Image Segmentation

Javier González Jiménez

Reference Books:

- *Computer Vision: Algorithms and Applications*. Richard Szeliski. Springer. 2010.

http://szeliski.org/Book

Content

- Introduction
- Classical methods
  - Contour-based techniques
  - Thresholding
  - Clustering-based techniques
    - Region growing
    - K-Means
    - Expectation-Maximixation
    - Mean-Shift (not included)
- Deep NN

1. Introduction

Image segmentation consists of **splitting the image in regions** whose pixels have similar properties or meaning.

1. Similar properties: color, texture, localization, …

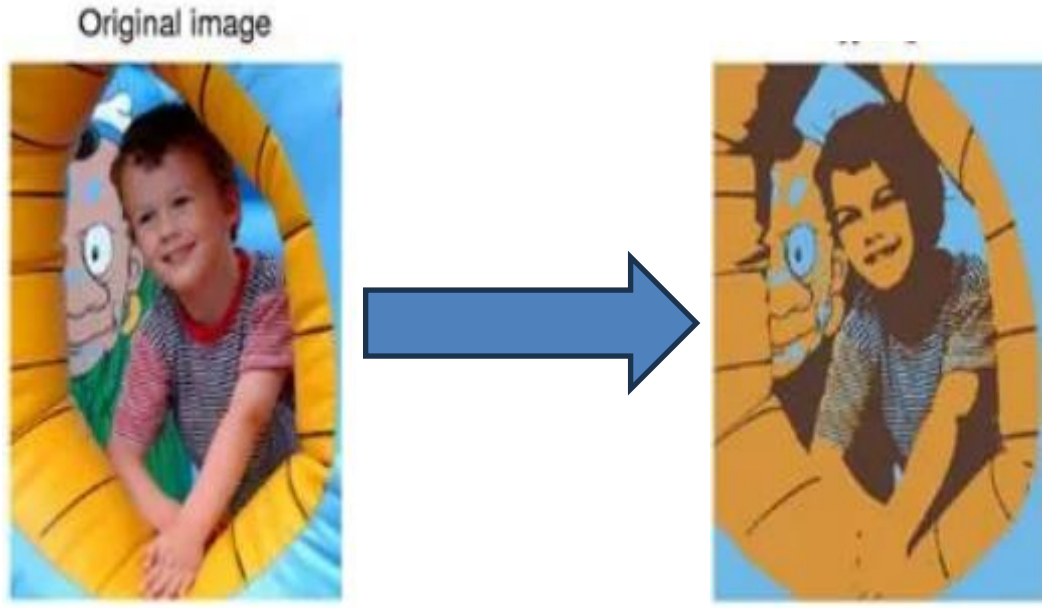


Image segmented in 3 regions whose pixels share some properties

Addressed with Classical segmentation methods and some DNN models (SAM)

1. Introduction

2. Similar meaning

Semantic segmentation

Pixels from the same object category must be in the same segmented región,
e.g. blue color for chairs

Instance segmentation

Identify the shape of each object instance,
e.g. each color represent a different chair

Addressed with *Deep NN* methods

Approaches

- *Classical methods*:
  - Based on **predefined algorithms** and heuristics
  - **Exploit pixel similarities.** Not semantic meaning
  - Useful for non-complex images, typically in **controlled scenarios** (e.g. industry, medicine)
- *Deep NN methods:*
  - Leverage **large datasets and complex NN** to learn meaningful representations directly from data.
  - Superior performance on a wide range of tasks, particularly in **non-controlled scenarios**.
  - More **computationally demanding**
  - Perform **segmentation + object recognition at once**

2.1 Classical methods: Contour-based techniques

Atempt to indentify the image regions by detecting their contours

Image contours: edge pixels that enclose a region of similar intensities

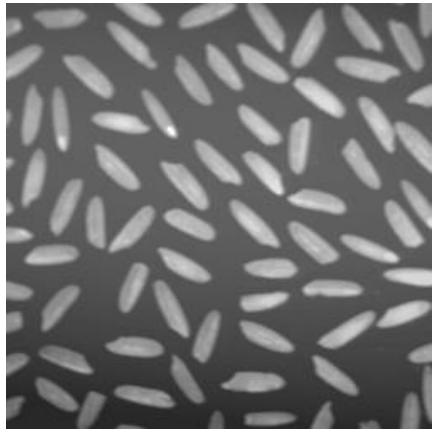
Two main approaches:

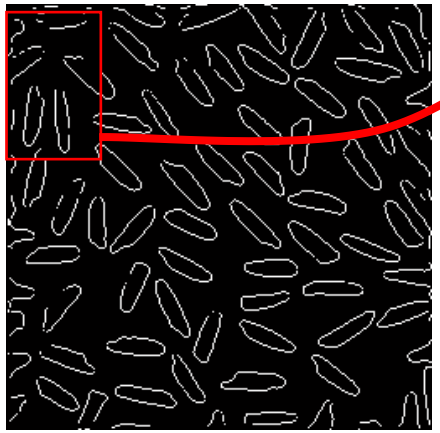
Local techniques → { LoG + zero crossing; Edge following → Canny operator }

Global techniques → Hough transform

Contour-based techniques

LoG + Zero crossing





Open contours

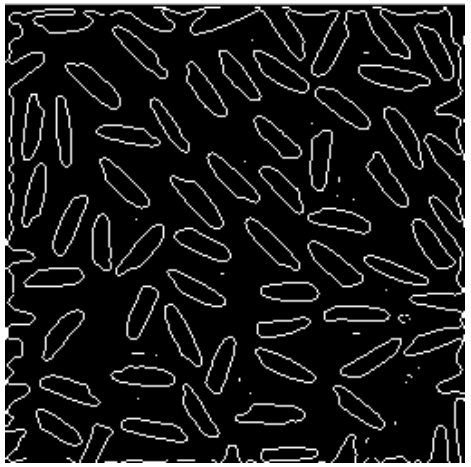
σ=2
th=0.007

MATLAB

```
I = imread('rice.tif');
%BW=edge(I,'log',thresh,sigma)
log = edge(I,'log',0.001,2);
figure, imshow(log)
```

σ=2

th=0.001

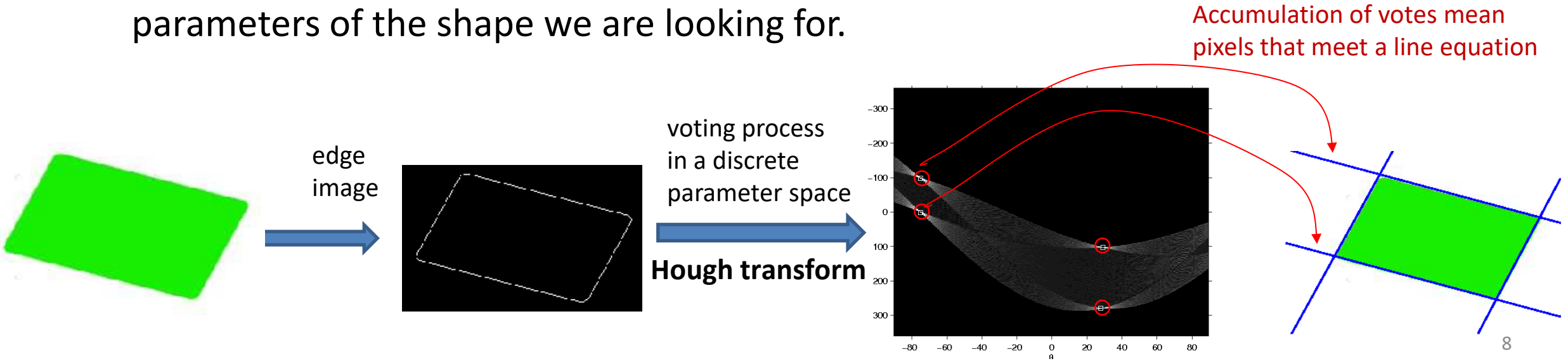


A simple method for region segmentation when a region has **similar pixel intensities**.

Contour-based techniques

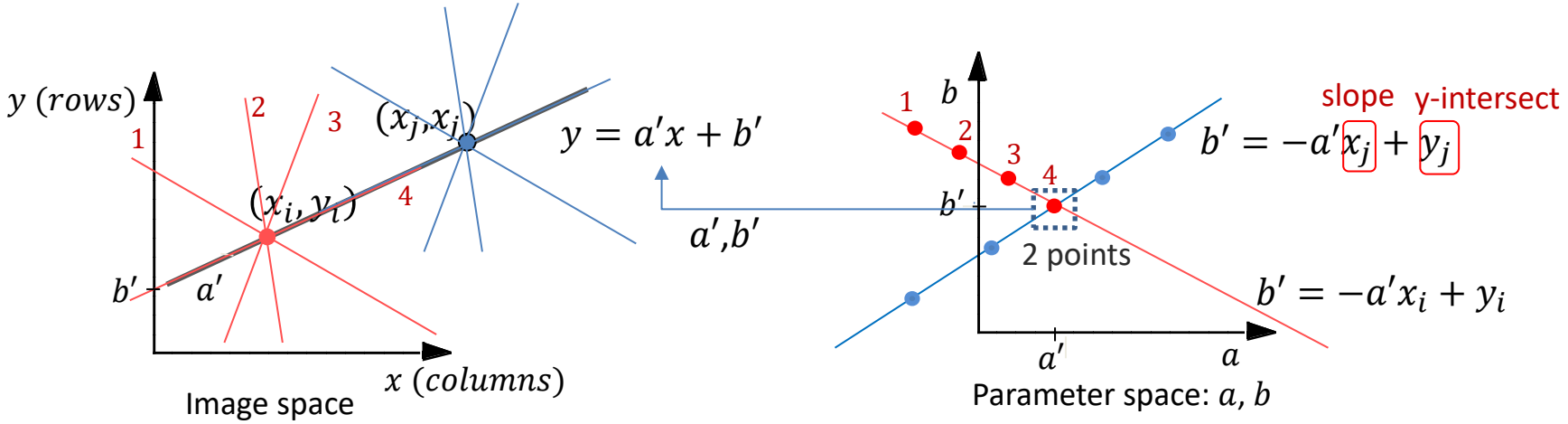
Hough transform

- Algorithm patented by Paul Hough to recognize lines in photographs (Hough, 1962)
- Can detect any shape in the image
  - **Analytical form** (classic Hough): typically line, circle, ellipse
  - **Numerically described form** (generalized Hough): shape given by a table
- Based on a voting scheme: each point *(x,y)* of **an edge** image votes for all the parameters of the shape we are looking for.



Hough transform: Line detection

Basic (explicit) form: $y = ax + b$

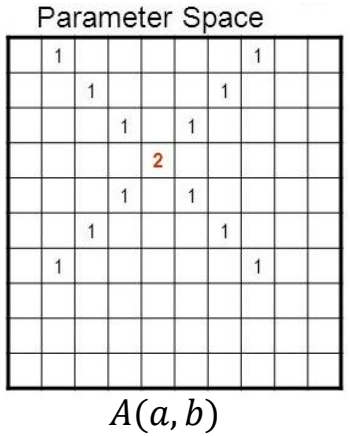


- Lines through points (x_i, y_i) are pairs (a,b) that defines a line in the (a, b)-parameter space

- Each image point (x_i, y_i) transforms to a line in the (a, b)-parameter space

$$(x_i, y_i) \rightarrow b = -ax_i + y_i$$

Implemented as a discrete accumulator



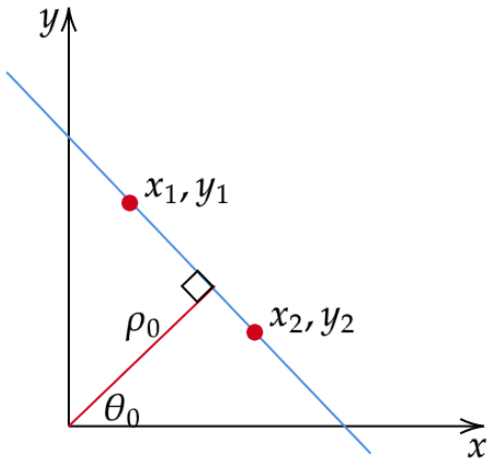
Algorithm

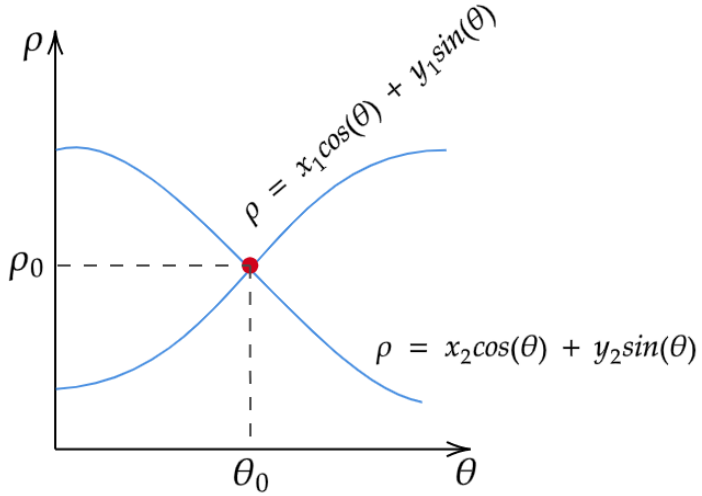
- Quantize parameter space (a, b)
- Create Accumulator array $A(a, b)$
- Set $A(a, b) = 0 \quad \forall a, b$
- For each (x_i, y_i) in the edge image
  For $a_k = 1\ to\ N$
  $b_k = -a_k x_i + y_i$
  $A(a_k, b_k) = A(a_k, b_k) + 1$
- Find local maxima in $A(a, b)$

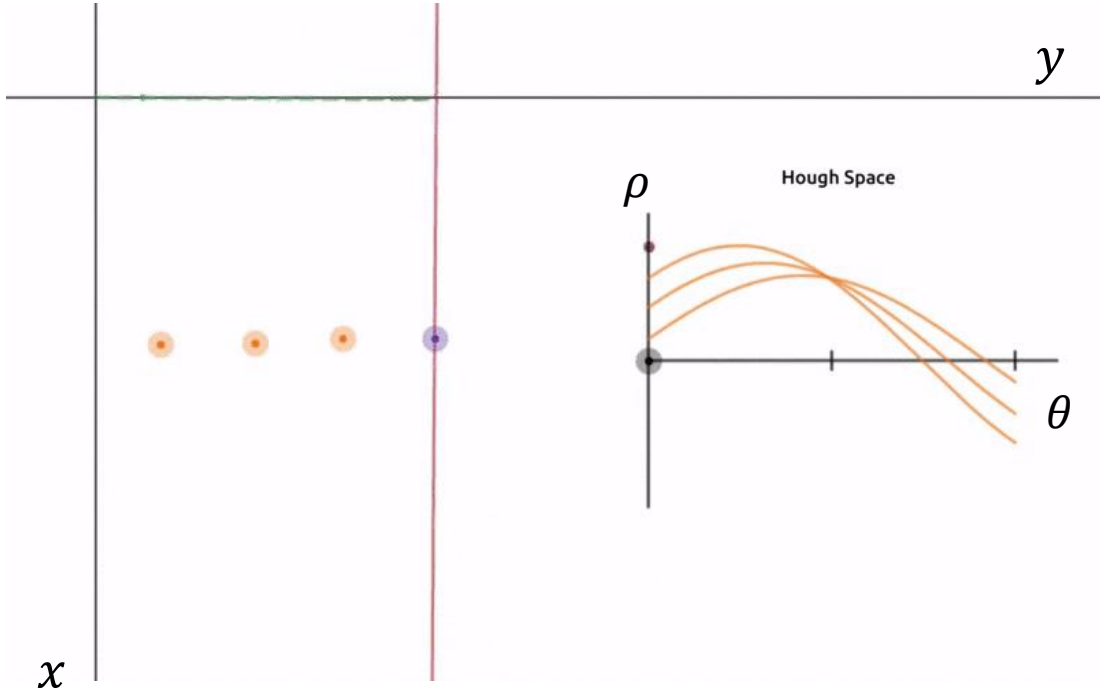
Hough transform: Line detection

The explicit form $y = ax + b$ presents problems for vertical lines ($a = \infty$)

→ Better: Normal form: $\rho = xcos\theta + ysin\theta$





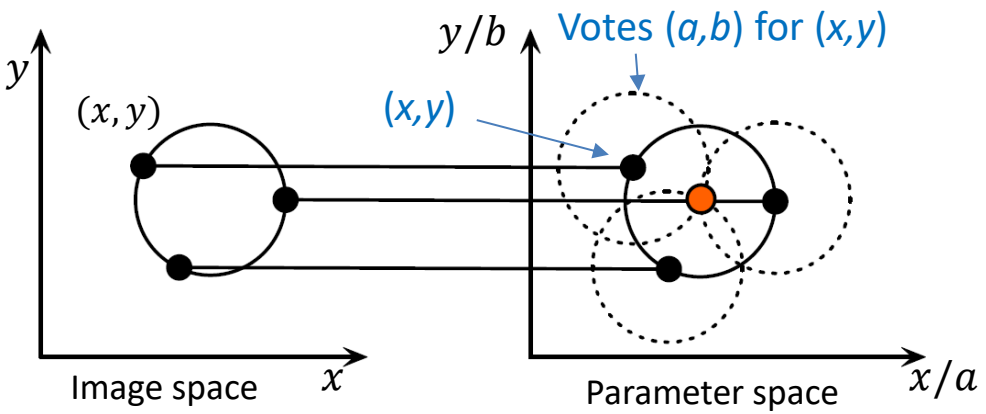


- The set of lines that pass through an image point (x, y) describes a sinusoidal curve $\rho = f_{xy}(\theta) = xcos\theta + ysin\theta$ in the parameter space $\rho - \theta$
- The line that passes through the 4 points is detected with the 4 intersections (4 votes)

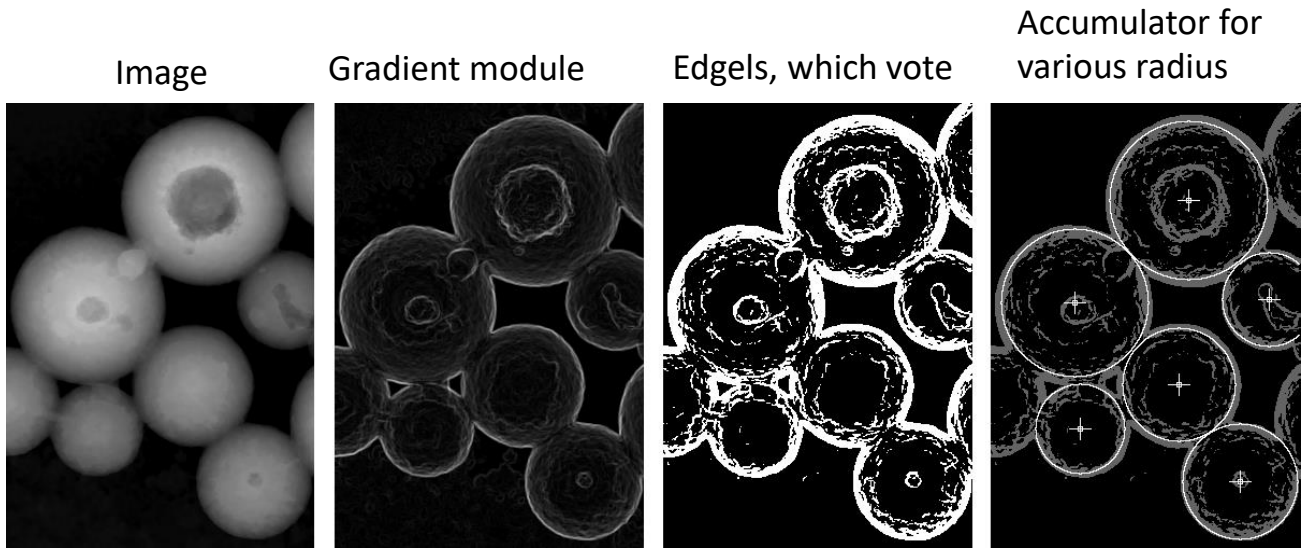
Hough transform: circle detection

Circle equation: $r^2 = (x-a)^2 + (y-b)^2$

A circle of known radius *r* has two parameters: the coordinates of the center *(a,b)*



- Note that in the $a-b$ space the expression: $r^2 = (x-a)^2 + (y-b)^2$ can be seen as a circle at center (x, y)
- Each point (x, y) in image space votes for a circle (dashed) of candidate centers (a, b) in the parameter space.
- Ideally, all the generated circles in the parameter space intersect at certain (a, b) which is the unknown center of the circle in the image.



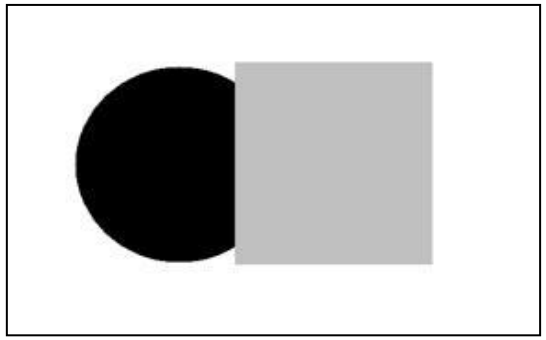
Cells with máximum votes indicated by +

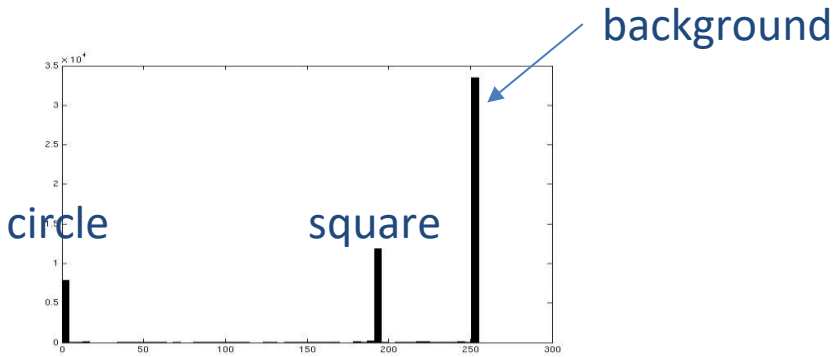
2.2 Classical methods: Thresholding

Assumption: different objects present different intensities

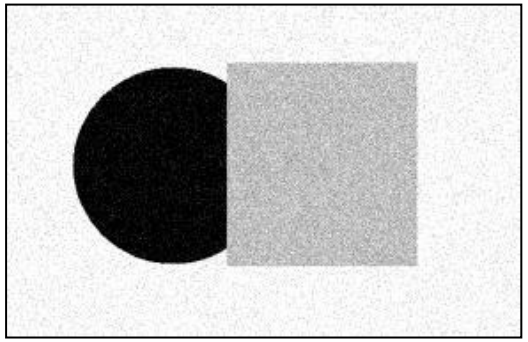
Application: Segment objects from the background (then, we considered two regions in the image)

Ideal image of 3 objects

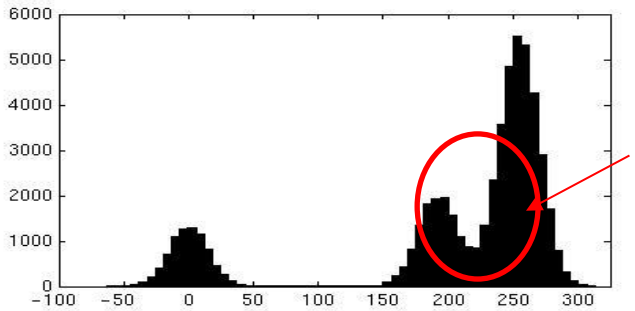




Noisy image



Image

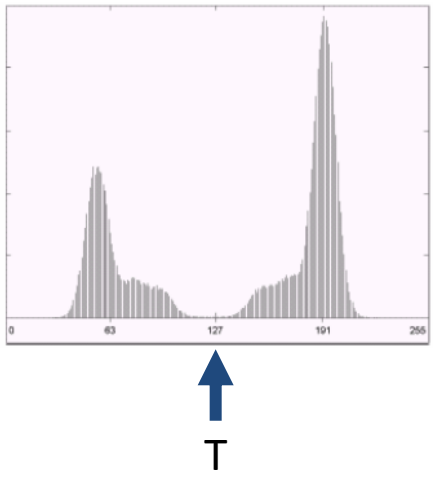


Histogram

The assumption does not hold: some background pixels are darker than some pixels of the square

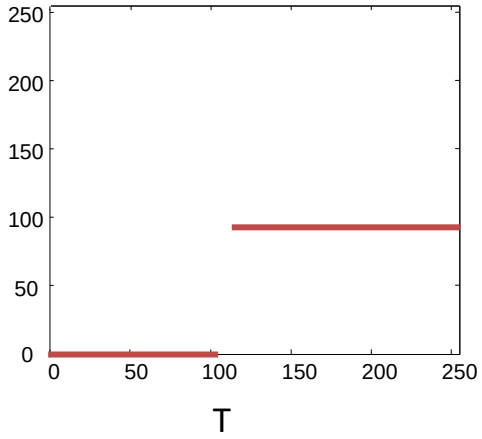
2.2 Classical methods: Thresholding

Example:



Objetive: find the threshold T that best separates pixels of different objects

Applying a binarization LUT



Segmented image

The threshold may be **global** or **adaptive**:

Global threshold: Same threshold for all pixels

- Applyied through image binarization

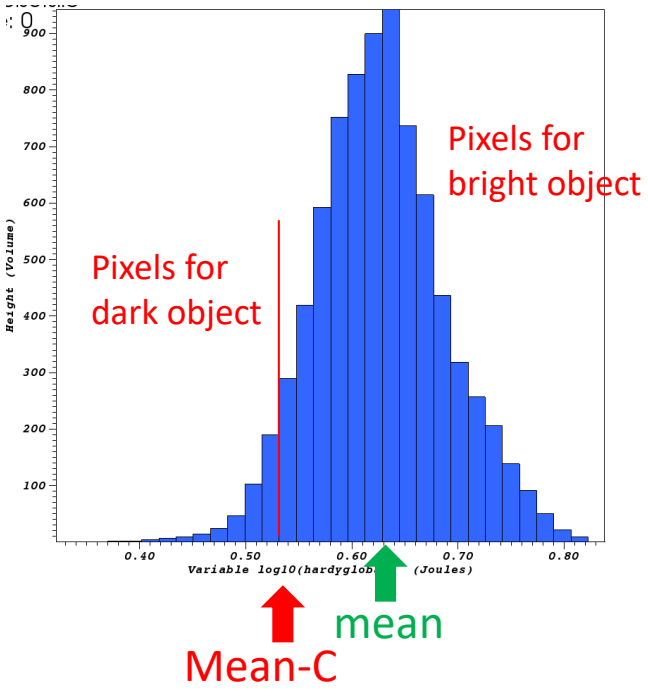
Adaptive threshold: a different threshold is selected for each local region (a window) around each pixel.

- gives better results for images with varying illumination.

Example in OpenCV

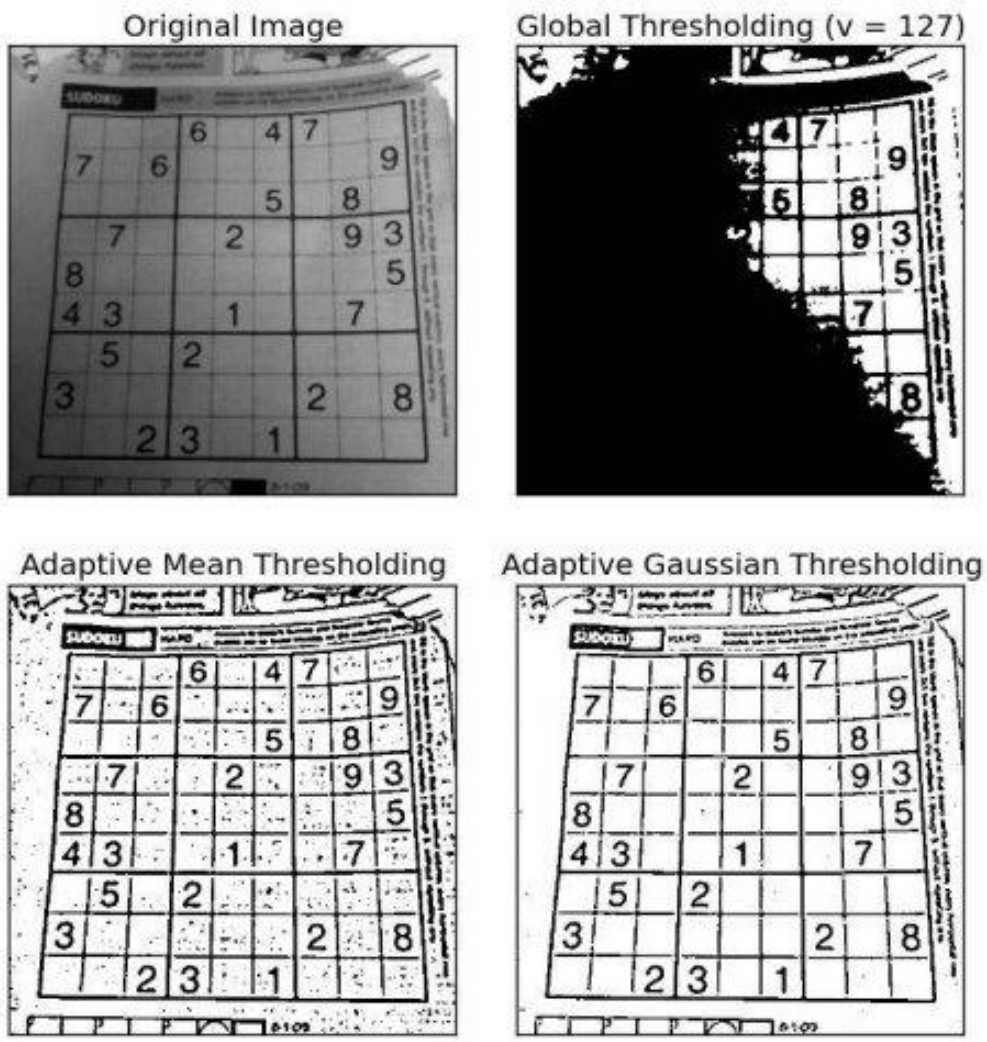
- **cv.ADAPTIVE_THRESH_MEAN_C**: The threshold value is the mean of the neighborhood area minus a constant **C**.
- **cv.ADAPTIVE_THRESH_GAUSSIAN_C**: The threshold value is a gaussian-weighted mean of the neighborhood values minus a constant **C**.

Constant **C** to control the number of pixels in each region.



The threshold may **global** or **adaptive. Example**

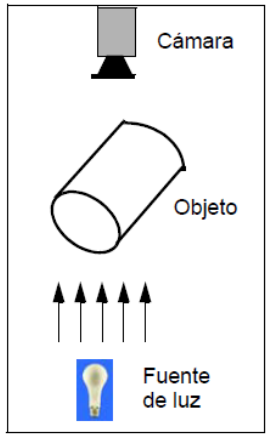
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('sudoku.png',0)
img = cv.medianBlur(img,5)
ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C,\
cv.THRESH_BINARY,11,2)
th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,\
cv.THRESH_BINARY,11,2)
titles = ['Original Image', 'Global Thresholding (v = 127)',
'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]
for i in range(4):
plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
plt.title(titles[i])
plt.xticks([]),plt.yticks([])
plt.show()
```



2.2 Classical methods: Thresholding

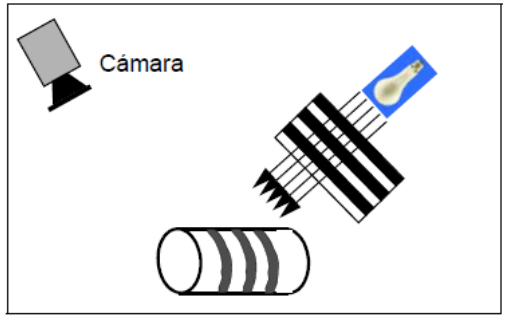
Thresholding requires a particular scene illumination: Back-projection

Types of illumination

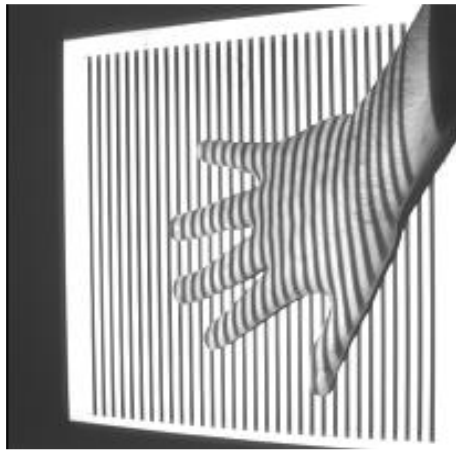


Back-projection

Used for binarization

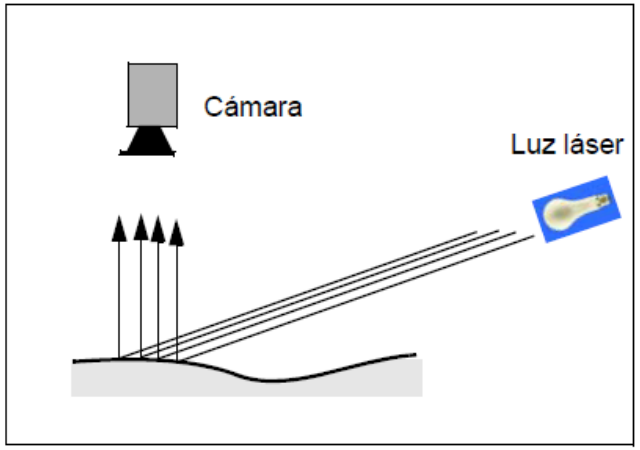


Structural illumination

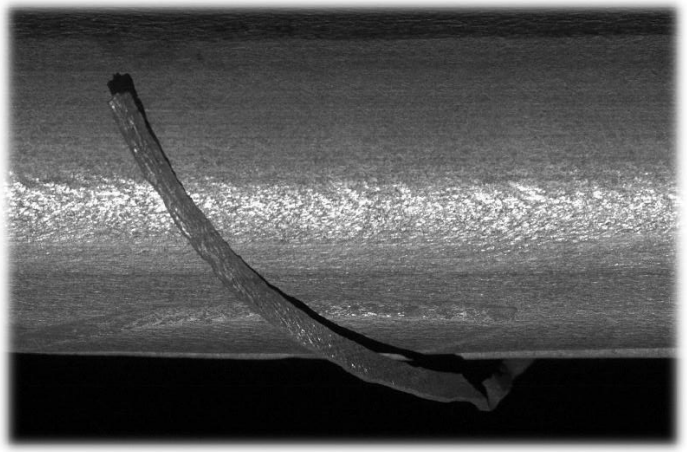


Used for 3D reconstruction

Types of scene illumination

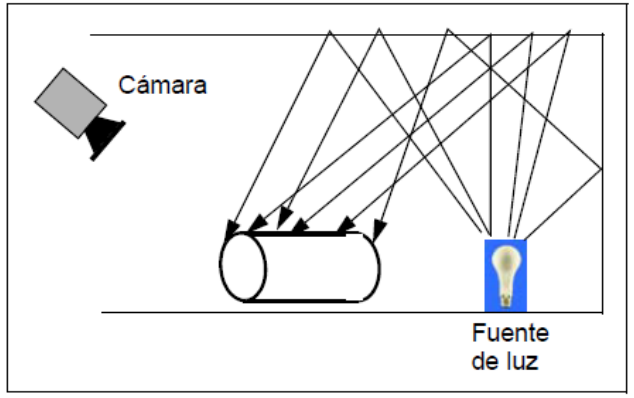


Directional illumination



© National Instruments

Used for surface inspection



Diffuse illumination

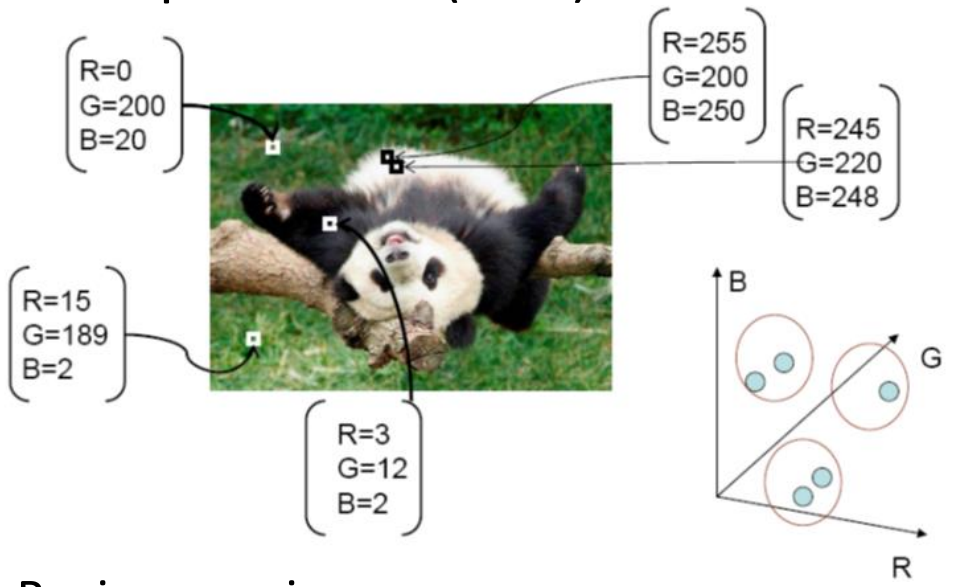
Used for color segmentation

2.3 Classical methods: Clustering-based techniques

Idea: Group together pixels that are similar according to some properties (*Clustering problem)*

Properties to decide on similarity: intensity, texture, color, pixel location, etc.

Example: property is the pixel color (RGB)



Each pixel is a point in the RGB space

Similar pixels are close to each other in this space

Techniques:

- Region growing
- K-Means
- Expectation-Maximixation
- Mean-Shift

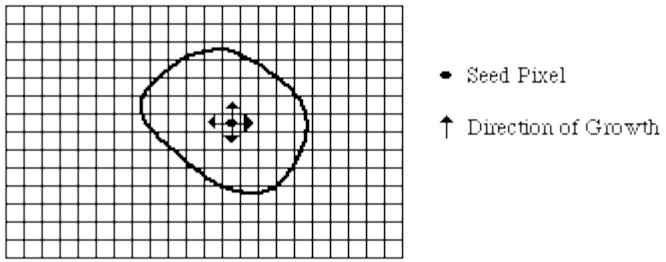
Source: K. Grauman

2.3 Classical methods: Clustering-based techniques

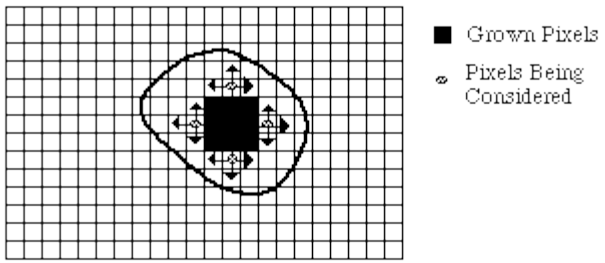
Region growing

- Start from a set of seed-pixels which are recursively grown with neighbouring pixels that show similar properties

- If *n* seed-pixels are used, the algorithm ends up with *n* regions, at most (some can be merged)



(a) Start of Growing a Region



(b) Growing Process After a Few Iterations

Region growing. Example

Similarity criterion: intensity difference less than 3

0	0	5	6	7
1	1	5	8	7
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

0	0	5	6	7
1	1	5	8	7
0	(1)	6	(7)	7
2	0	7	6	6
0	1	5	6	5

0	0	5	6	7
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
0	1	5	6	5

a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

Similarity criterion: intensity difference less than 8

0	0	5	6	7
1	1	5	8	7
0	(1)	6	(7)	7
2	0	7	6	6
0	1	5	6	5

0	0	5	6	7
a	a	a	b	b
a	a	a	b	b
a	a	a	b	b
0	1	5	6	5

a	a	a	a	b
a	a	a	b	b
a	a	a	b	b
a	a	a	b	b
a	a	a	a	b

0	0	5	6	7
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
0	1	5	6	5

a	b	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	b	b	b	b

Result may depend on the implementation:
if the growing process starts from seed 7

2.3 Classical methods: Clustering-based techniques

Region growing

Key decisions:

1. How many seed pixels
   - Typically, the number of objects we are looking for.
   - Merging of connected regions during the process is possible
2. Where to place the seed pixels
   - we must apply any knowledge of the expected regions, if none, at random
3. How to select the similarity criterion to add new pixels
   - ***Texture***: Images of good resolution are needed.
   - ***Color/Intensity***: We can take into account the (dynamic) mean and standard deviation of the current region.
   - ***Expected Shape***: We can weight particular directions that more likely fit the expected object shape

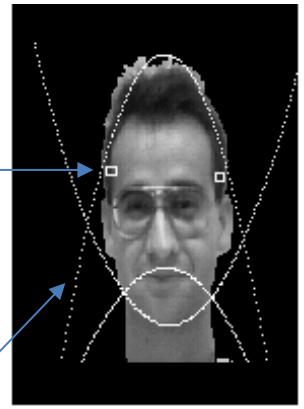
Region growing

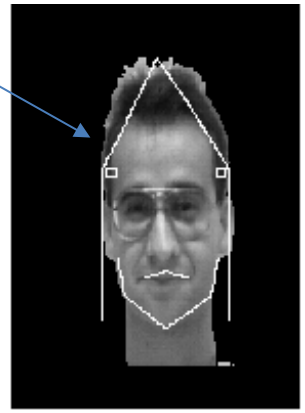
Example: hair segmentation of a previously segmented head

Seeds placed at specific pixels (detected by a previous algorithm)

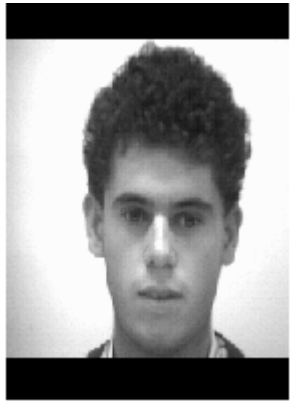
Expected shapes of the hair used as property

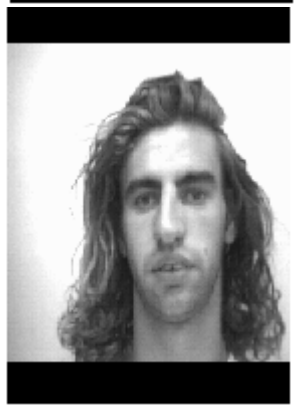
Approximated by these polygons



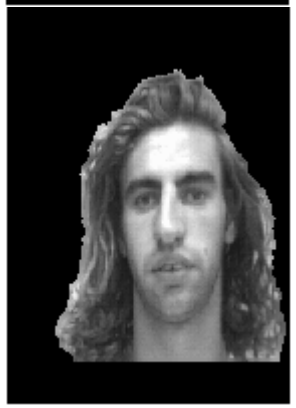


Original image





Segmentad head (given)



Segmented hair

Property used here: hair pixels must be close to line segments (polygonal shape) and darker than the face pixels

2.3 Classical methods: Clustering-based techniques

K-means

A clustering technique: Given a set of data elements X, make K clusters out of them

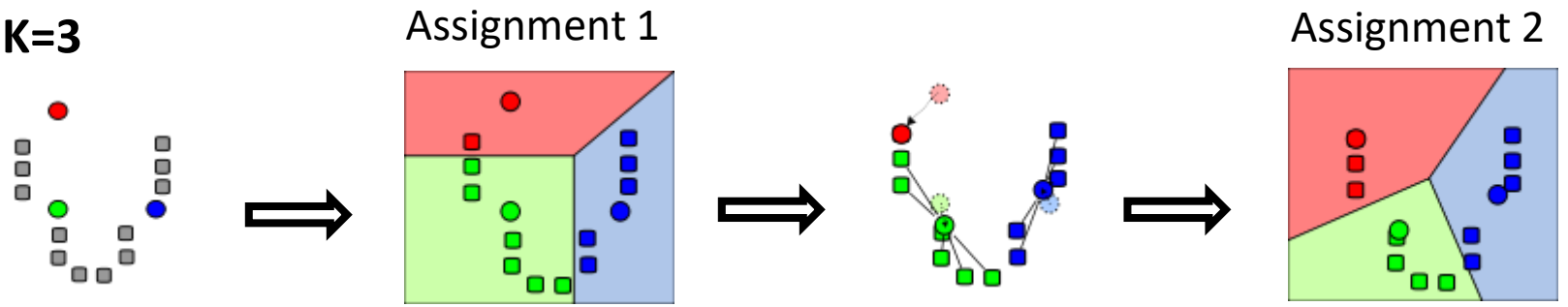
How: Find the K **nearest cluster centers** m_j that minimizes the sum of squared Euclidean distances (D) between points $X = \{x_i\}$ and m_j

$$\arg\min_{\{m_j\}} D(X, \{m_j\}) = \sum_{j=1}^{K} \sum_{i=1}^{M} a_{ij}(x_i - m_j)^2 \qquad a_{ij} = \begin{cases} 1 & x_i \text{ is assigned to cluster } j \\ 0 & \text{otherwise} \end{cases}$$

General algorithm:

1. Pick K **initial means** representing the clusters (not necessary elements of the set)
2. **Assign** each element **to the closest mean**, so creating new clusters (solve for a_{ij})
3. Compute the new means m_j for each cluster j
4. Repeat steps 2 and 3 until convergence

Example K=3



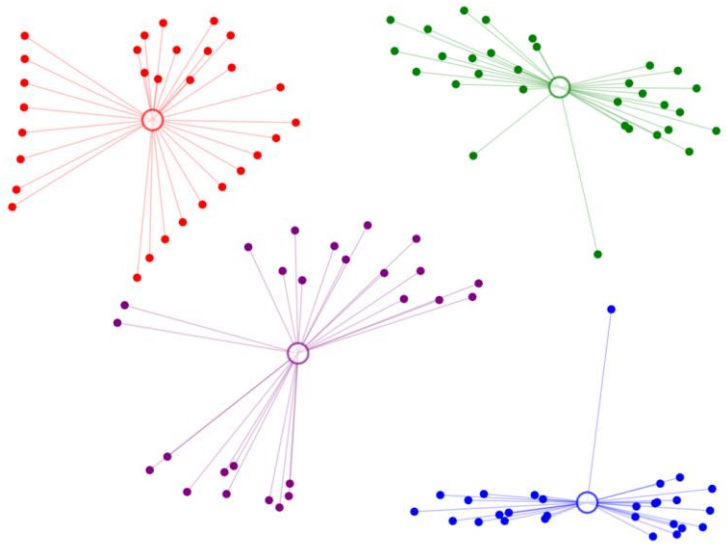
(Wikipedia)

2.3 Classical methods: Clustering-based techniques

K-means

Similar to region growing, but …

- All the pixels in the image are classified (assigned to a region) at each step, not just the neighbours
- The algorithm stops when the centers of the regions do not move (in region growing the stop condition is “there are no pixels to add”)



Interactive illustration:
https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

2.3 Classical methods: Clustering-based techniques

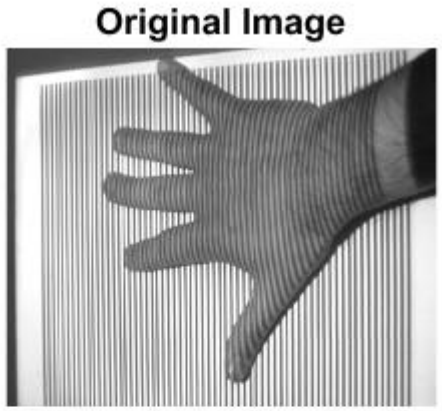
K-means for segmentation

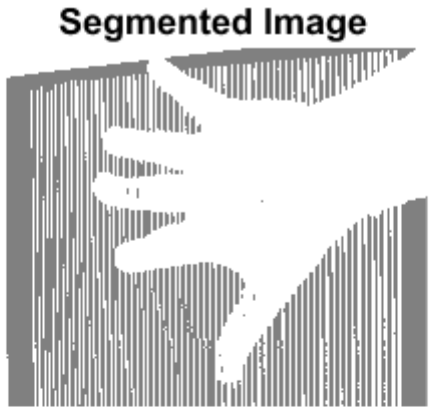
- Each **pixel** of the image is represented by a **feature vector** (e.g., color, intensity, texture, etc.)
- Each **region** is represented by the **mean of the feature vector** of the pixels in it
- In the feature space we need to define a **distance between vectors** (e.g. Euclidean distance)

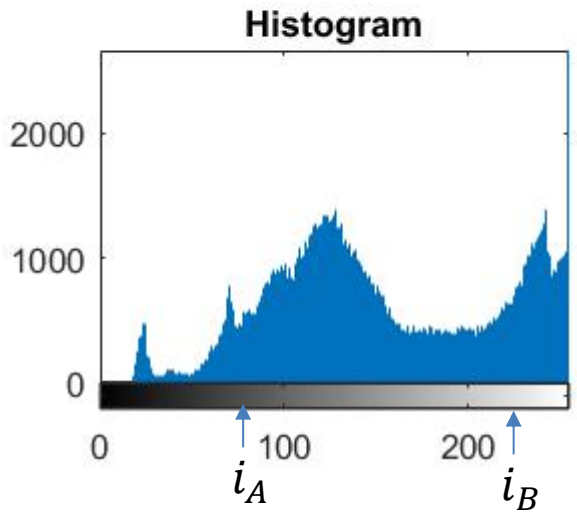
Algorithm

1. Select K pixels in the image (manually, at random, with some heuristic, ...) which will represent K regions
2. Assign each pixel in the image to the more similar region (the closest one according to the adopted distance)
3. Update the mean feature vector of the regions
4. Repeat 2 and 3 until no image pixel changes from one region to another (i.e. region centers do not change) → CONVERGENCE Criterium
5. Merge connected regions if similar

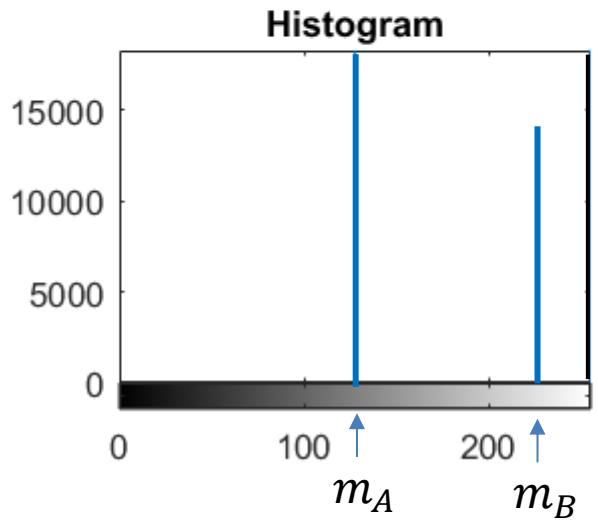
K-means example: 2-means using intensity as feature vector (dimension 1)







2 initial means



Final means

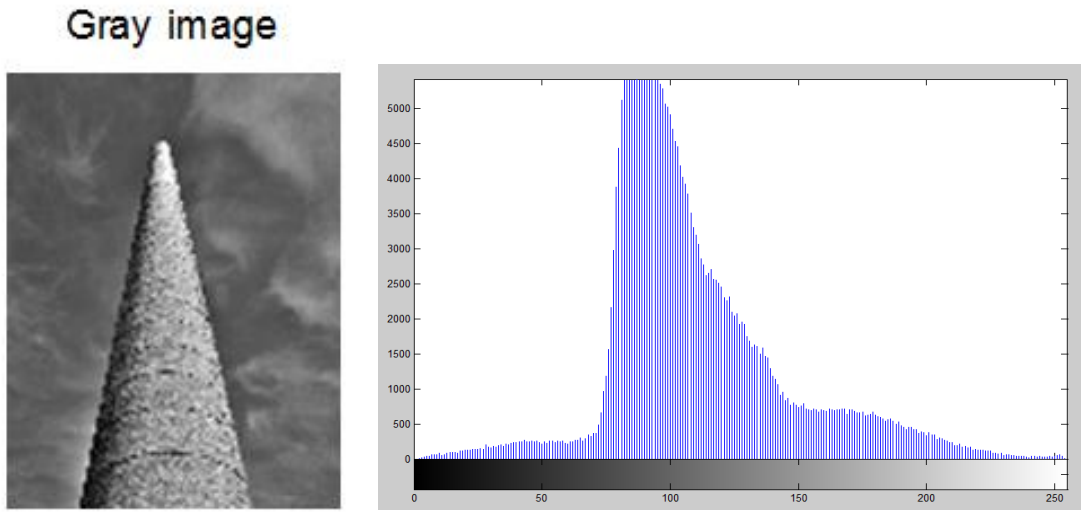
Algorithm

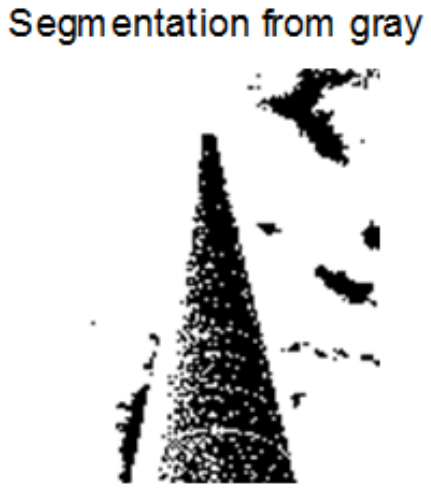
1. Choose 2 initial means in the histogram: $m_A = i_A$ and $m_B = i_B$
2. Compute distance of each intensity i to the means: $\mathrm{d}(i, m_A)$, $\mathrm{d}(i, m_B)$
3. Assign intensities to the closest cluster. We have 2 regions:
   - $A = \{i: \mathrm{d}(i, m_A) \leq \mathrm{d}(i, m_B)\}$
   - $B = \{i: \mathrm{d}(i, m_A) > \mathrm{d}(i, m_B)\}$
4. Compute new means:
   - $m_A = \frac{1}{N_A} \sum_k A(k)$
   - $m_B = \frac{1}{N_B} \sum_k B(k)$
5. Go to 2 if m_A or m_B is different from the previous one. Otherwise stop (Convergence)

Equivalent to thresholding with the moving threshold at the middle of the two means

K-means: Example of 2-means using intensity as feature vector

If the original histogram is not multimodal, the result is not good at all





- Only intensity is not descriptive enough to segment the regions
- But we can use a more descriptive feature vector, for example, color: x={R,G,B}

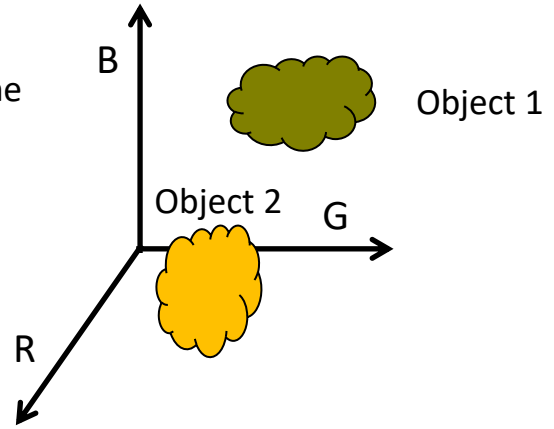
K-means. Example using color R,G,B as feature vector

M pixels in the image

$$\texttt{data} = \begin{bmatrix} R_1 & R_2 & \cdots & R_M \\ G_1 & G_2 & \cdots & G_M \\ B_1 & B_2 & \cdots & B_M \end{bmatrix}$$

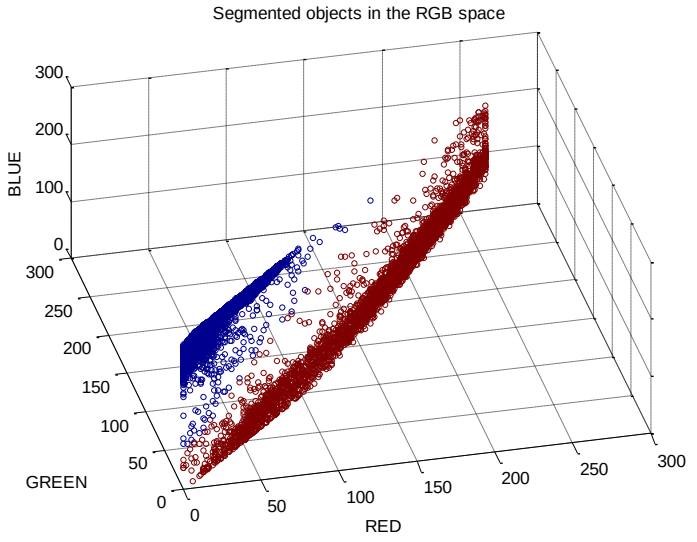
Feature vectors: R,G,B

2 clusters in the RGB space



Color image

Segmentation from color

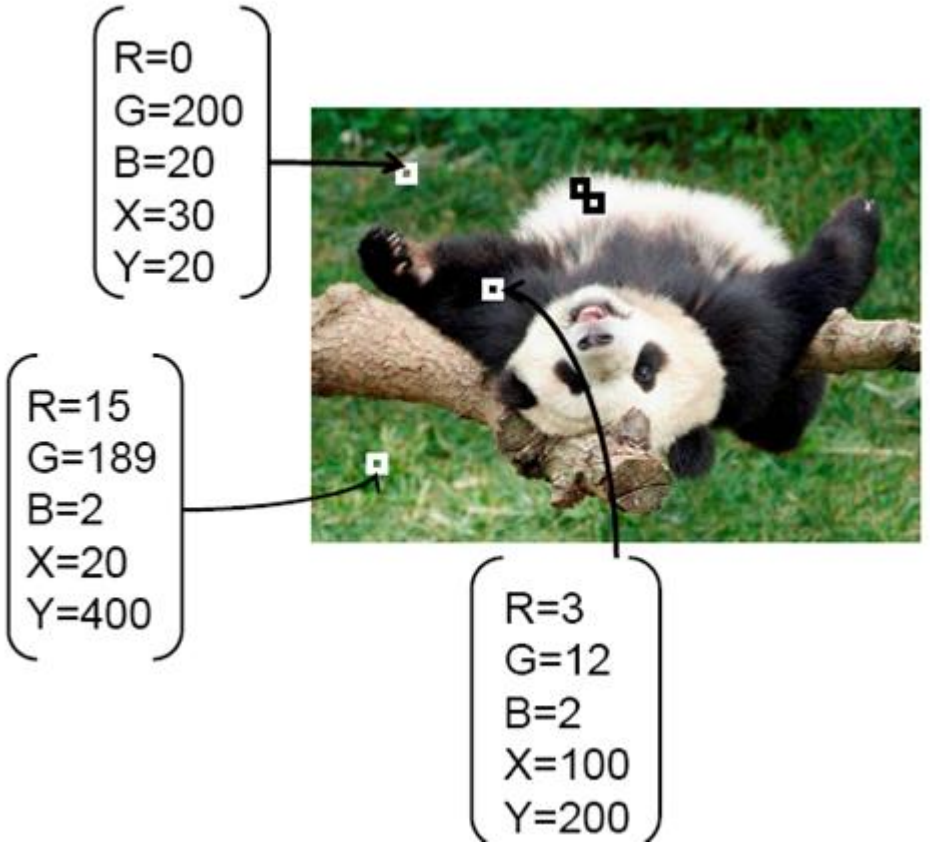


Notice: If the two initial RGB values (means) are from the sky the result will not be correct.

K-means

Still, color RGB may be not enough for a good segmentation ⇨ We need to add some spatial data, eg. (r,g,b,x,y)

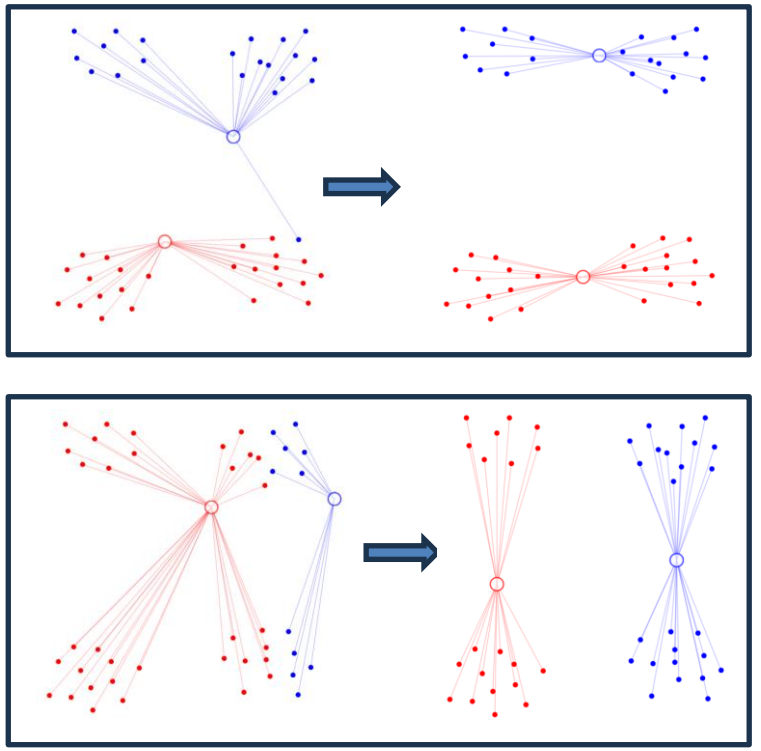
Pixel position

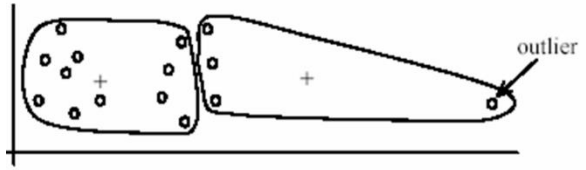


K-means

- Pros
  - Simple implementation
  - Always converge to a local minima (but no guarantee to reach the global minima)
- Cons
  - Fix K: Guess about the number of cluster needed
  - Depending on the initial means, the result may be different
  - Very sensible to outliers
  - The shape of the clusters in the feature space are assumed to be circular (because the use of the Euclidean distance)

Same data, different initializations





(A): Undesirable clusters

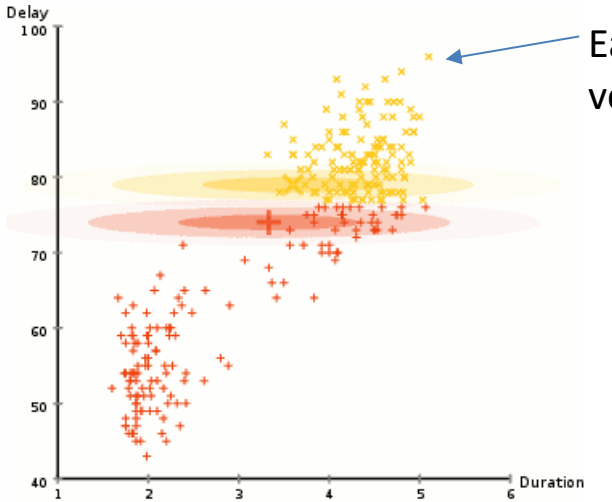
(B): Ideal clusters

2.3 Classical methods: Clustering-based techniques

Expectation – Maximization (EM)

- Is the generalization of the K-means method
- It's a *soft* clustering since it does not give "hard" clusters but the probability that an element (feature vector x of a pixel) belongs to each cluster C_j: $p(x|C_j)$
- Probabilities of the features are assumed to be Gaussians for each cluster: $p(x|C_j) \sim N(\mu_j, \Sigma_j)$
- At each iteration not only the mean is refined (as in K-means), but also the covariance matrix of the cluster, i.e. the shape of the cluster in the feature space

Example for a two-dimensional feature vector.



What is a posterior probability $P(C_j|\mathbf{x}_i)$?

- The probability that we have of an event C_j **after an evidence** (or observation) $\boldsymbol{x_i}$
- In segmentation, each C_j is one of the possible clusters (segmented objects)
- The tool for updating a probability once we gather an evidence is the Bayes Rule

Prior probability: our guess about the probability based on what we know now, before new data becomes available

Likelihood *of* C_j

Posterior probability

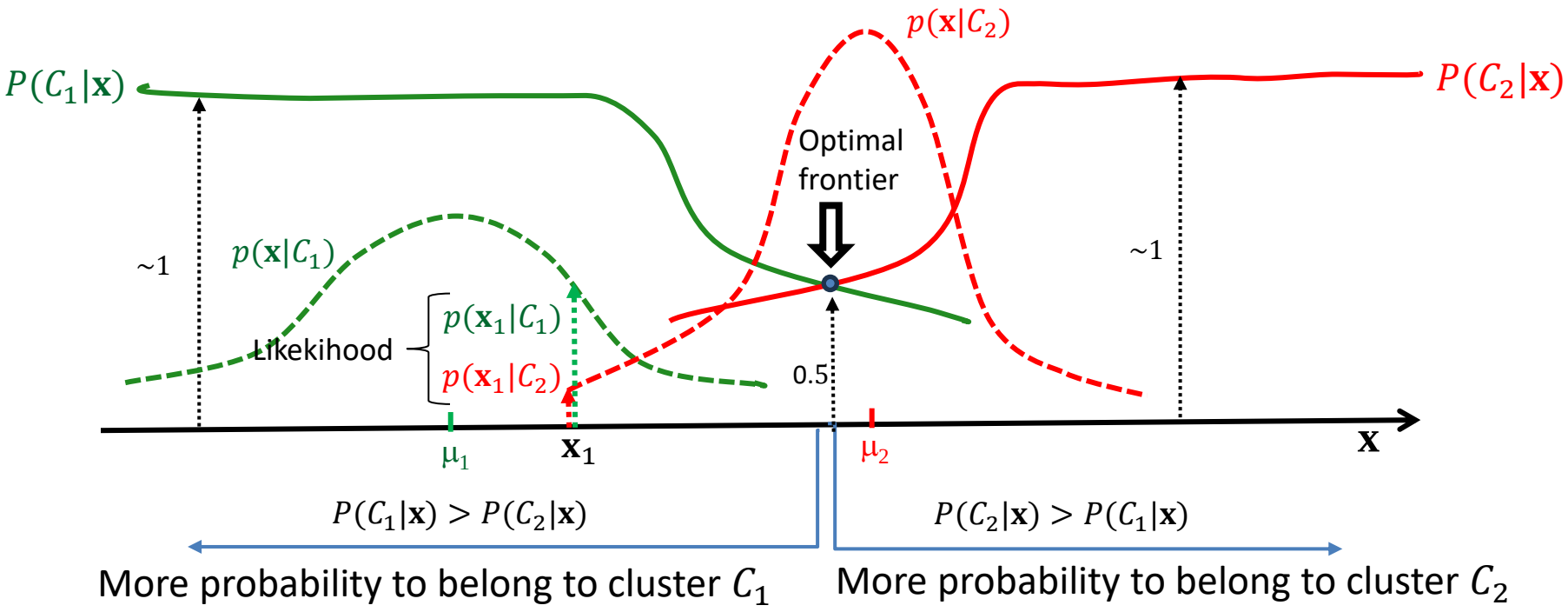
Bayes rule: $$P(C_j|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|C_j)P(C_j)}{p(\mathbf{x}_i)} = k\, p(\mathbf{x}_i|C_j)P(C_j)$$

Total probability (does not depend on C_j), $k = 1/p(\mathbf{x}_i)$

- $p(\mathbf{x}_i|C_j)$ is the **p**robability **d**ensity **f**unction (pdf) that a pixel from the cluster C_j has the feature vector $\mathbf{x}_i$
- In Bayes, $\mathbf{x}_i$ is **given (known),** then $p(\mathbf{x}_i|C_j)$ takes values (called *likelihood*) for each C_j. For example, for two clusters we will have: $p(\mathbf{x_i}|C_1)$ and $p(\mathbf{x_i}|C_2)$

Using the posterior probability to decide what cluster C_i a feature vector $\mathbf{x}$ belongs to

Example of two clusters C_1 C_2 :



$$P(C_1|\mathbf{x}) + P(C_2|\mathbf{x}) = 1$$

2.3 Classical methods: Clustering-based techniques

Expectation – Maximization

Normal (Gaussian) distribution: Feature vector $\mathbf{x}$ is vector of continuous random variables that follow the pdf:

$$p(\mathbf{x}|C_j) = \frac{1}{(2\pi)^{n/2}\left|\Sigma_j\right|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T \Sigma_j{}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)}$$

Feature vector of dimension *n*: $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_f \quad \cdots \quad x_n]^T$

Mean vector: $\boldsymbol{\mu} = [\mu_1 \quad \mu_2 \quad \cdots \quad \mu_f \quad \cdots \quad \mu_n]^T$ Gives the position of the Gaussian

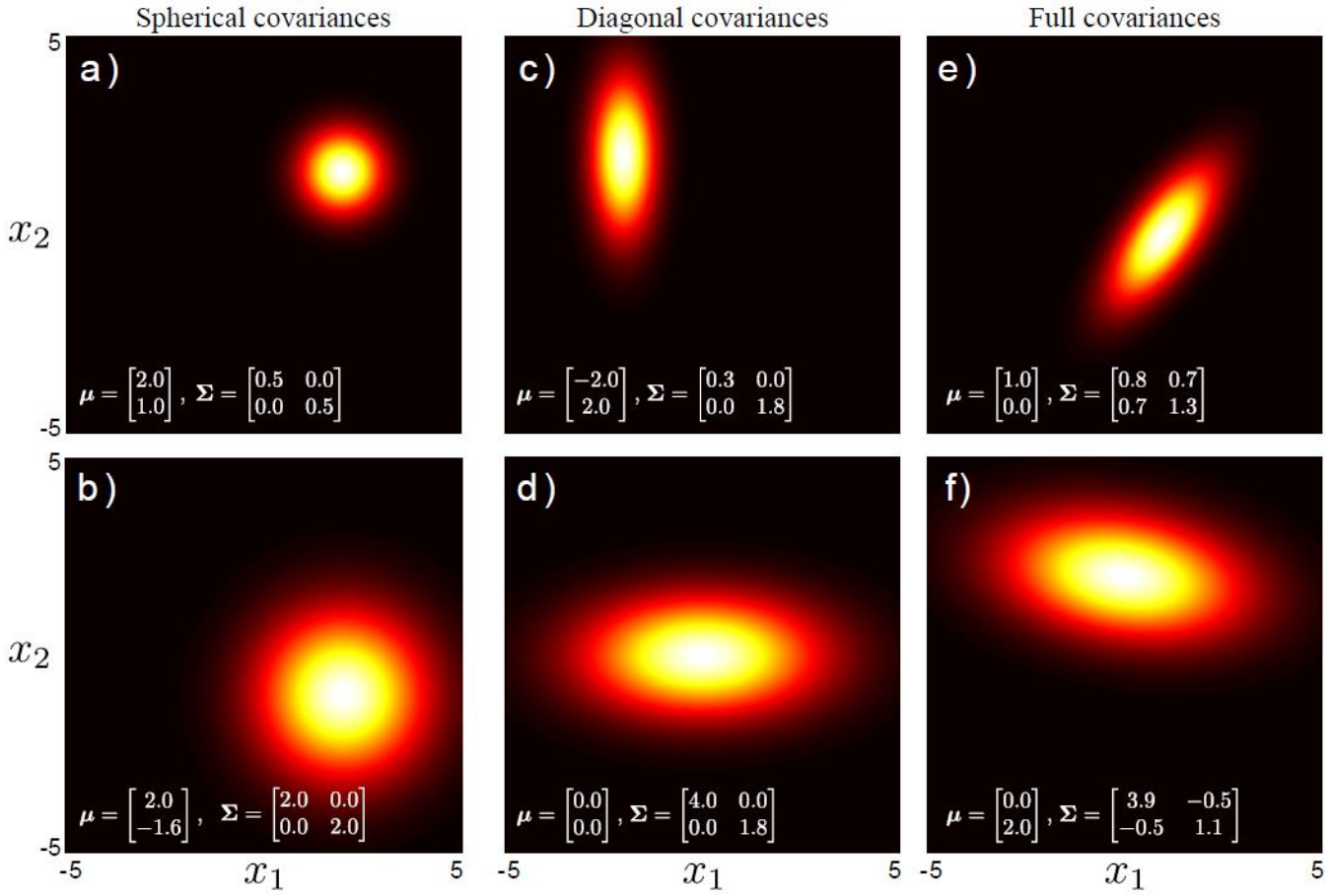
Covariance matrix: Gives the shape of the Gaussian

$$\Sigma = \mathrm{E}[(\mathbf{x}-\boldsymbol{\mu})\cdot(\mathbf{x}-\boldsymbol{\mu})^T] = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1f} & \cdots & \sigma_{1n} \\ \vdots & & \vdots & & \vdots \\ \sigma_{n1} & \cdots & \sigma_{nf} & \cdots & \sigma_{nn} \end{bmatrix} = \begin{bmatrix} \sigma_{11} & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & \sigma_{nn} \end{bmatrix}$$

If the features are independent to each other

Expectation – Maximization

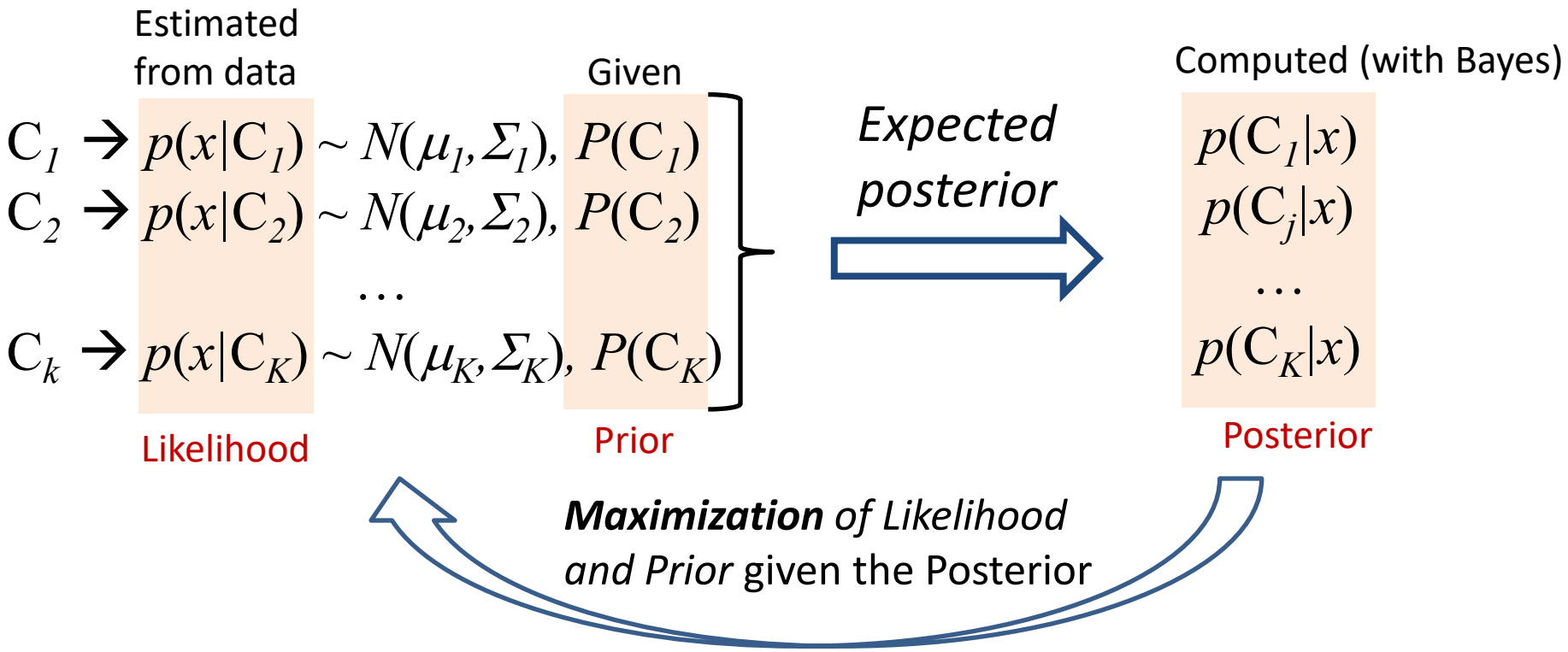
2D Gaussian pdf for different μ and Σ



From: Simon Prince; published by Cambridge University Press 2012.

Expectation – Maximization

Each *cluster* C_j has a different and unknown Gaussian pdf $p(x|C_j) \sim N(\mu_j, \Sigma_j)$



If same $P(C_j)$ and $\Sigma_j = \sigma^2 I$ (identical isotropic gaussians) for all the clusters, **EM is equivalent to K-means**

EM Algorithm:

Commonly, initialization from k-means

Inputs Data: $X = \{x_1, \dots, x_M\}$, **Initials:** $p(x_i|C_j) = N(x_i|\mu_j, \Sigma_j), p(C_j), j = 1, \dots, K$

Iteration steps (until convergence): when no change ocurrs in a complete iteration

Expectation Step: With the current $P(x_i|C_j)$ and $P(C_j)$ compute (via Bayes) the **expected probabilities** $P(C_j|x_i)$ (that x_i belongs to each cluster C_j)

$$P(C_j|x_i) = \frac{p(x_i|C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i|C_j) \cdot p(C_j)}{\sum_j p(x_i|C_j) \cdot p(C_j)}$$

Assign x_i to the cluster C_j with the highest probabilities $P(C_j|x_i)$

Maximization Step: Estimate $(\mu_j, \Sigma_j), p(C_j)$ that **maximize the Likelihood** $p(x|C_1)$
(Maximun Likelihood Estimation - MLE)

$$\mu_j = \frac{\sum_i p(C_j|x_i) \cdot x_i}{\sum_i p(C_j|x_i)} \qquad \Sigma_j = \frac{\sum_i p(C_j|x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j|x_i)}$$

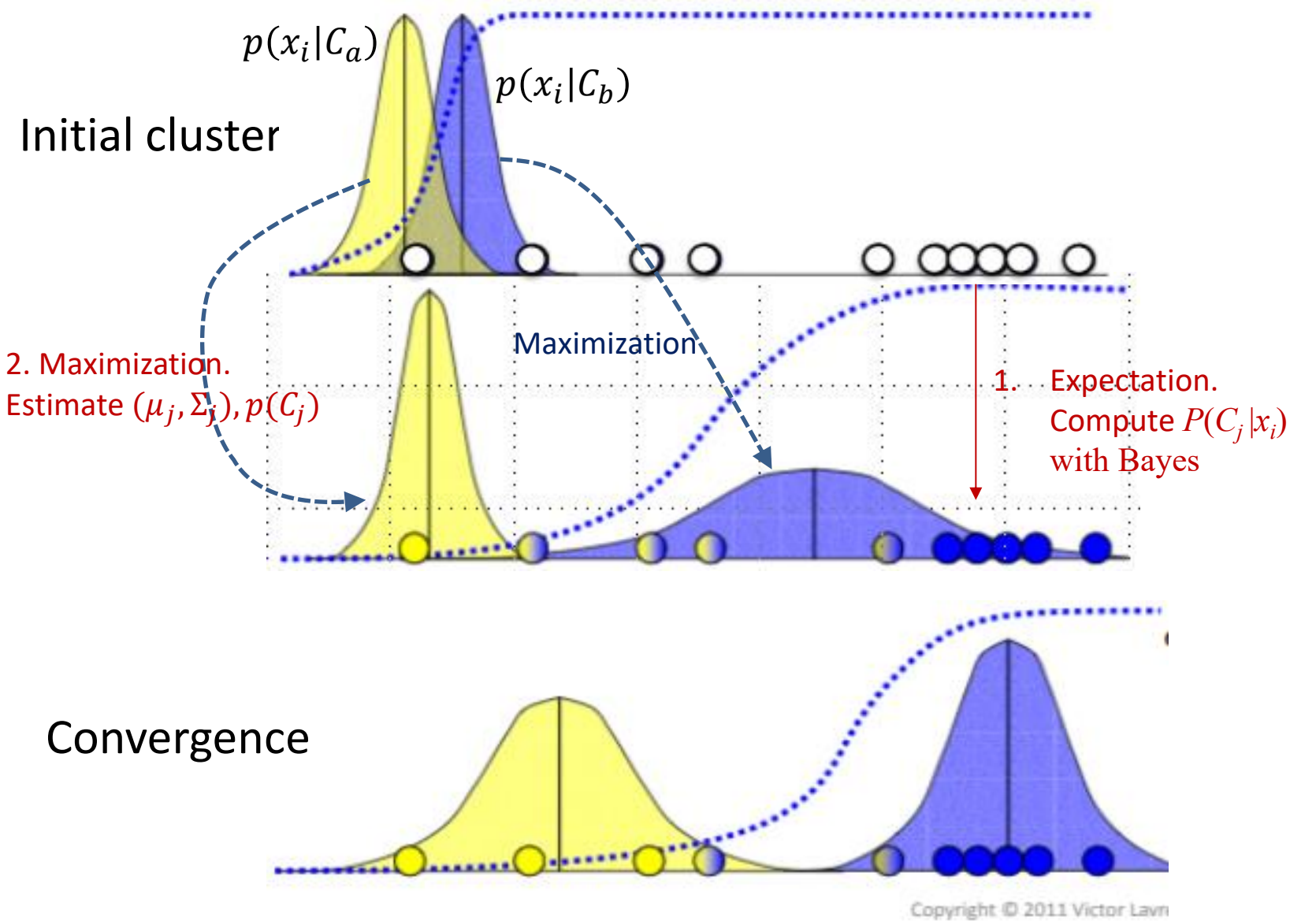
MLE of the mean and covariance

effective number of points associated with cluster j

$$p(C_j) = \sum_i p(C_j|x_i)\, p(x_i) = \frac{\sum_i p(C_j|x_i)}{K}$$

If no other information is available, each x_i is considered equal-probable $p(x_i) = 1/K$

Example 1D, two clusters: *a* and *b,* initial equal priors $p(C_a) = p(C_b)$



$$P(x_i|b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i-\mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b|x_i) = \frac{P(x_i|b)P(b)}{P(x_i|b)P(b)+P(x_i|a)P(a)}$$

$$a_i = P(a|x_i) = 1 - b_i$$

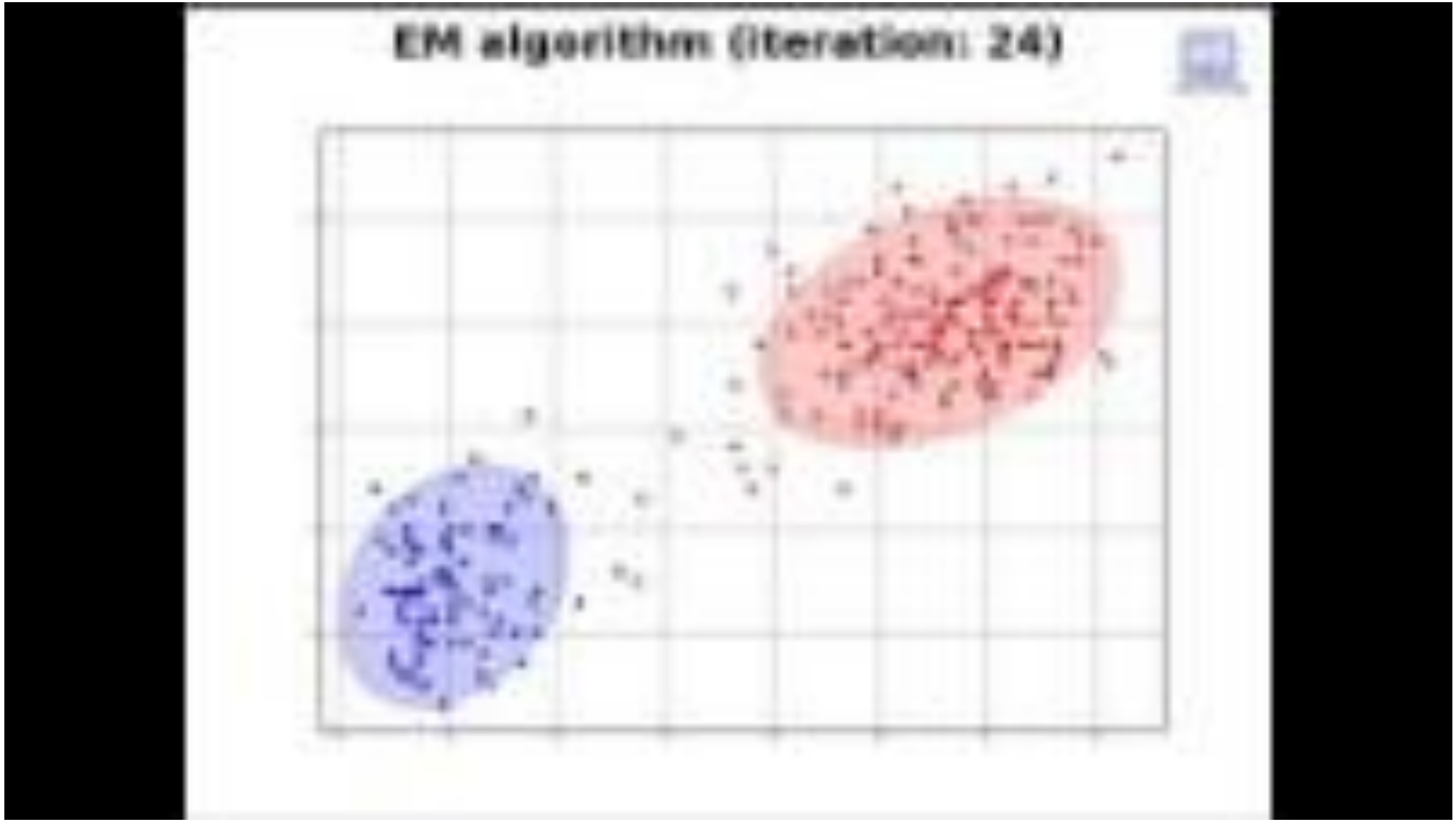
$$\mu_b = \frac{b_1x_1 + b_2x_2 + \ldots + b_nx_n}{b_1 + b_2 + \ldots + b_n}$$

$$\sigma_b^2 = \frac{b_1(x_1-\mu_1)^2 + \ldots + b_n(x_n-\mu_n)^2}{b_1 + b_2 + \ldots + b_n}$$

$$\mu_a = \frac{a_1x_1 + a_2x_2 + \ldots + a_nx_n}{a_1 + a_2 + \ldots + a_n}$$

$$\sigma_a^2 = \frac{a_1(x_1-\mu_1)^2 + \ldots + a_n(x_n-\mu_n)^2}{a_1 + a_2 + \ldots + a_n}$$

Example 2D, two clusters

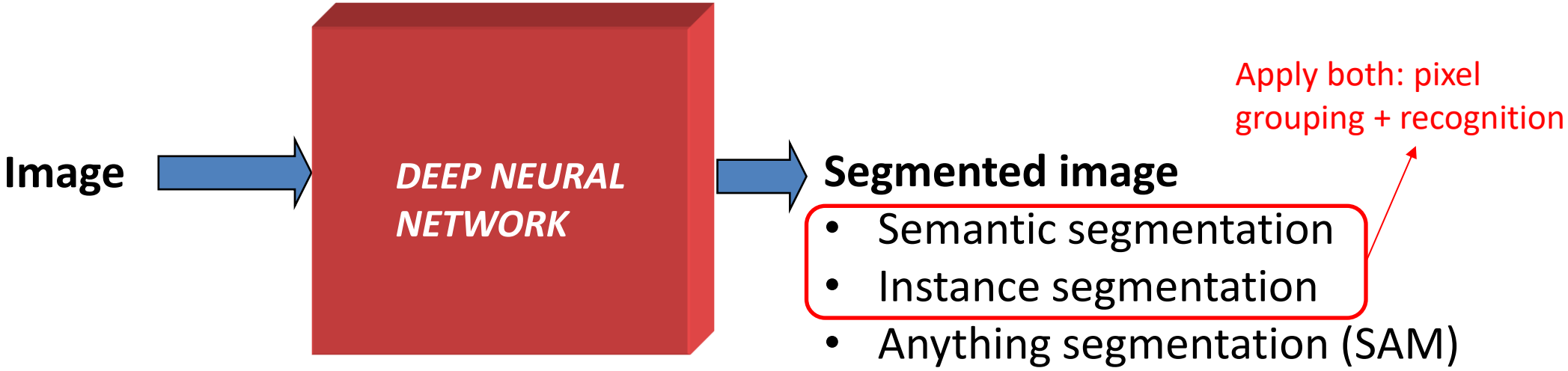


K-means *vs.* EM

	K-means	EM
Cluster Representation	Mean	Mean, variance
Cluster Initialization	Randomly select K means	Initialize K Gaussian distributions (μ_j, Σ_j) and $P(C_j)$
Expectation: Estimate the cluster of each data	Assign each point to the closest mean	Compute $P(C_j \mid x_i)$
Maximization: Re-estimate the cluster parameters	Compute means of current clusters	Compute new (μ_j, Σ_j), $P(C_j)$ for each cluster j

3. DNN-based segmentation

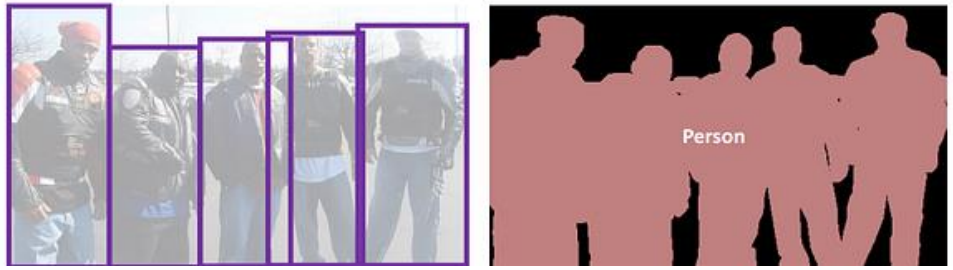
- Leverage **large datasets and complex NN** to learn what pixels must be in the same region
- **Not explicit model** to explain why pixels are grouped that way
- Segmented regions may come from similar pixels attributes (**SAM**), same object category (**semantic**) o same **instance** object.
- Superior performance on a wide range of tasks, particularly in **non-controlled scenarios**.
- More **computationally demanded**



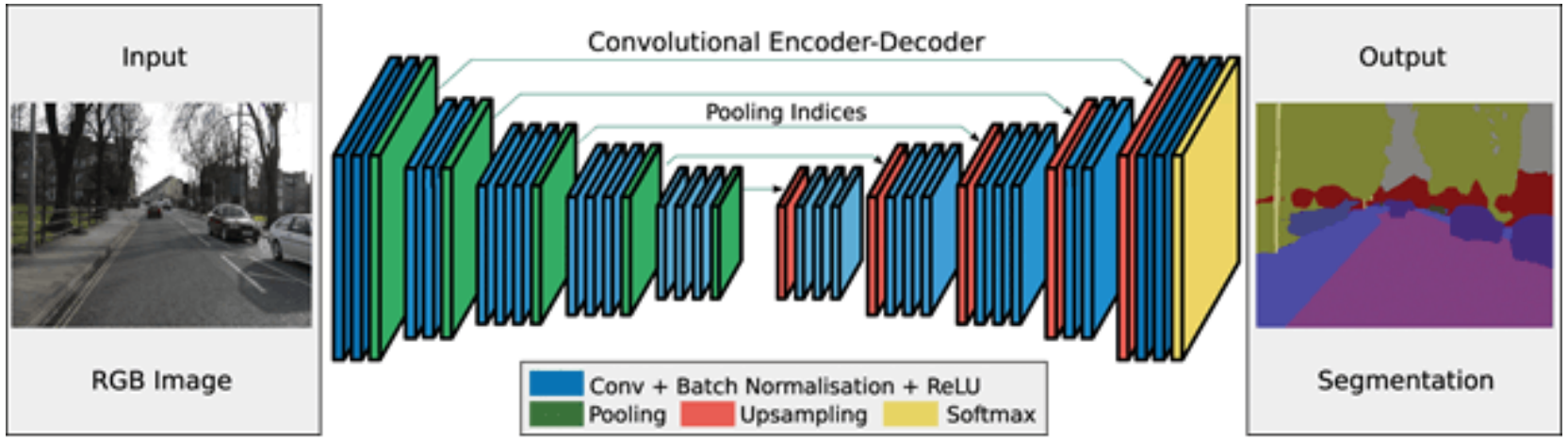
- **Datasets**: Stanford Background, Microsoft COCO, MSRC, KITTI, Microsoft AirSim, SA-1B

Semantic segmentation

- An **evolution of object detection**. For each pixel an object category is assigned



- All object **categories are known** to the model
- **Basic model** for semantic segmentation: Convolutional encoder-decoder



Semantic segmentation: Things and Stuff

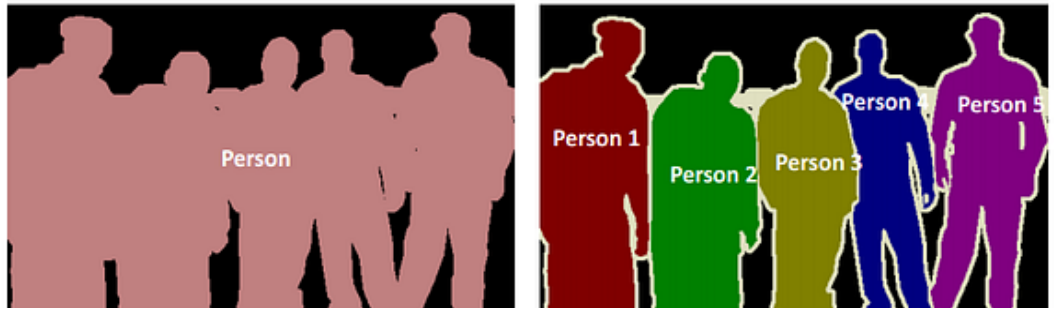
- **Semantic classes** can be either
  - Things: objects with a well-defined shape (little shape variance), e.g. car, person
  - Stuff: amorphous background regions very variable in size, e.g. grass, sky.

  Under certain image conditions, some classes can be both things or stuff. For example, a large group of people can be interpreted as multiple "**persons**"—each a distinctly shaped, countable thing—or a singular, amorphously shaped "**crowd**".

- **Stuff and things are important to deep learning models:**
  - a metal thing on a road is usually a car
  - the blue background behind a boat is probably water
  - the blue background behind a plane is probably sky

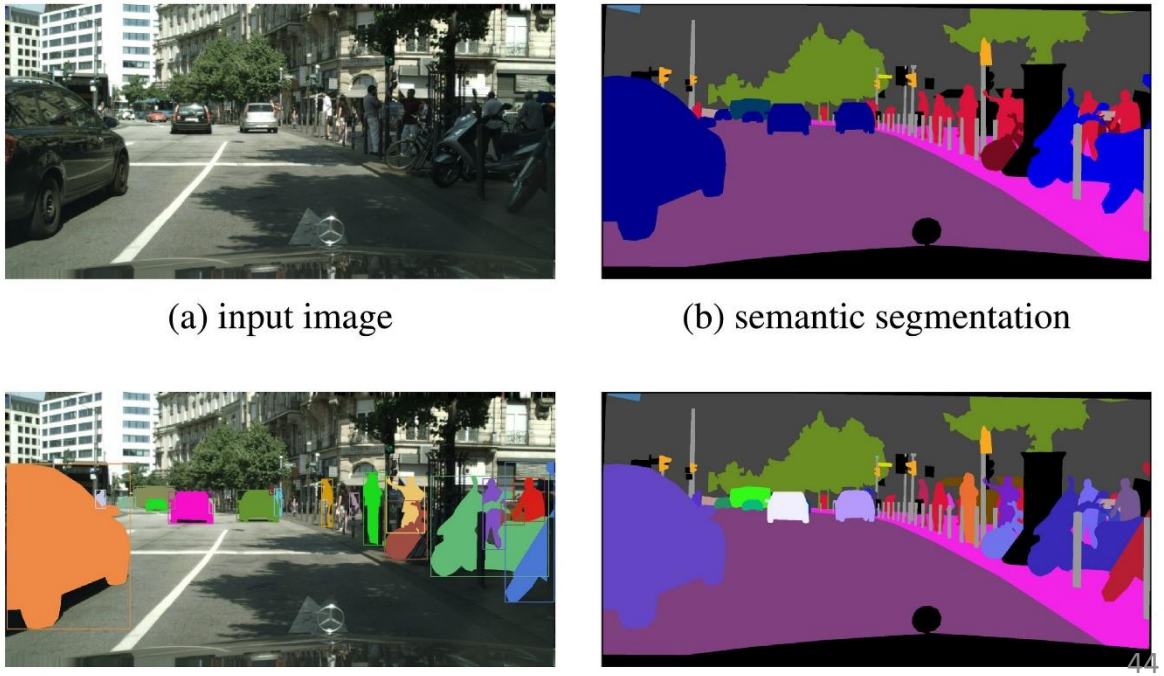
Instance segmentation

- More complex than Semantic segmentation: assign an ID to each pixel of the same category



Semantic Segmentation **Instance Segmentation**

- "Things" can be identified with an ID (e.g. person), but a "stuff" cannot
- If detecting both, we have a panopnic segmentation



(a) input image (b) semantic segmentation

(c) instance segmentation (d) panoptic segmentation

3. DNN-based segmentation

Segment Anything Model (SAM) by Meta AI (https://segment-anything.com/)
a new AI model that can "cut out" any object, in any image, with a single click

- designed to generalize across various segmentation tasks without requiring task-specific training.
- uses a promptable interface, where prompts can be points, boxes, or masks.
- SAM is **not a semantic segmentation model** since it does not categorize pixels into predefined classes.

Strengths: High flexibility and generalization across diverse segmentation tasks.

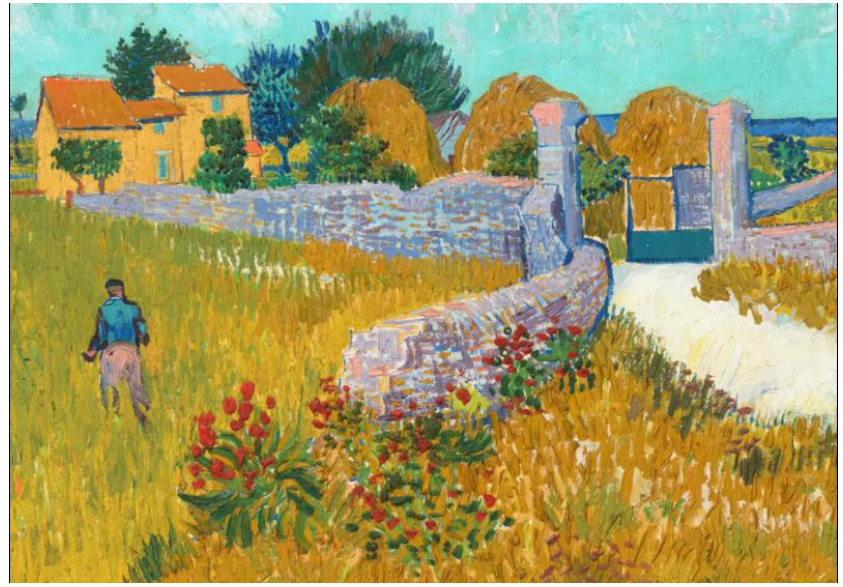
Limitations: May underperform on specific tasks compared to task-specialized models.

Segment Anything Model (SAM)

Prompts: Prompts specifying what to segment in an image allow for a wide range of segmentation tasks without the need for additional training..

Interactive prompts with points and boxes

Random or regular prompt points to automatically segment everything



Zero-shot generalization to segment unfamiliar objects without requiring additional training

Segment Anything Model (SAM)

SAM's promptable design enables flexible integration with other systems

A user's gaze from an AR/VR headset to select an object

Bounding box prompts from an object detector can enable text-to-object segmentation

Segment Anything Model (SAM)

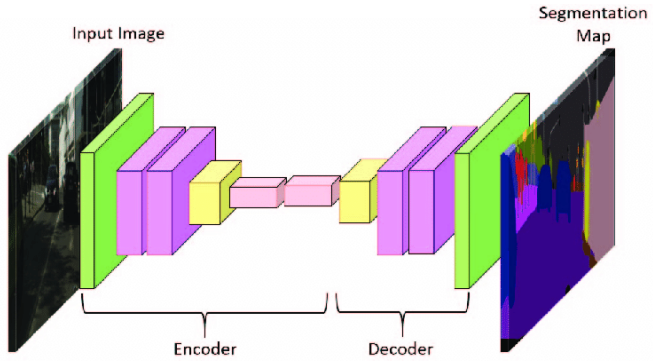
SA-1B Dataset (11M images, 1B+ masks)

SAM's advanced capabilities are the result of its training on millions of images and masks collected through the use of a model-in-the-loop "data engine."

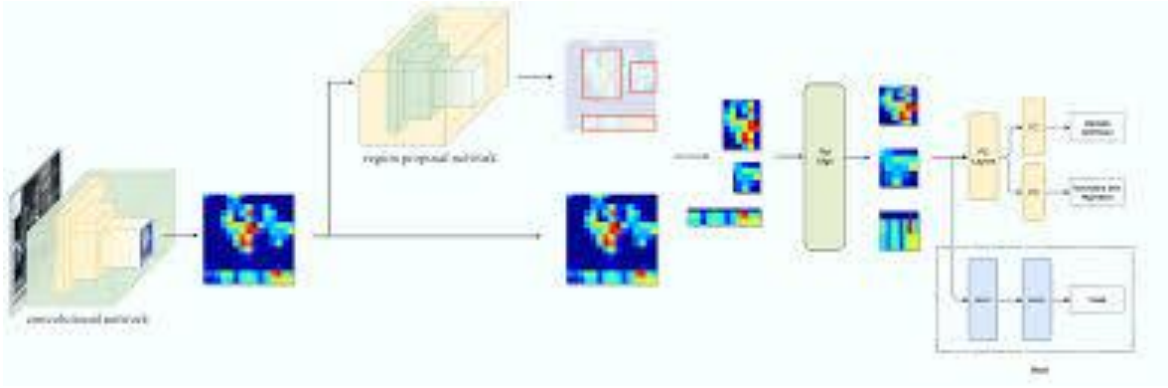
Interactive Segmentation: a user might refine the segmentation iteratively by providing additional prompts

More on SAM in practical sessions and DNN-based recognition.

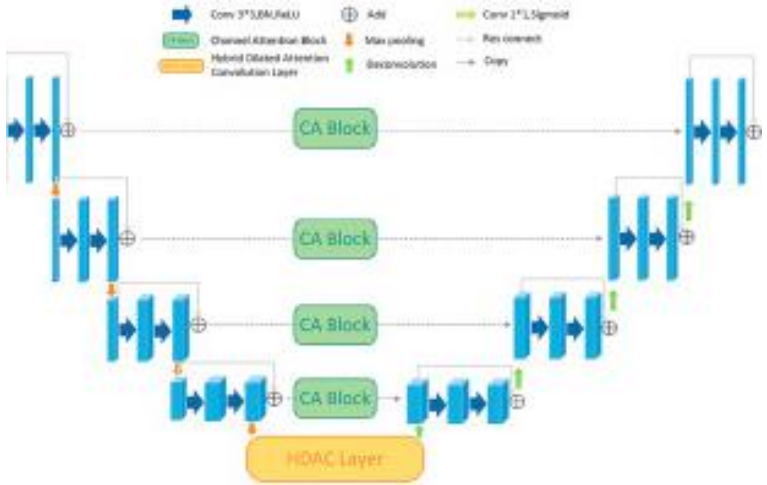
Some models for Segmentation



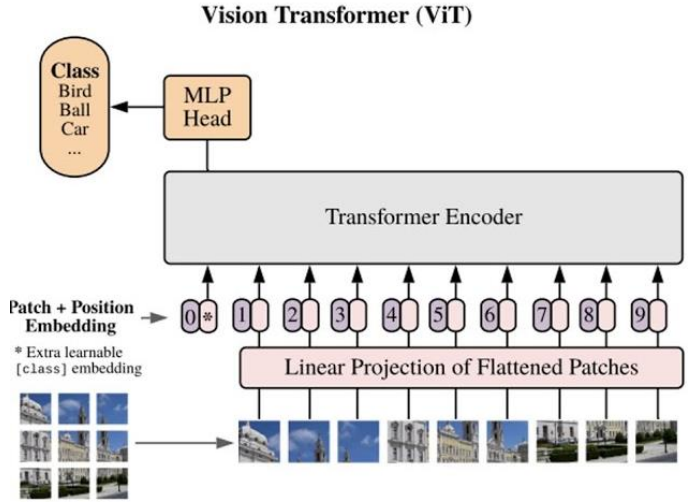
Fully Convolutional Networks (FCNs): An *encoder-decoder* network



Mask R-CNNs: combine a Region Proposal Network (RPN) that generates the bounding boxes and an FCN that generates segmentation masks within each bounding box.



U-Nets: modify FCN architecture to reduce data loss during downsampling with *skip connections*



VisionTransformer: uses attention mechanisms instead of convolutional layers.

4. Conclussions

- **Classical techniques: contour, thresholding, cluster-based**
  - regions whose pixels have similar atributtes, with no semantic meaning
  - rely on well-defined mathematical methods,
  - suitable for simpler tasks
  - need limited data and computational resources.
- In contrast, **deep learning-based approaches (since 2012)**
  - regions with a semantic meaning or not (SAM)
  - excel in complex and high-dimensional scenarios
  - ability to learn from data and adapt to various applications,
  - require significant computational resources and data.