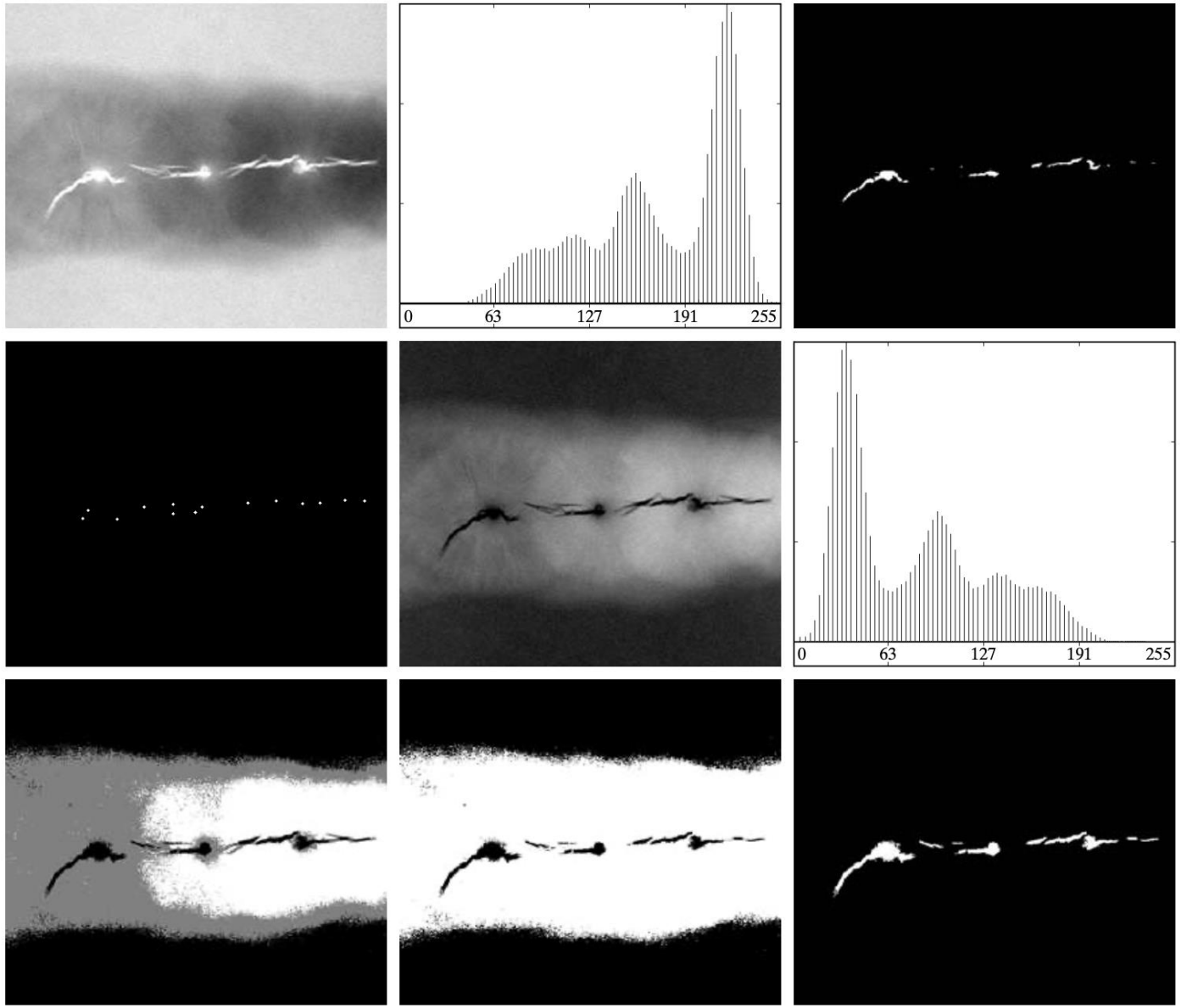
. For example, ***P*(R_k)=TRUE** if all pixels in ***R_k* have the same gray level**.

Region-Based Segmentation: Region Growing

Grouping of **pixels** or **subregions** into **larger regions** based on **predefined criteria for growth.**

- **General Procedure:**
  1. Start with a set of "**seed**" points.
  2. From each seed, grow regions by appending recursively the neighboring pixels that satisfy the "similar criterion".

- **Growing Criteria:**
  1. The absolute gray-level difference between any pixel and the seed is less than a specific value.
  2. Apply on 8-connectivity



a b c
d e f
g h i

FIGURE 10.51 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between (a) and (c). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference image thresholded with the smallest of the dual thresholds. (i) Segmentation result obtained by region growing. (Original image courtesy of X-TEK Systems, Ltd.)

Region-Based Segmentation: Region Growing

- **Algorithm of basic region-growing based on 8-connectivity:**

  - **Start with a single pixel (seed) and add new pixels slowly:**

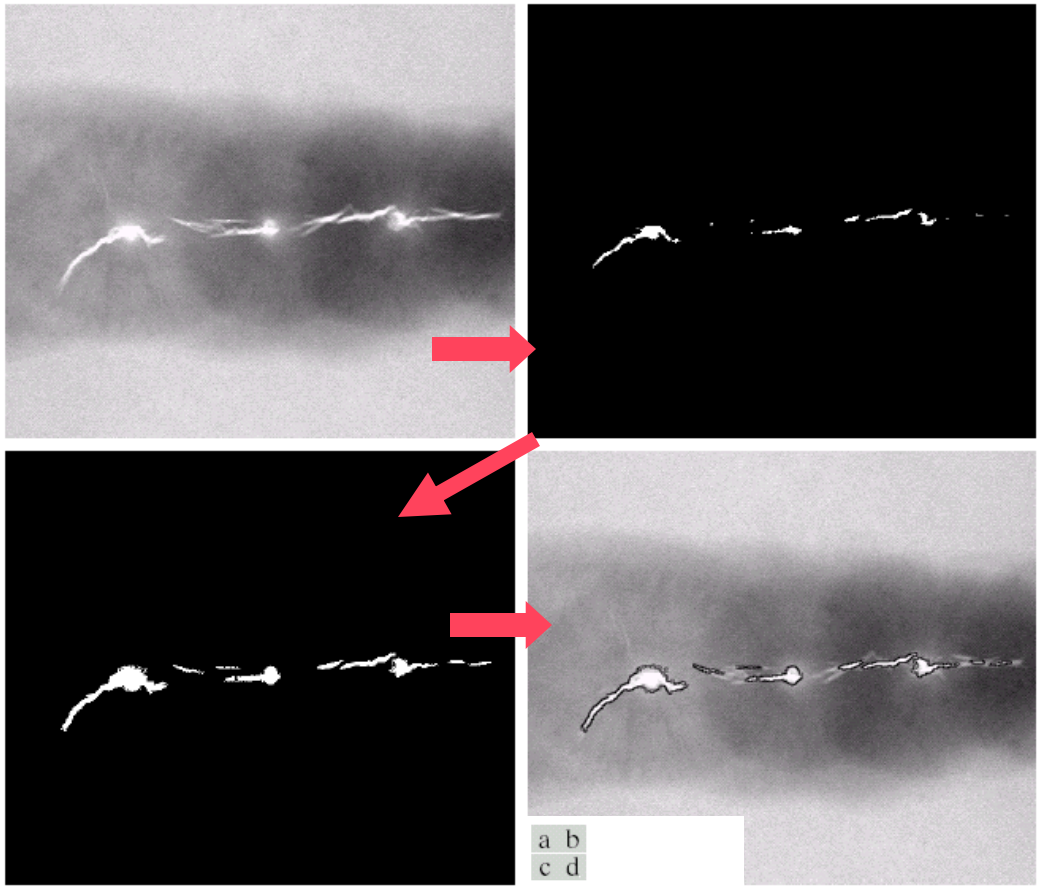
1. Choose the seed pixel
2. Check the neighboring pixels and add them to the region if they are similar to the seed
3. Repeat step 2 for each of the newly added pixels; stop if no more pixels can be added.

$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$

8-neighbourhood

Example of **similarity**:

$|f(seedi) - f(pixel)| < Threshold$



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).

- **Advantages:**
  - Borders found are perfectly thin and connected.
  - The algorithm is stable w.r.t. noise.

Region Growing: illustration example

α)

	1	2	3	4	5	6	7
1	5	6	1	1	1	0	0
2	6	7	7	1	1	1	6
3	3	6	7	7	7	6	5
4	4	3	7	7	7	5	3
5	3	3	2	1	3	3	4
6	2	2	1	2	1	2	3
7	1	1	1	1	1	2	2

β)

	1	2	3	4	5	6	7
1	β	β	α	α	α	α	α
2	β	β	β	α	α	α	β
3	γ	β	β	β	β	β	β
4	γ	γ	β	β	β	β	γ
5	γ	γ	γ	α	γ	γ	γ
6	γ	γ	α	γ	α	γ	γ
7	α	α	α	α	α	γ	γ

Similarity criteria:

$|f(seed_i) - f(neighbors)| <= 1$

Region-Based Segmentation: Region Splitting and Merging

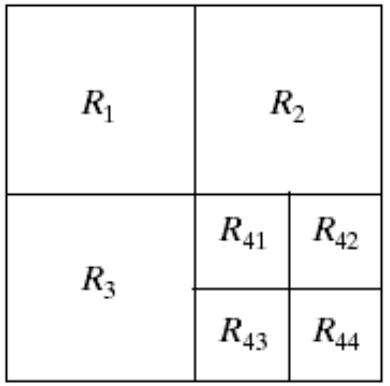
Region splitting is the opposite of region growing.

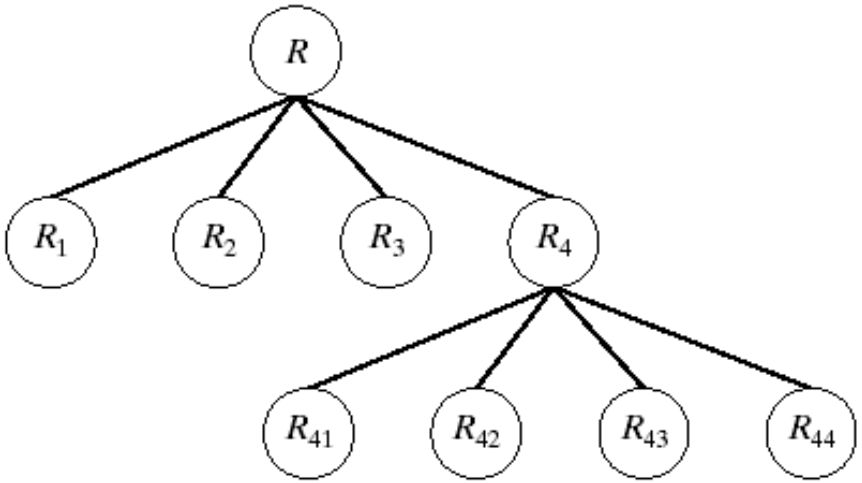
- First, there is a large region (possibly the entire image).
- Then, a predicate (measurement) is used to determine if the region is uniform.
- If not, then the method requires that the region be split into two regions.
- Then, each of these two regions is independently tested by the predicate (measurement).
- This procedure continues until all resulting regions are uniform.

The main problem with region splitting is determining where to split a region.
One method to divide a region is to use a quadtree structure.
Quadtree: a tree in which nodes have exactly four descendants.

a b

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





Region-Based Segmentation: Region Splitting and Merging

- The **split** and **merge** procedure:
  - Split into four disjoint quadrants any region R_i for which $P(R_i) = \text{FALSE}$.
  - Merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k) = \text{TRUE}$. (the quadtree structure may not be preserved)
  - **Stop** when **no further merging or splitting is possible**.

a b c

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

A Gateway to ML Clustering

Classical Image Segmentation

Machine Learning Image Clustering

Clustering: Color Image Segmentation

Goal: Break up the image into meaningful or perceptually similar regions

A Gateway to ML Clustering

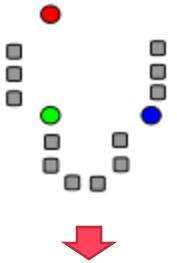
- **Clustering:** Grouping together similar 'objects' (e.g. instances in a dataset) and representing them with a single token, and relatively dissimilar to the objects belonging to other clusters.

- How do we cluster? (ML Unsupervised learning algorithms)

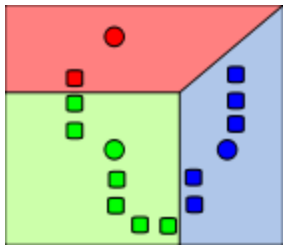
1. **K-means Clustering**
   – Iteratively re-assign points to the nearest cluster center
2. **Agglomerative Clustering**
   – Start with each point as its own cluster and iteratively merge the closest clusters
3. **Mean-shift Clustering**
   – Estimate modes of probability density function (**pdf**)

1). K-means clustering algorithm

1. Randomly select **K** centers



2. Assign each point to **nearest center**



3. Compute new center (**mean**) for each **cluster**

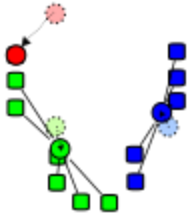
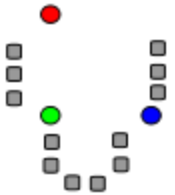


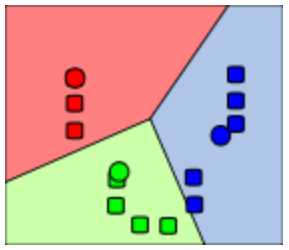
Illustration: http://en.wikipedia.org/wiki/K-means_clustering

K-means clustering algorithm

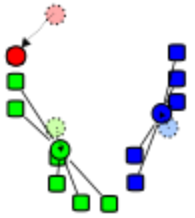
1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster

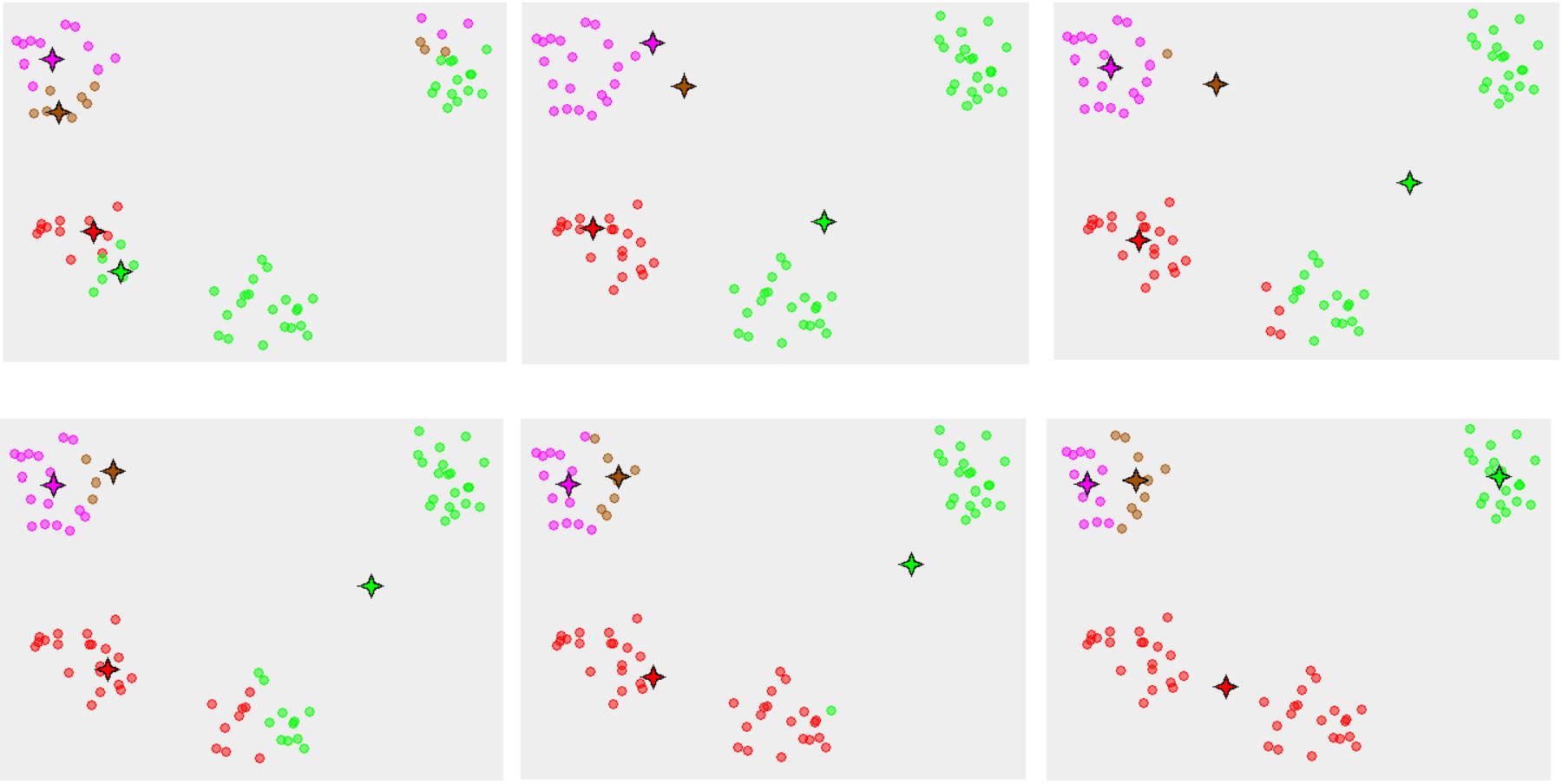


Back to 2

Repeat 2-3 until no points are re-assigned (t=t+1)

Illustration: http://en.wikipedia.org/wiki/K-means_clustering

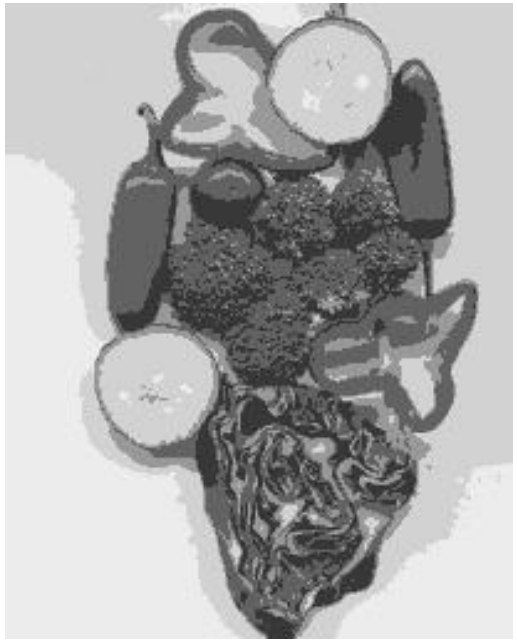
K-means algorithm convergence



K-means clustering using intensity or color

Image

Clusters on intensity



Clusters on color

K=5

K=11

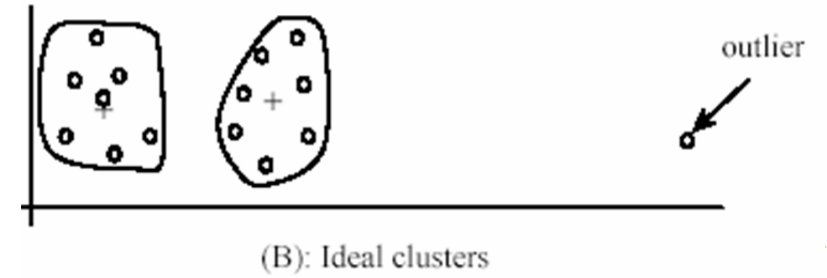
K-means clustering Pros & Cons

Pros

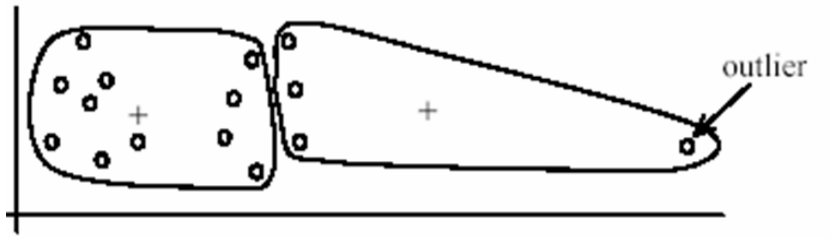
- Finds cluster centers that minimize conditional variance (good representation of data)
- Simple and fast*
- Easy to implement

Cons

- Need to choose K
- Sensitive to outliers

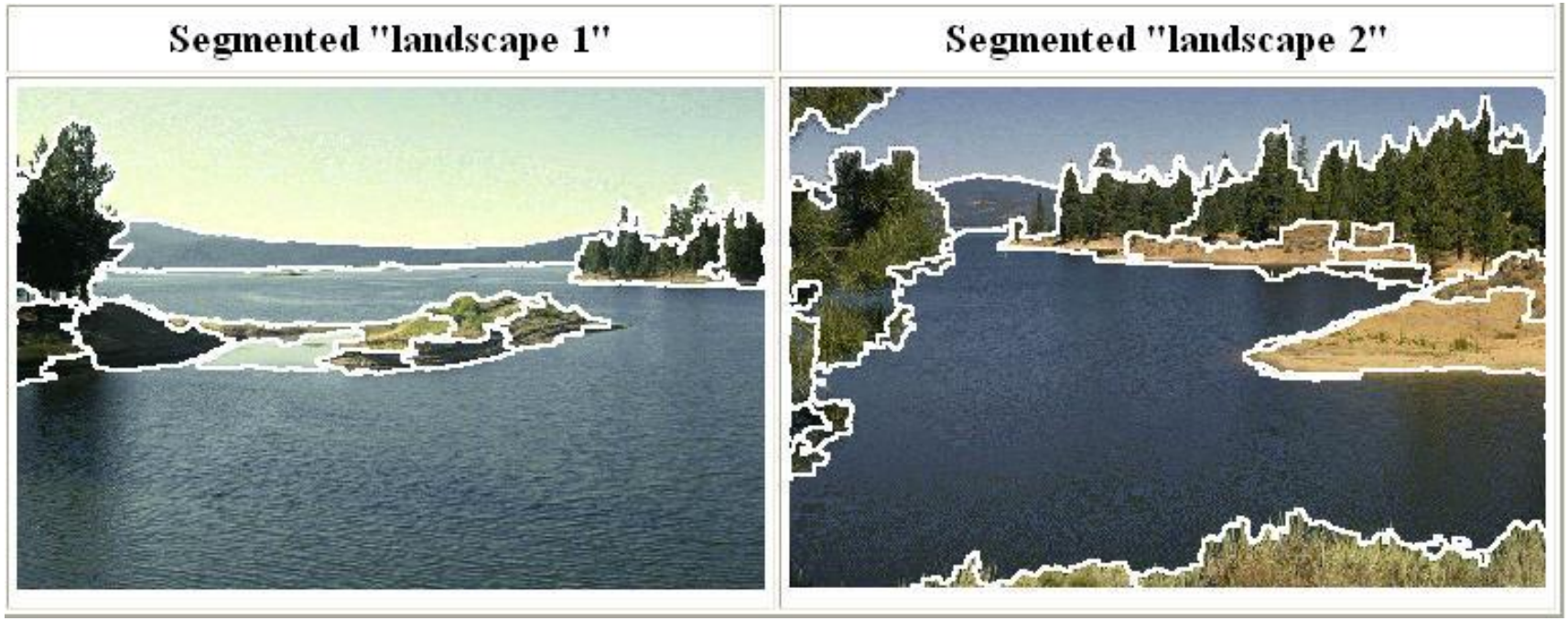


(B): Ideal clusters



2). Mean-Shift clustering algorithm

- **Mean shift clustering** is a method used to find clusters in data **without specifying the number of clusters** beforehand.
- **Mean shift clustering** is a **non-parametric**, reiterative algorithm that seeks to identify groups in a dataset by discovering the sharper peaks in a density function.



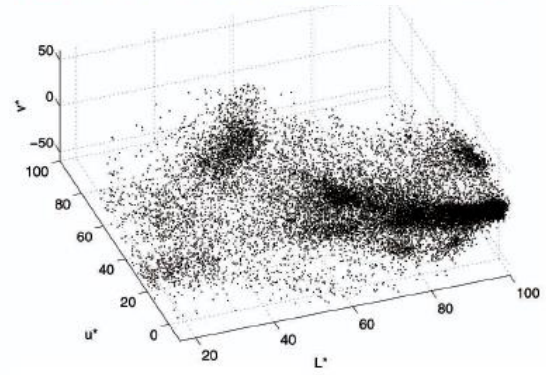
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

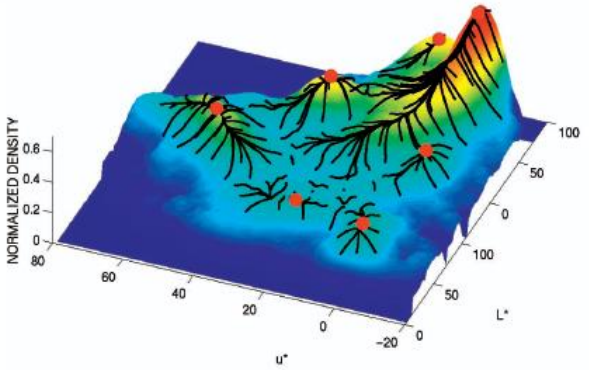
2). Mean-Shift clustering algorithm

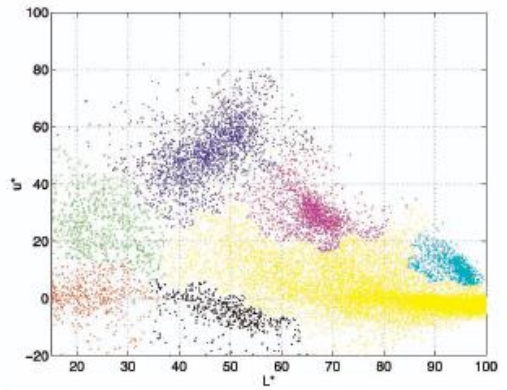
- Mean shift clustering is **non-parametric, unsupervised method** that effectively identifies **the modes or high-density regions of data**, which correspond to the **different segments in an image**.
- By iteratively shifting data points towards the average of points within a given window, mean shift clustering can discover arbitrarily shaped and to partition the image into meaningful regions based on pixel intensity and color.
- The method's ability to handle complex structures and its robustness to noise further enhance its effectiveness in accurately segmenting images.

Try to find *modes* of a non-parametric density.

Color space

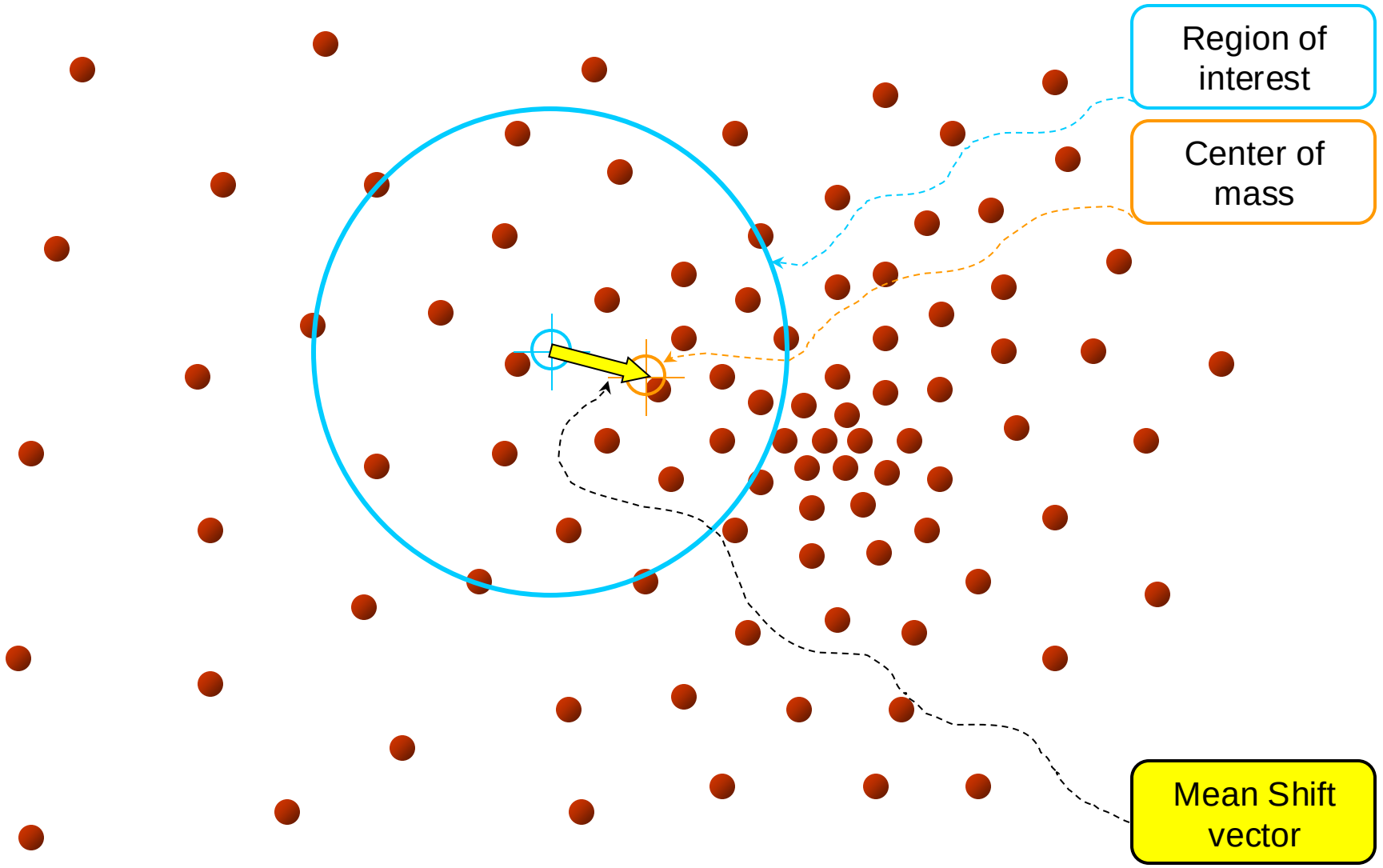






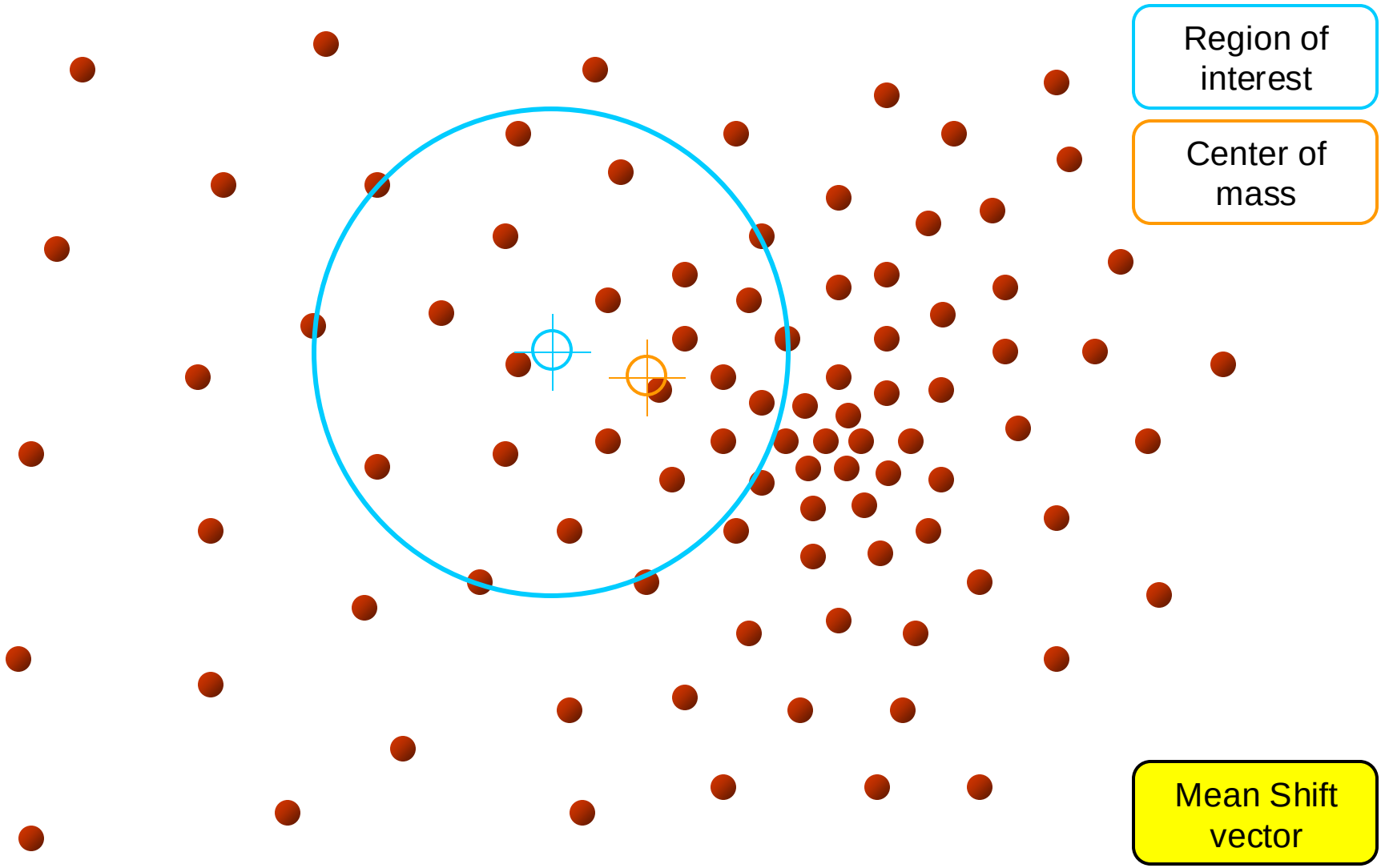
Color space clusters

Mean-Shift Clustering Steps



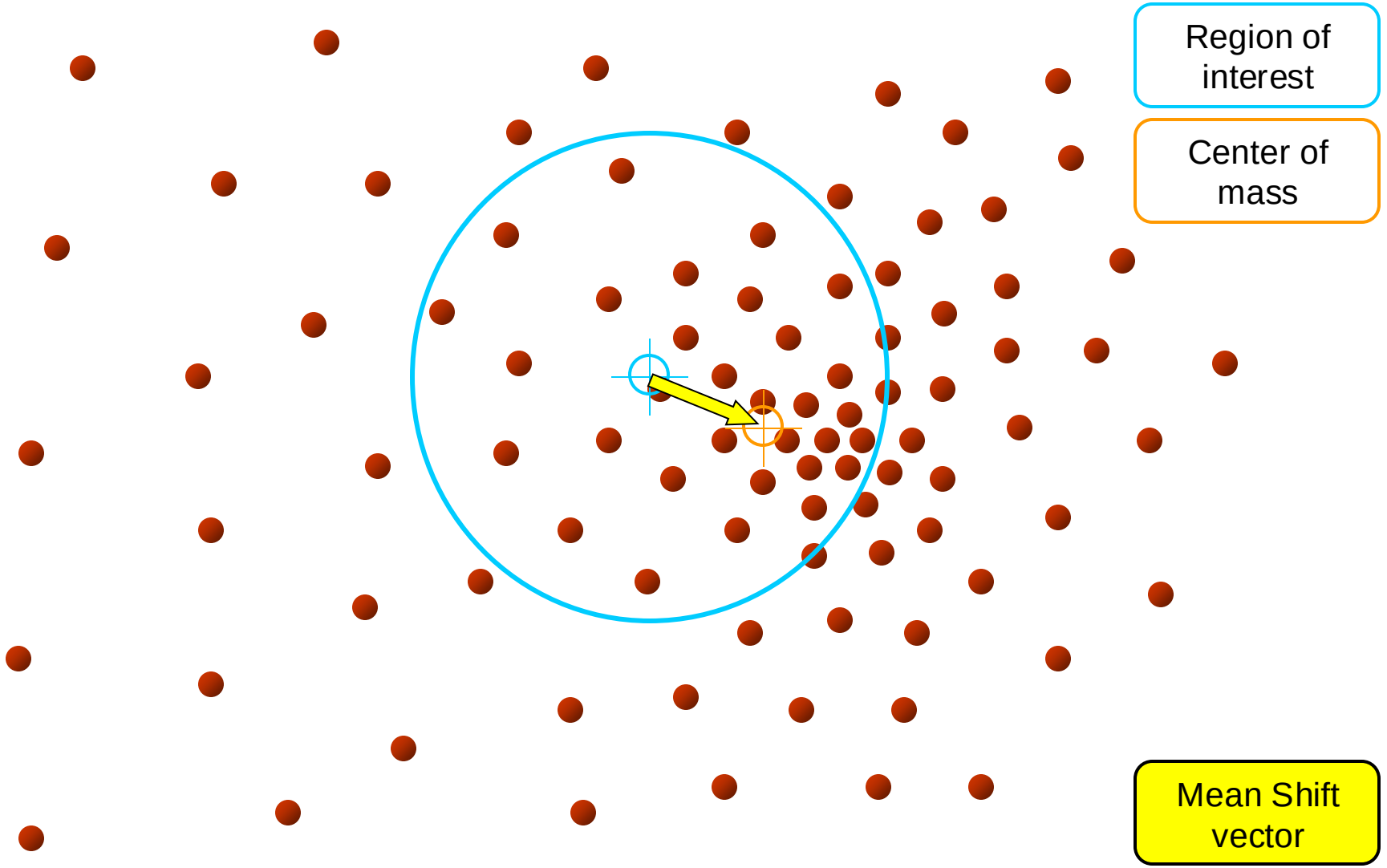
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering Steps



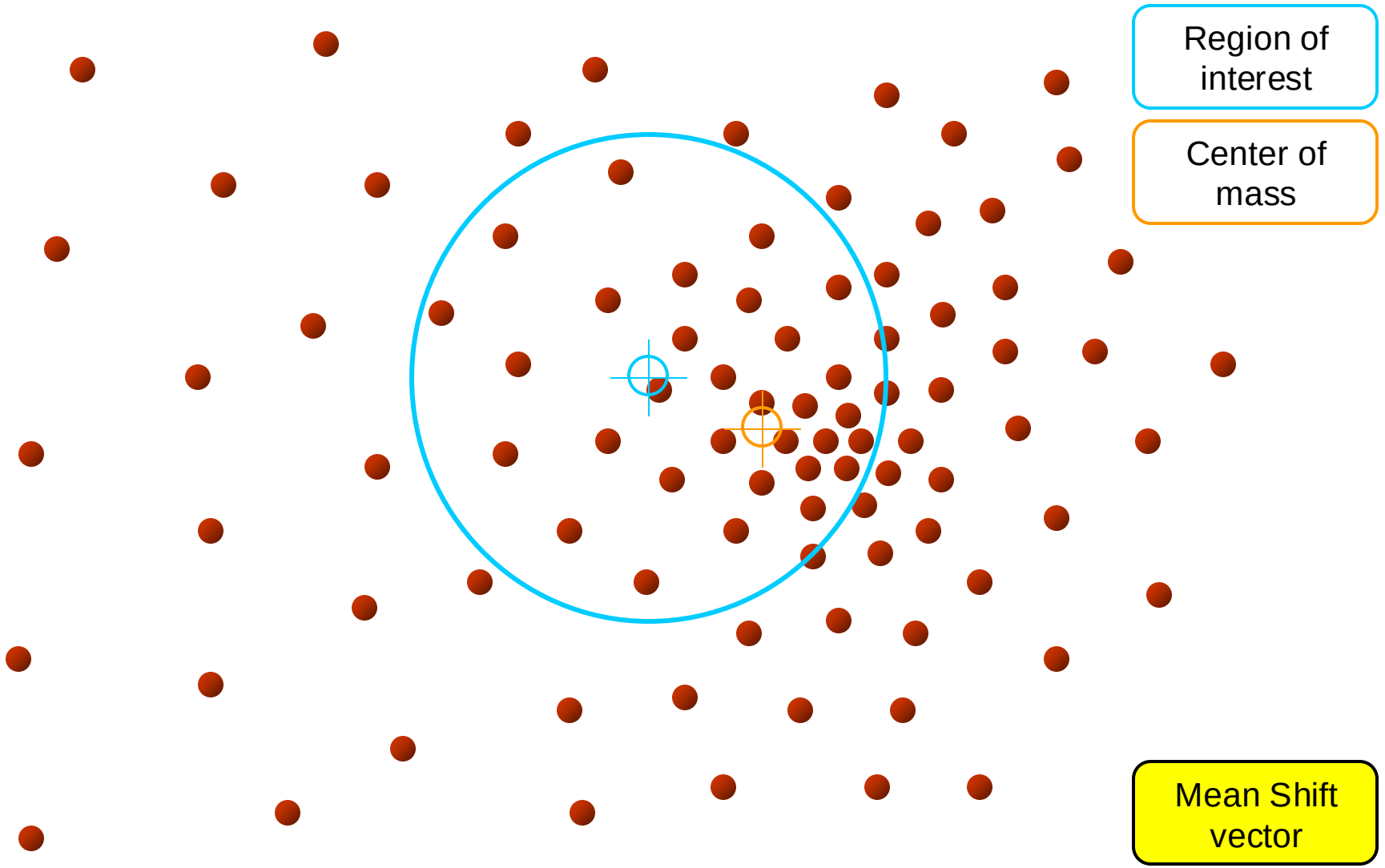
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering Steps



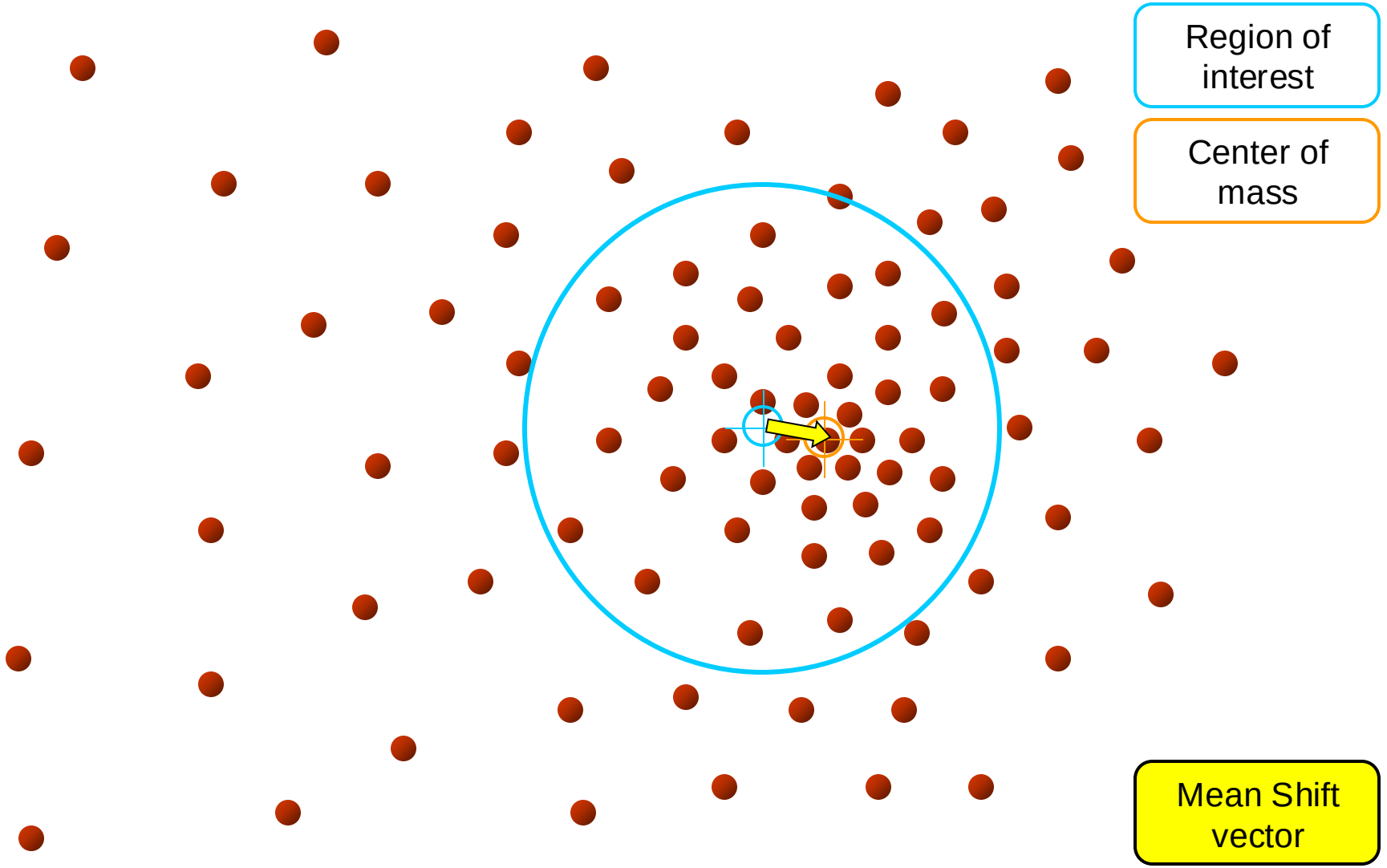
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering Steps



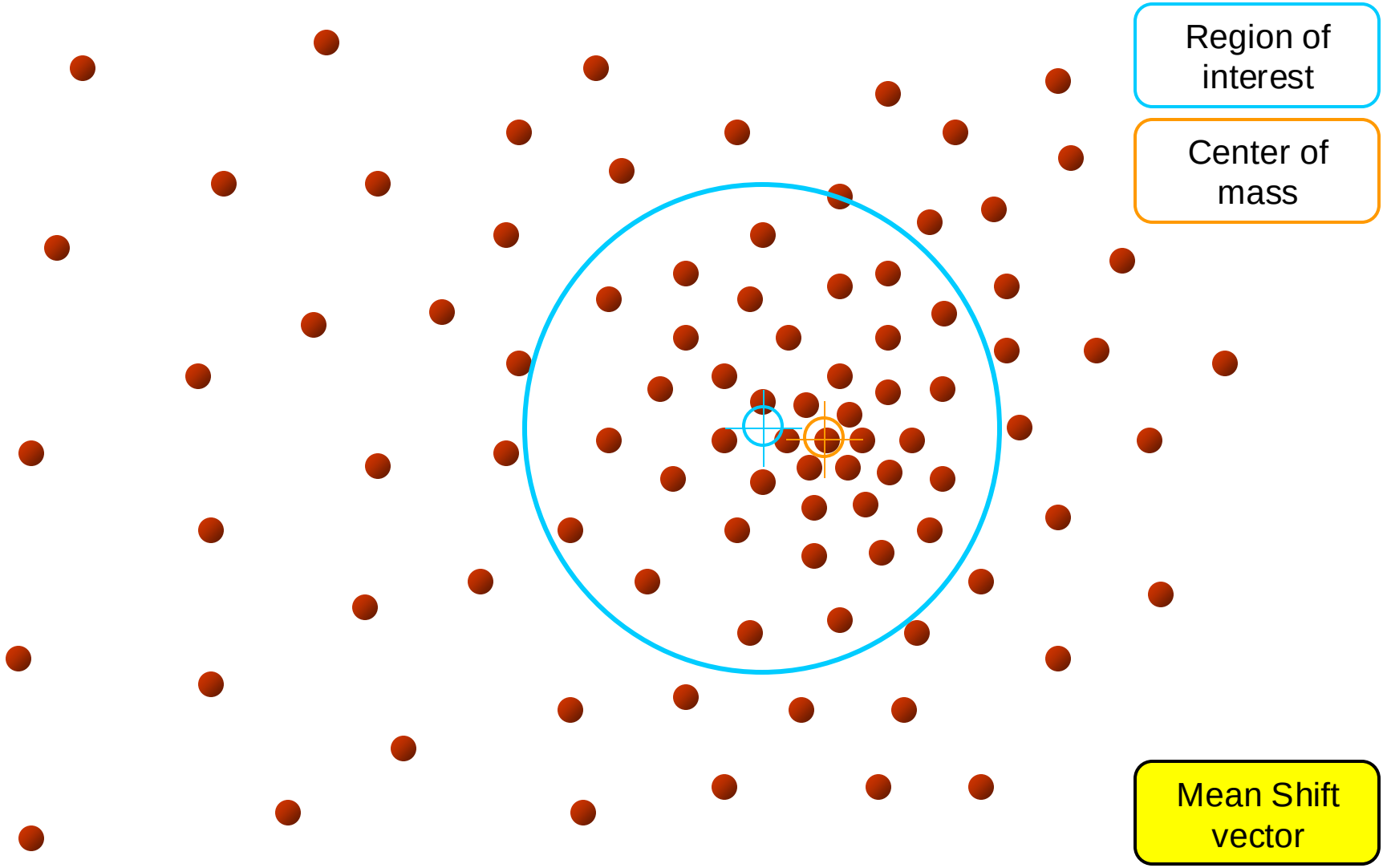
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering Steps



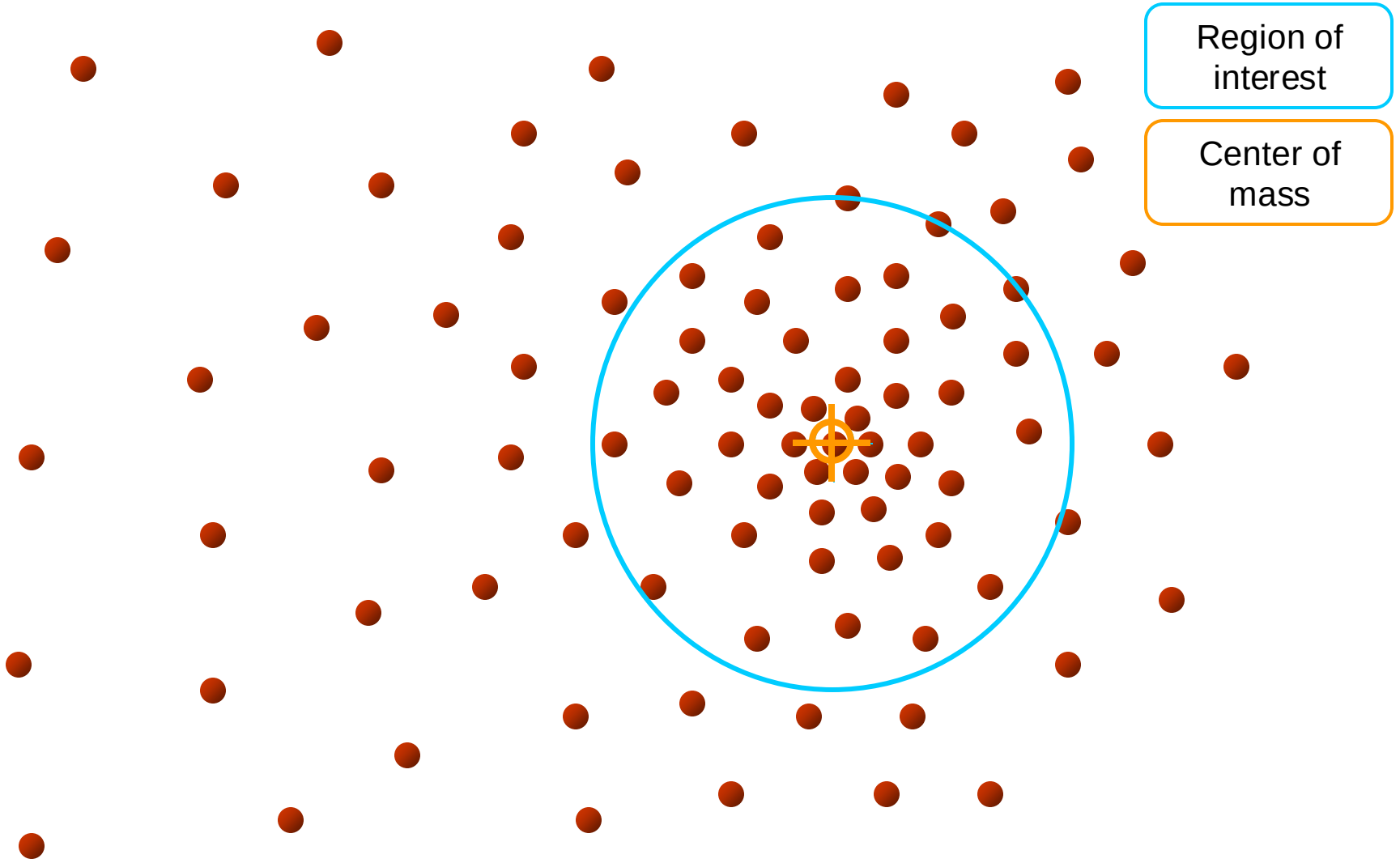
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering Steps



Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering Steps



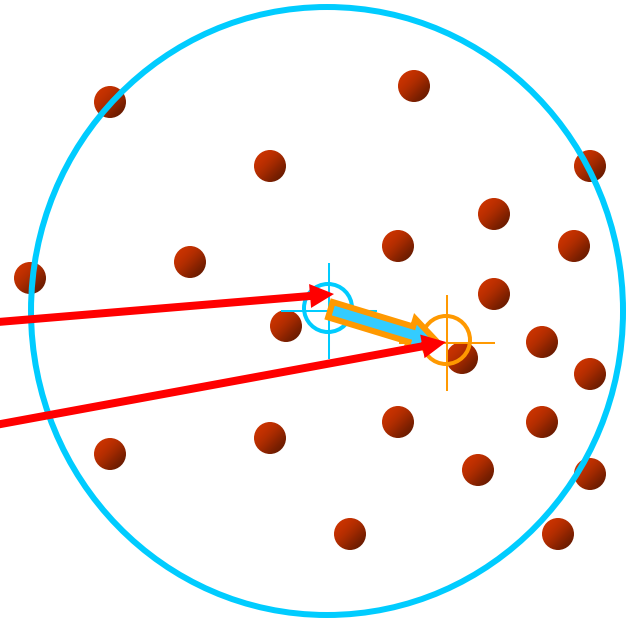
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering Steps

Simple Mean Shift procedure:

- Compute mean shift vector **m(x)**
- Iteratively translate the kernel window by **m(x)** until convergence.

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)} \right] - \mathbf{x}$$

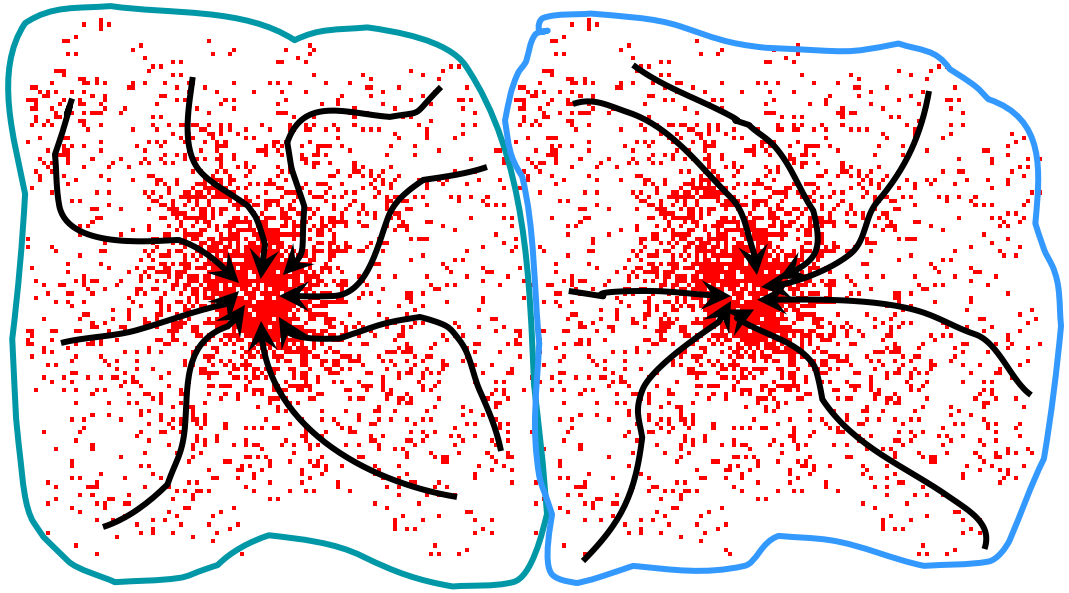


$$g(\mathbf{x}) = -k'(\mathbf{x})$$

Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering Steps

- **Attraction basin:** the region for which all trajectories lead to the same mode.
- **Cluster:** all data points in the attraction basin of a mode.



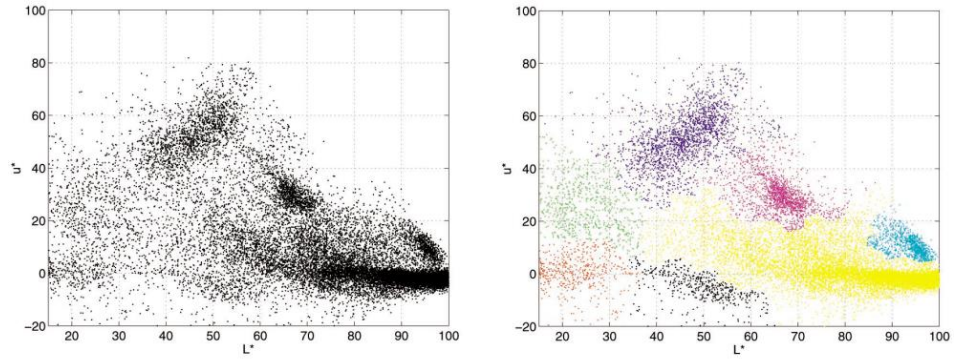
Slide by Y. Ukrainitz & B. Sarel

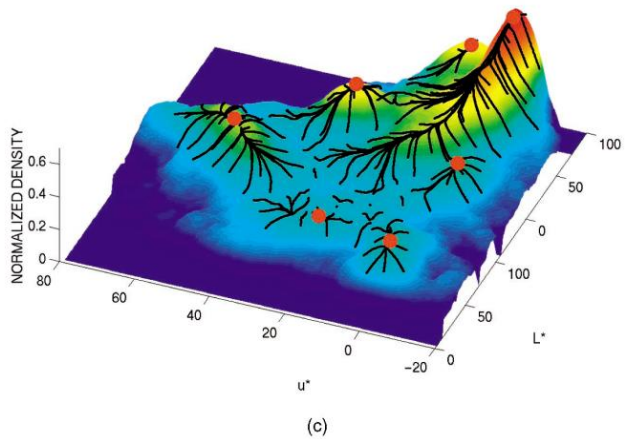
Mean-Shift Clustering Steps

The **mean shift algorithm** seeks ***modes*** of the given set of points

1. Choose kernel and bandwidth
2. For each point:
   a) Center a window on that point
   b) Compute the mean of the data in the search window
   c) Center the search window at the new mean location
   d) Repeat (b,c) until convergence
3. Assign points that lead to nearby modes to the same cluster

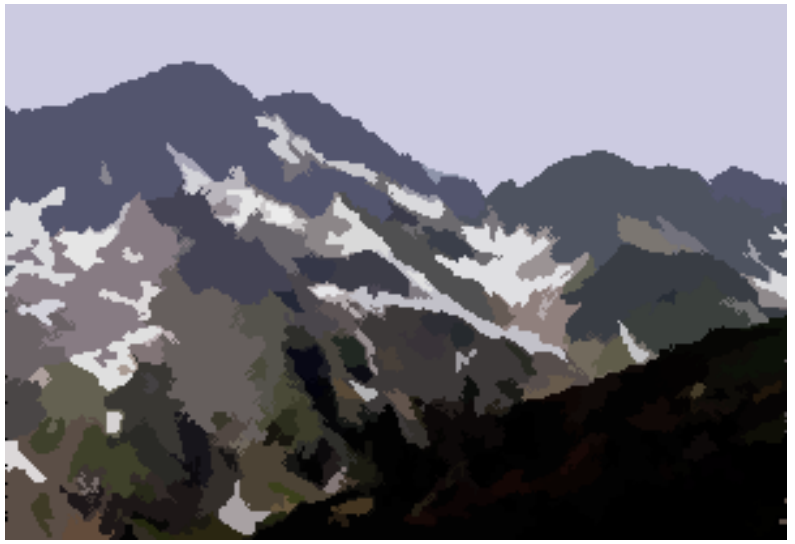
Mean-Shift Clustering Steps





Results

Mean-Shift clustering (segmentation) results



Comaniciu and Meer 2002

Mean-Shift clustering Pros & Cons

Pros:

- Good general-practice segmentation
- Flexible in number and shape of regions
- Robust to outliers

Cons:

- Have to choose kernel size in advance
- Not suitable for high-dimensional features

Usage:

- Over segmentation
- Multiple segmentations
- Tracking, clustering, filtering applications

Classical Vs. ML Segmentation

Aspect	Traditional Image Segmentation	ML/DL Image Segmentation
Approach	Rule-based algorithms	Data-driven models
Data Requirements	Low or no requirement for labeled data	High requirement for labeled data (for supervised learning)
Flexibility	Specific to the task and conditions	Highly adaptable to various tasks and conditions
Performance	Generally consistent for known scenarios	Can achieve state-of-the-art results, especially in complex scenarios
Robustness	Less robust to variation in data	More robust to variations due to learning from data
Scalability	Limited scalability	Highly scalable, especially with deep learning
Semantic Understanding	Typically lacks semantic understanding of the scene	Can perform semantic segmentation (classifying each pixel)
Real-time Processing	Often suitable for real-time applications	Real-time possible but may require optimized models
Interpretability	More interpretable due to rule-based nature	Less interpretable, particularly with DL models
Adaptation to New Data	Manual re-tuning required	Can be retrained or fine-tuned with new data
Example Methods	Thresholding, edge detection, region growing	Convolutional Neural Networks (CNNs), U-Net, Mask R-CNN

References

- Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.
- Computer Vision © James Hays, Brown University
- First Principles of Computer Vision, Shree Nayar, Computer Science Department, School of Engineering and Applied Sciences, Columbia University.
- Time & Frequency domain, a course on neuroscience time series analyses.
- Richard Szeliski, Computer Vision: Algorithms and Applications

NEXT:

Interest Points Detection

Practical examples.