WORKSHOP

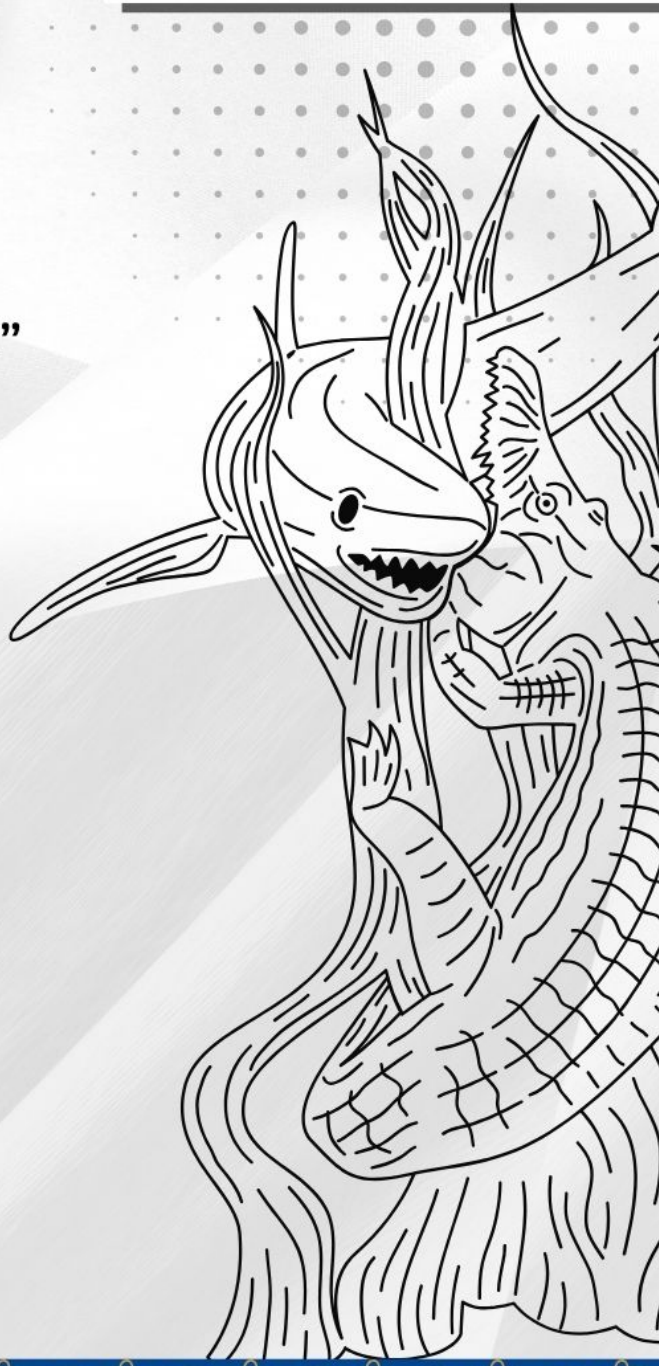
"Deep Learning and Its Applications in Assisting Human"

Section 1 :

IMAGE SEGMENTATION

"basic concept and techniques"

Gede Aditra Pradnyana

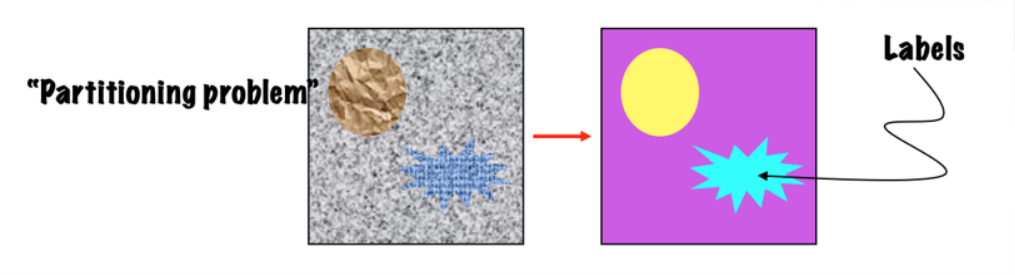


Outline

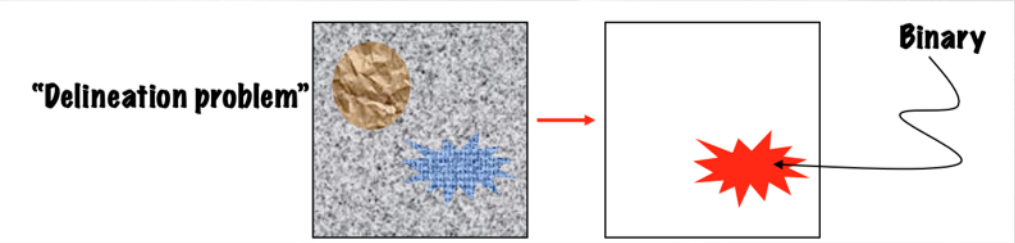
- What is image segmentation?
- Why do we do image segmentation?
- What is the best way to segment images?
- Image Thresholding
- Region-based Segmentation
- Clustering-based Segmentation
- Deep learning-based Segmentation

What is segmentation?

- **Partitioning** image into meaningful pieces

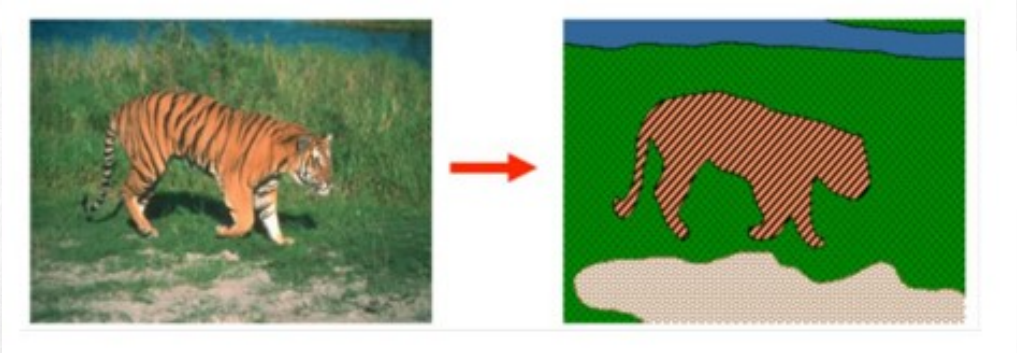


- **Isolating** a specific region of interest



What is segmentation?

- Identify groups of pixels that go together



Why we do segmentation?

- It's not a great idea to process the entire image at the same time, as there will be regions in the image that do not contain any information.
- We can divide or partition the image into various parts called segments.
- By dividing the image into segments, we can use the important segments to process the image.

Why we do segmentation?

- **Detection/Recognition**
  - Where is the vehicle?
  - What type of vehicle is it?
  - Is that person already wearing a mask?
- **Quantifying Object Properties**
  - How big is the tumor? Is it expanding or shrinking?
  - How many people visited the store during the afternoon?
  - How extensive are the cracks in this building?

What is the Best Way to Segment Images?

- **Depends on:**
  - Type of data (e.g., medical, industrial, natural scenes).
  - Image complexity (e.g., noise, contrast, texture).
  - Available resources (e.g., computational power, labeled data).
  - Our task requirements (What you are looking for: shape, size, variability).

Common Approach

- **Thresholding**
  - *How it works*: Separates pixels into different classes based on intensity (e.g., grayscale threshold).
  - *Best for*: Simple, high-contrast images (e.g., separating text from the background).
  - *Example*: Segmenting handwriting images.
- **Region-Based Segmentation**
  - *How it works*: Groups neighboring pixels with similar properties (e.g., intensity or color).
  - *Best for*: Images with clearly defined regions and smooth transitions.
  - *Example*: Separating connected regions in medical scans.

Common Approach

- **Edge-Based Segmentation**
  - *How it works*: Identifies boundaries by detecting edges using methods like Sobel, Canny, or Laplacian filters.
  - *Best for*: Objects with clear boundaries but less suitable for noisy or textured images.
  - *Example*: Detecting cracks in buildings.
- **Clustering**
  - *How it works*: Groups pixels based on features like color or texture into clusters.
  - *Best for*: Images with distinct feature clusters.
  - *Example*: Identifying land types (e.g., water, vegetation) in satellite imagery.

Common Approach

- **Machine Learning-Based Segmentation**
  - *How it works*: Uses classical ML models (e.g., Random Forest, SVM) to classify each pixel based on features.
  - *Best for*: Applications where labeled data is limited and simplicity is desired.
  - *Example*: Classifying defect types in industrial quality control.
- **Deep Learning-Based Segmentation**
  - *How it works*: Employs convolutional neural networks (CNNs) to learn complex spatial patterns.
  - *Types*: Semantic segmentation (labels all pixels into classes) and instance segmentation (differentiates between individual instances of objects).
  - *Best for*: Complex tasks with high variability or when large labeled datasets are available.
  - *Example*: Tumor segmentation in medical imaging.

Hybrid Methods

- **Hybrid Methods**
  - *How it works*: Combines multiple techniques (e.g., edge detection + deep learning).
  - *Best for*: Complex images where a single method is insufficient.
  - *Example*: Satellite image segmentation with preprocessing (e.g., clustering) followed by deep learning.

General Strategy

- **Pixel-based Method**
  - Classify the pixel first, and generate region.
- **Region-based Method**
  - Regions are locally homogeneous.
  - Regions satisfy some properties.

Pixel-based Segmentation

- **Basic/Simple Thresholding**
  - Classify the pixel intensity into two classes using a single threshold value.
- **Adaptive Thresholding**
  - Use adaptive threshold to segment the image into two classes.
- **Range Thresholding**
  - Utilize Gaussian probability to segment pixels.

Basic Thresholding

- Classify every pixel as belonging to an object (foreground) or background using a threshold value.
- Basic segmentation operation:

$$T(x, y) = \begin{cases} 255 & \text{if } I(x, y) > T \\ 0 & \text{if } I(x, y) \leq T \end{cases}$$

- **T** is the threshold (user-defined or automatic).
- Global vs Local Thresholding.

Binary

$$\mathtt{dst}(x, y) = \begin{cases} \mathtt{maxval} & \text{if } \mathtt{src}(x, y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Inverted Binary

$$\mathtt{dst}(x, y) = \begin{cases} 0 & \text{if } \mathtt{src}(x, y) > \mathtt{thresh} \\ \mathtt{maxval} & \text{otherwise} \end{cases}$$

Truncated

$$\mathtt{dst}(x, y) = \begin{cases} \mathtt{threshold} & \text{if } \mathtt{src}(x, y) > \mathtt{thresh} \\ \mathtt{src}(x, y) & \text{otherwise} \end{cases}$$

To Zero

$$\mathtt{dst}(x, y) = \begin{cases} \mathtt{src}(x, y) & \text{if } \mathtt{src}(x, y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

To Zero Inverted

$$\mathtt{dst}(x, y) = \begin{cases} 0 & \text{if } \mathtt{src}(x, y) > \mathtt{thresh} \\ \mathtt{src}(x, y) & \text{otherwise} \end{cases}$$

Basic Thresholding

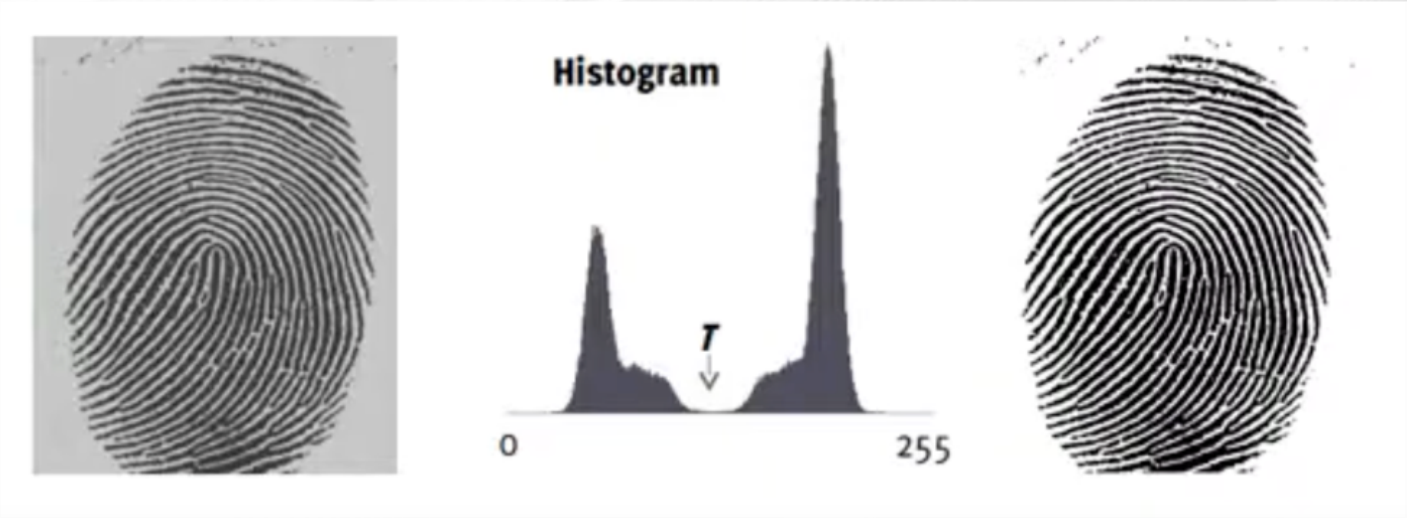
- **How does it work?**
  - Example:

8	5	11	13
18	18	20	5
8	1	5	8
13	11	15	18

T=15 →

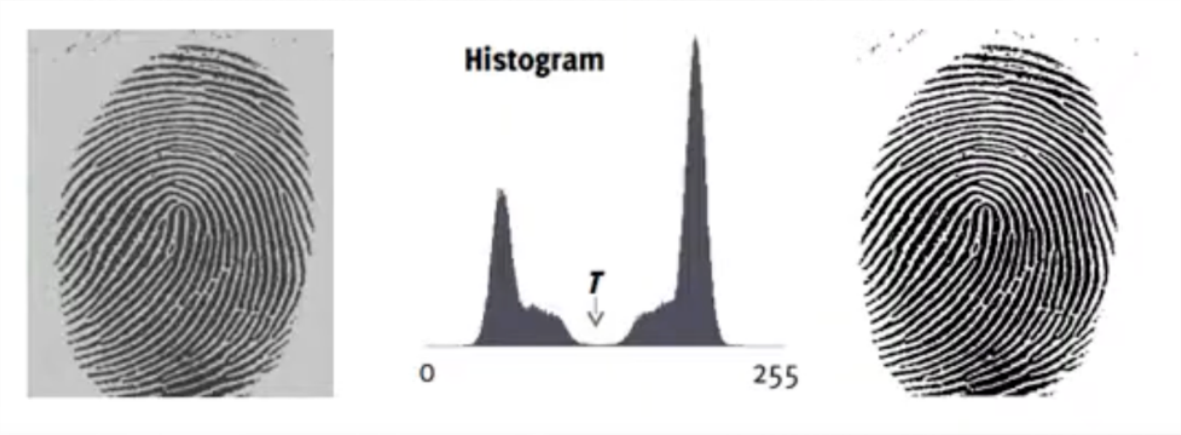
0	0	0	0
255	255	255	0
0	0	0	0
0	0	255	255

- **How to get an optimal threshold?**
  - Using histogram information.



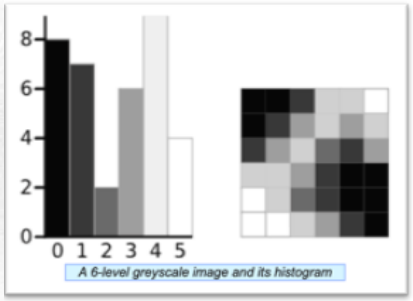
Global Thresholding

- A single threshold value is applied to an entire image.
- These techniques are particularly useful in scenarios where there is significant variation in illumination across different regions of the image.

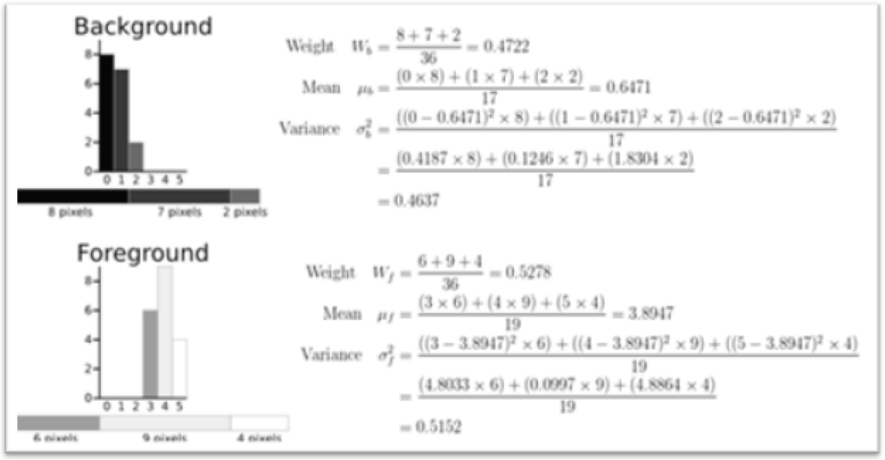


Otsu Method

- Otsu's method calculates the optimal threshold (e.g., $T = 3$) by minimizing the within-class variance.
- Separate the image into two classes: background and foreground.



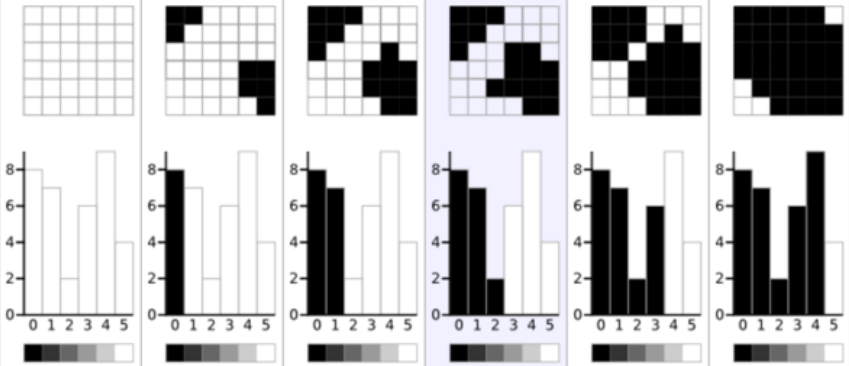
T = 3

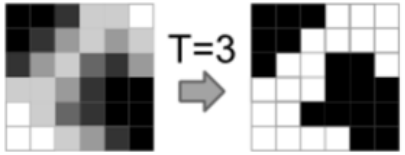


$$\sigma_w^2 = w_B\sigma_B^2 + w_F\sigma_F^2$$

Within Class Variance: $\sigma_w^2 = 0.4722 + 0.4637 + 0.5278 + 0.5152 = 0.4099$

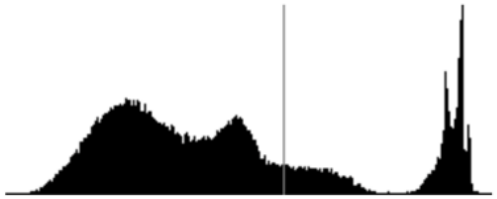
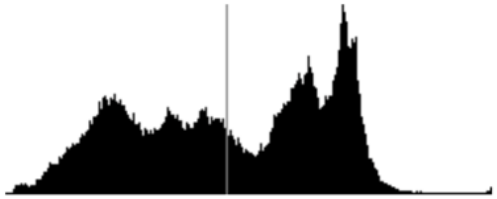
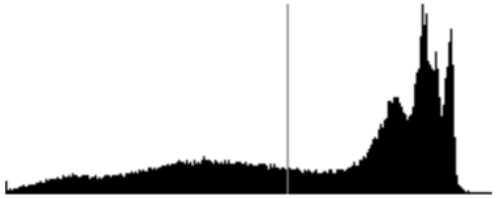
Otsu Method

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
Weight, Background	$W_b = 0$	$W_b = 0.222$	$W_b = 0.4167$	$W_b = 0.4722$	$W_b = 0.6389$	$W_b = 0.8889$
Mean, Background	$M_b = 0$	$M_b = 0$	$M_b = 0.4667$	$M_b = 0.6471$	$M_b = 1.2609$	$M_b = 2.0313$
Variance, Background	$\sigma^2_b = 0$	$\sigma^2_b = 0$	$\sigma^2_b = 0.2489$	$\sigma^2_b = 0.4637$	$\sigma^2_b = 1.4102$	$\sigma^2_b = 2.5303$
Weight, Foreground	$W_f = 1$	$W_f = 0.7778$	$W_f = 0.5833$	$W_f = 0.5278$	$W_f = 0.3611$	$W_f = 0.1111$
Mean, Foreground	$M_f = 2.3611$	$M_f = 3.0357$	$M_f = 3.7143$	$M_f = 3.8947$	$M_f = 4.3077$	$M_f = 5.000$
Variance, Foreground	$\sigma^2_f = 3.1196$	$\sigma^2_f = 1.9639$	$\sigma^2_f = 0.7755$	$\sigma^2_f = 0.5152$	$\sigma^2_f = 0.2130$	$\sigma^2_f = 0$
Within Class Variance	$\sigma^2_W = 3.1196$	$\sigma^2_W = 1.5268$	$\sigma^2_W = 0.5561$	$\boldsymbol{\sigma^2_W = 0.4909}$	$\sigma^2_W = 0.9779$	$\sigma^2_W = 2.2491$



Result of Otsu's Method

Try Otsu Method in Google Colab

Greyscale Image	Binary Image	Histogram
		
		
		

Local Thresholding

- **Having just one value of T may not suffice.**
- Subdivide the image into non-overlapping rectangles.
- These rectangles are chosen small enough so that the illumination of each is approximately uniform.
- Determine a global threshold for each subimage.

Adaptive Thresholding

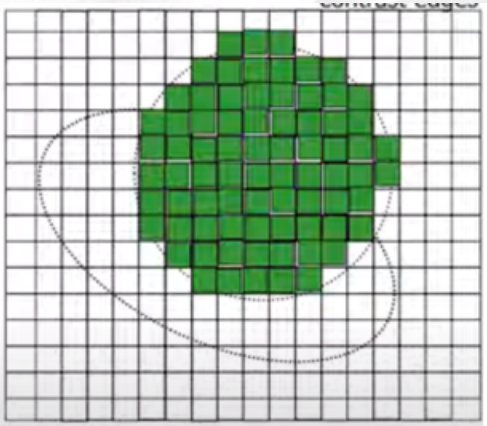
- The goal in adaptive thresholding is to statistically examine local regions of our image and determine an optimal value of T for each region.
- It is common practice to use either the arithmetic mean or the Gaussian mean of the pixel intensities in each region.

Region-based Segmentation

- **Region Growing** → Start from a seed point and check the neighborhood points.
- **Watershed/Flood Fill** → Inspired by how to fill water.
- **Clustering** → Intensity values are assigned as data to clusters.

Region Growing

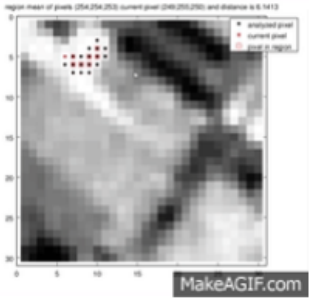
- **Seed Selection** → Start with an initial set of pixels K (initial seed(s)).
- **Region Expansion:**
  - Neighboring pixels are analyzed to check if they meet a predefined similarity criterion (e.g., intensity, color, texture).
  - If the criteria are satisfied, the pixel is added to the region, and its neighbors are further analyzed.
- The process stops when no more pixels meet the criteria or when all image pixels are evaluated.

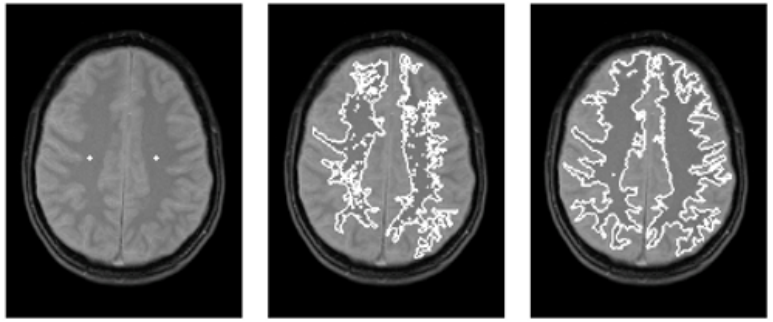


Key Components

- **Similarity Criteria**:
  - Based on pixel intensity, gradient, or color similarity.
  - Example: $|I(x,y) - I(seed)| < T$, where T is a threshold value.
- **Connectivity**:
  - Determines how neighboring pixels are defined.
  - **4-connectivity**: Considers up, down, left, and right neighbors
  - **8-connectivity**: Includes diagonal neighbors.

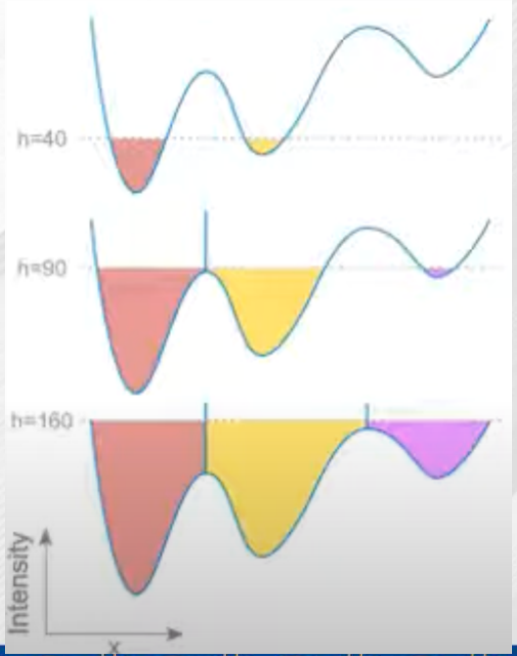
Random pixels are selected in the image and are merged to neighbors if they are homogeneous





Watershed Segmentation

- Inspired by the concept of topography → Think of a grayscale image as a landscape.
- **Flooding the Surface**: Imagine water being poured into the valleys of the surface.
- **Basins (regions)**: Formed as the water fills up the valleys.
- **Watershed Lines**: As water from different basins meets, boundaries are formed, which are the watershed lines.
- **Segmentation**: The final output assigns each pixel to a region or marks it as part of a boundary.



Clustering-based Image Segmentation

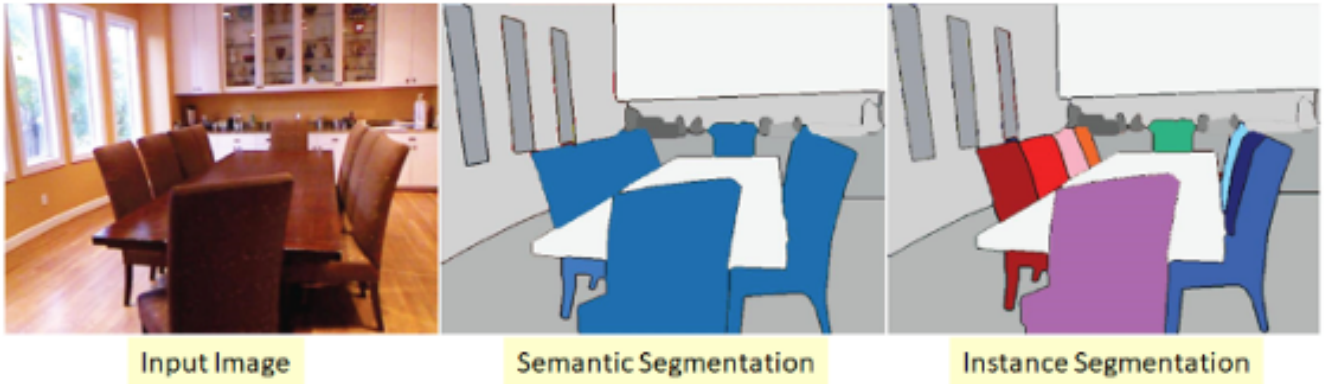
- Represent image pixels as feature vectors.
  - For example, each pixel can be represented as:
    - **Intensity** $\rightarrow$ One-dimensional feature vectors.
    - **Color** $\rightarrow$ Three-dimensional feature vectors.
    - **Color + Coordinates** $\rightarrow$ Five-dimensional feature vectors.
- Cluster them into k clusters $\rightarrow$ k segments.
  - K-means, DBSCAN, Hierarchical Clustering, etc.
  - Divisive vs Merging approaches.
- **Note:**
  - K-means works well for small datasets, but it struggles with large datasets.
  - It processes all samples at every iteration, leading to high time complexity.

Deep Learning-based Image Segmentation

- Deep learning-based image segmentation uses neural networks to classify each pixel in an image, effectively dividing it into distinct regions based on the context.
- Unlike traditional techniques (e.g., thresholding, clustering), deep learning methods learn complex features and patterns directly from data, leading to highly accurate and robust segmentation.

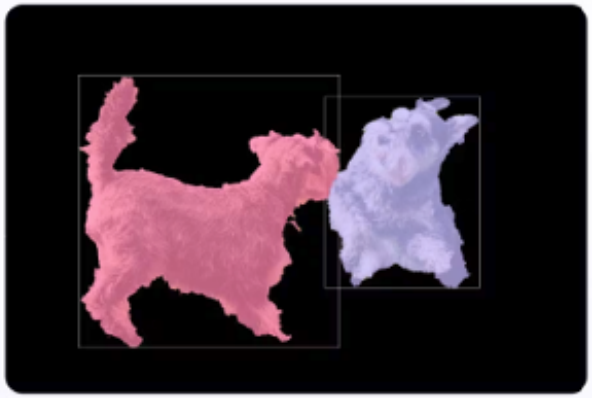
Type of Deep-learning-based Image Segmentation

- Semantic Segmentation:
  - Assigns a label (e.g., “car,” “road”) to every pixel in the image.
  - Example: Classifying pixels as “sky,” “tree,” or “building” in a scene.
  - Limitation: Cannot distinguish between multiple instances of the same object (e.g., two cars in an image are treated as one “car”).
- Instance Segmentation:
  - Distinguishes between individual objects of the same class.
  - Example: Segmenting each car in a parking lot as a separate entity.
- Panoptic Segmentation:
  - Combines semantic and instance segmentation to label every pixel while distinguishing between object instances.



(a) Image

(b) Semantic Segmentation



(c) Instance Segmentation

(d) Panoptic Segmentation

How?

- Deep learning models for segmentation typically follow an **encoder-decoder architecture**:
  1. **Encoder:**
     - Extracts high-level features from the image using convolutional layers.
     - Often based on pretrained models like VGG, ResNet, or EfficientNet.
  2. **Decoder:**
     - Reconstructs a pixel-wise classification map from the encoded features.
     - Upsampling techniques (e.g., transposed convolution) are used to restore the spatial resolution.

Popular Architectures (Part 1)

1. **Fully Convolutional Networks (FCN):**
   - Replaces fully connected layers with convolutional layers for dense pixel predictions.
   - Outputs a lower-resolution feature map, which is *upsampled* for segmentation.
2. **U-Net:**
   - Designed for biomedical image segmentation but widely used in various domains.
   - Uses skip connections between the encoder and decoder to retain fine spatial details.
3. **SegNet:**
   - Encoder-decoder architecture similar to U-Net but focuses on computational efficiency.
   - Uses max-pooling indices for *upsampling*.

Popular Architectures (Part 2)

4. **Mask R-CNN:**
   - Extends Faster R-CNN (object detection) to perform instance segmentation.
   - Generates bounding boxes, class labels, and segmentation masks for each detected object.
5. **DeepLab:**
   - Uses *atrous* convolutions (dilated convolutions) to capture multi-scale context.
   - DeepLab v3+ incorporates an encoder-decoder structure with *atrous spatial pyramid pooling (ASPP)*.
6. **Transformers for Segmentation:**
   - Vision Transformers (ViT) and models like *Segmenter* use attention mechanisms for segmentation.
   - Captures global dependencies in the image.