KRISHAI TECHNOLOGIES

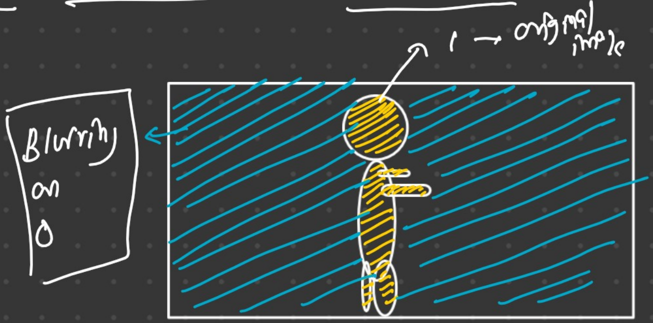
Image Segmentation Using openCV

www.krishnaik.in

Image segmentation is the process of dividing an image into meaningful regions (e.g., foreground vs. background, or object-specific regions). It's used in tasks like object detection, medical imaging, and autonomous driving.

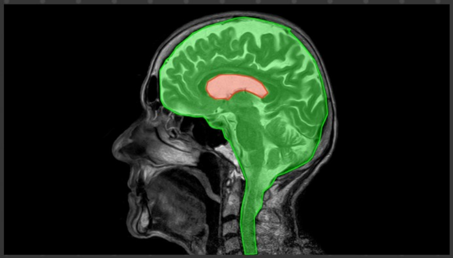
Object Detection →

→ Selective Search



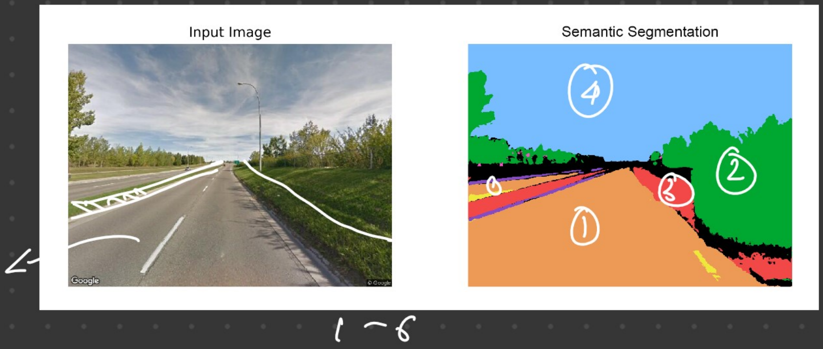
Semantic Segmentation →

→ Healthcare



Semantic Segmentation →

→ Autonomous Driving



Thresholding

Simple Thresholding

→ Global Thresholding

→ 127, 255
→ 127, 255

A single threshold value (e.g., 127) is applied uniformly across the entire image.

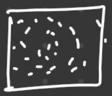
Regions with intensity above the threshold are completely filled, as all pixels in those regions share similar intensity levels.

Adaptive Thresholding

→ Local Thresholding

The threshold value is computed for smaller regions (blocks) of the image based on the intensity of neighboring pixels.

Since the threshold dynamically adjusts based on the local intensity patterns, it often highlights edges where there are rapid changes in pixel intensity.



→ Types

Mean

Threshold = $\boxed{\text{Mean of local region} - C}$

↓

This is the average pixel intensity in the small region you're analyzing.

Gaussian

$\boxed{\text{weighted sum of local region} - C}$

↓

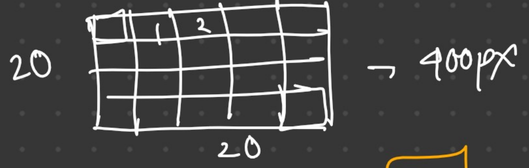
Pixels closer to the center have a greater influence, making the threshold calculation more robust

C = constant to adjust the threshold

7×7

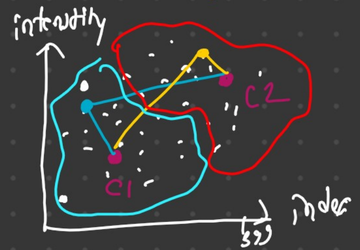
→
→ mean , C → 2
→ −2

K-means Clustering



This algorithm groups pixels into clusters based on their intensity or color similarity.

$\boxed{k=2}$

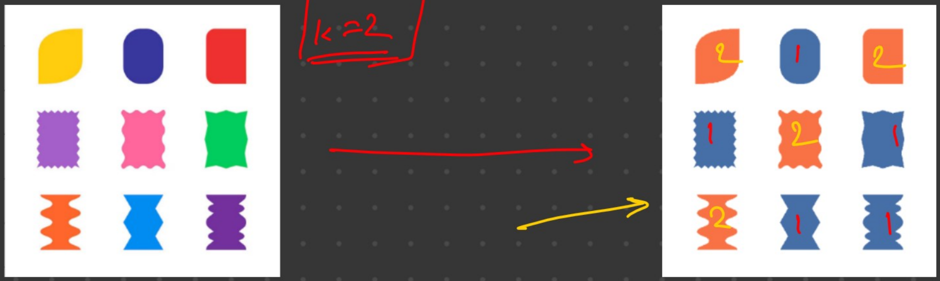


- Flatten the image into a 2D array (pixels and their intensities).
- Specify the number of clusters (e.g., foreground, background).
- Use the K-Means algorithm to group pixels into clusters.
- Assign each cluster a specific label (or color).

$\boxed{R-G-B}$

$\boxed{C1 \rightarrow 6}$
$\boxed{C2 \rightarrow 300}$

$\boxed{k=2}$



Watershed Algorithm

The Watershed Algorithm is used for image segmentation by treating pixel intensities like a topographic surface. It separates overlapping objects in an image by finding watershed lines, which represent boundaries.

Convert the image to grayscale if it's in color.

Apply a binary threshold to create a binary image for separating foreground and background.

Use morphological operations (e.g., cv2.morphologyEx) to remove noise.

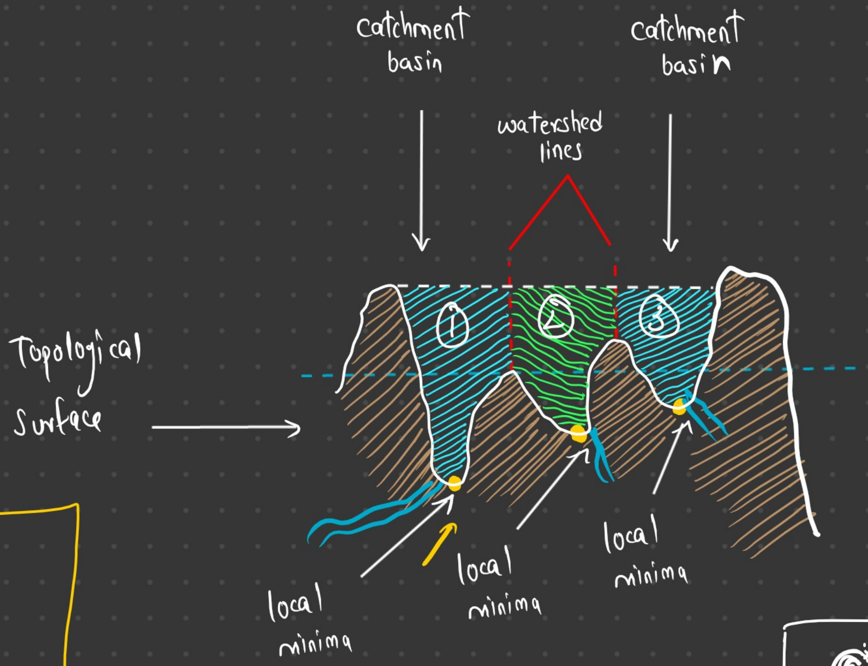
Compute the distance transform, which measures the distance from each pixel to the nearest background pixel.

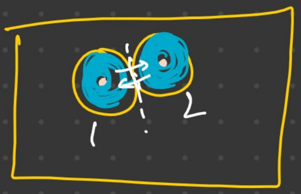
Label the foreground and background using markers.

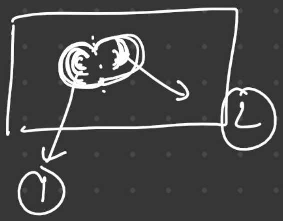
Use the watershed function to compute the watershed lines.

Mark boundaries where watershed lines exist.

marker →
0
1
2
3
4
-1







Bright Pixels: Represent "high" regions (like mountain peaks).

Dark Pixels: Represent "low" regions (like valleys).

Gradients: Represent slopes or edges, showing transitions between high and low areas.

Seed points represent different regions of interest. If we want to segment coins in an image. Coins will be seed points.

When water from two regions meets, a boundary (watershed line) is formed.

Understanding Watershed's "Boundary" Concept

Imagine each seed point (markers) represents a specific object (e.g., the center of a coin) or the background.

As the algorithm processes the image, it "floods" each region outward pixel by pixel.

When pixels from two or more regions (e.g., two coins) meet, the algorithm determines that a boundary exists between them.

These "meeting points" conceptually form the watershed lines.

array
marker
2

Output of the Algorithm: Markers

Each region identified is assigned a unique label (integer value) in the markers array.

Pixels belonging to boundaries are labeled with a special value (commonly -1), indicating the watershed lines.

The "boundaries" are not separate outputs. They are the pixels labeled -1 in the markers output.

The final markers array represents the segmented image.

The purpose of the Watershed Algorithm is not just to find boundaries but to assign labels to regions for segmentation.

This makes the output more useful for tasks like counting objects, measuring areas, or performing instance segmentation.

We can Visualize boundaries by highlighting -1 values. ① ② . . . .

Visualize individual regions by isolating specific labels.