Experiment No. 5
To perform Object segmentation
Name of Student: - 09_Amruta Poojary
Date of Performance: 16/8/2023
Date of Submission: 23/8/2023

Aim: To perform Object segmentation

Objective: To perform object segmentation using the Watershed and GrabCut algorithms

Theory:

1. Image segmentation with the Watershed algorithm

Image segmentation is the process of dividing an image into several disjoint small local areas or cluster sets according to certain rules and principles. The watershed algorithm is a computer vision technique used for image region segmentation

The watershed algorithm uses topographic information to divide an image into multiple segments or regions.

The algorithm views an image as a topographic surface, each pixel representing a different height.

The watershed algorithm uses this information to identify catchment basins, similar to how water would collect in valleys in a real topographic map.

The watershed algorithm identifies the local minima, or the lowest points, in the image.

These points are then marked as markers.

The algorithm then floods the image with different colors, starting from these marked markers.

As the color spreads, it fills up the catchment basins until it reaches the boundaries of the objects or regions in the image.

The **catchment basin** in the watershed algorithm refers to a region in the image that is filled by the spreading color starting from a marker

. The catchment basin is defined by the boundaries of the object or region in the image and the local minima in the intensity values of the pixels.

The algorithm uses the catchment basins to divide the image into separate regions and then identifies the boundaries between the basins to create a segmentation of the image for object recognition, image analysis, and feature extraction tasks.

The whole process of the watershed algorithm can be summarized in the following steps:

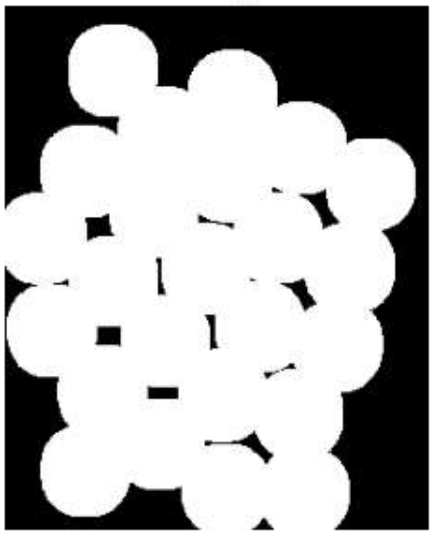
Marker placement: The first step is to place markers on the local minima, or the lowest points, in the image. These markers serve as the starting points for the flooding process.

Flooding: The algorithm then floods the image with different colors, starting from the markers. As the color spreads, it fills up the catchment basins until it reaches the boundaries of the objects or regions in the image.

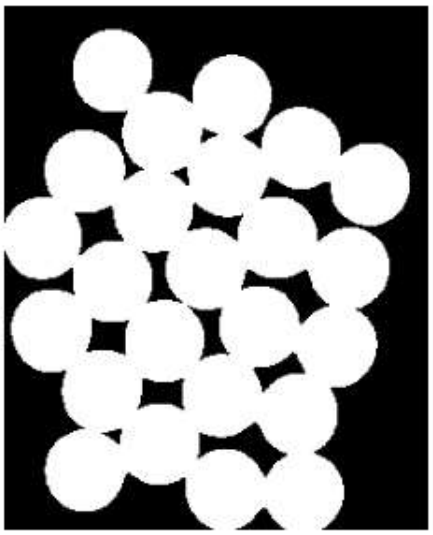
Catchment basin formation: As the color spreads, the catchment basins are gradually filled, creating a segmentation of the image. The resulting segments or regions are assigned unique colors, which can then be used to identify different objects or features in the image.

Boundary identification: The watershed algorithm uses the boundaries between the different colored regions to identify the objects or regions in the image. The resulting segmentation can be used for object recognition, image analysis, and feature extraction tasks.

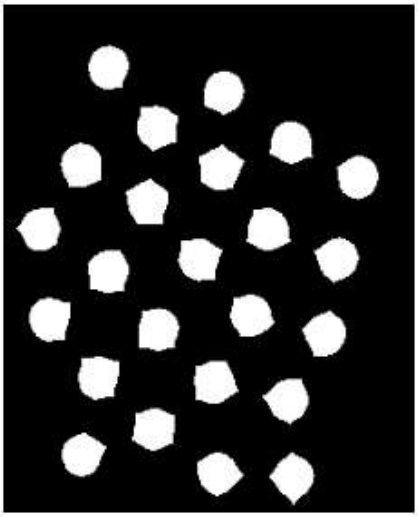
Sure Background



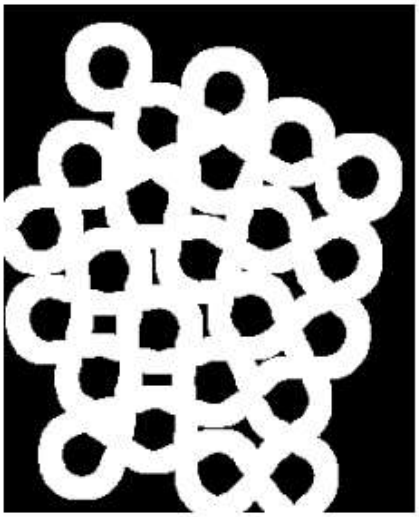
Distance Transform



Sure Foreground



Unknown



2. Example of foreground detection with GrabCut

GrabCut is a perfect tool for foreground/background segmentation. The GrabCut algorithm consists of the following steps:

1. A rectangle including the subject(s) of the picture is defined.

2. The area lying outside the rectangle is automatically defined as a background.

3. The data contained in the background is used as a reference to distinguish background areas from foreground areas within the user-defined rectangle.

4. A Gaussian Mixture Model (GMM) models the foreground and background, and labels undefined pixels as probable background and probable foreground.

5. Each pixel in the image is virtually connected to the surrounding pixels through virtual edges, and each edge is assigned a probability of being foreground or background, based on how similar it is in color to the pixels surrounding it

6. Each pixel (or node as it is conceptualized in the algorithm) is connected to either a foreground or a background node

7. After the nodes have been connected to either terminal (the background or foreground, also called the source or sink, respectively), the edges between nodes belonging to different terminals are cut (hence the name, GrabCut). Thus, the image is segmented into two parts. The following figure adequately represents the algorithm:

Code:

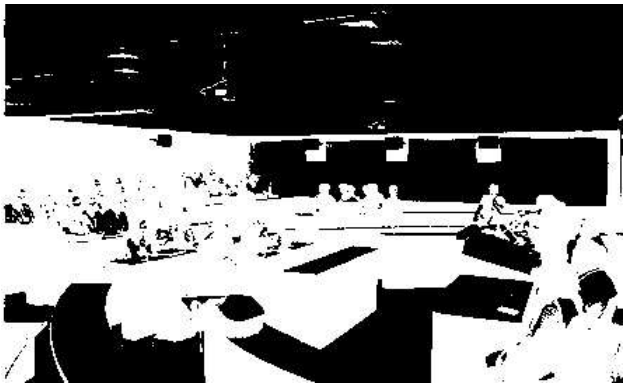
a.) Watershed Algorithm

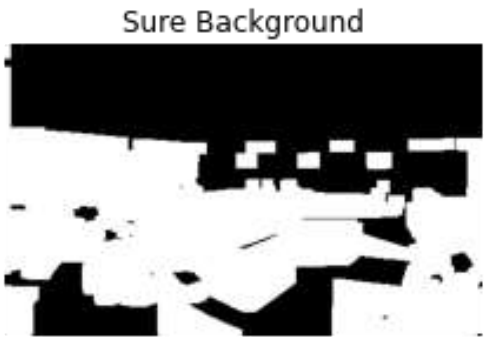
```
import cv2
import numpy as np
from IPython.display import Image, display
from matplotlib import pyplot as plt

# Plot the image
def imshow(img, ax=None):
        if ax is None:
                ret, encoded = cv2.imencode(".jpg", img)
                display(Image(encoded))
        else:
                ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
                ax.axis('off')

#Image loading
img = cv2.imread("seminar.png")

#image grayscale conversion
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Show image
imshow(img)

#Threshold Processing
ret, bin_img = cv2.threshold(gray,
                                                0, 255,
                                                cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
imshow(bin_img)

# noise removal
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
bin_img = cv2.morphologyEx(bin_img,
                                        cv2.MORPH_OPEN,
```

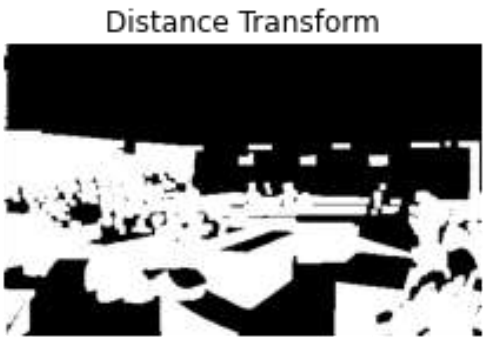
```
                                        kernel,
                                        iterations=2)
imshow(bin_img)

# Create subplots with 1 row and 2 columns
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(8, 8))
# sure background area
sure_bg = cv2.dilate(bin_img, kernel, iterations=3)
imshow(sure_bg, axes[0,0])
axes[0, 0].set_title('Sure Background')

# Distance transform
dist = cv2.distanceTransform(bin_img, cv2.DIST_L2, 5)
imshow(dist, axes[0,1])
axes[0, 1].set_title('Distance Transform')

#foreground area
ret, sure_fg = cv2.threshold(dist, 0.5 * dist.max(), 255,
cv2.THRESH_BINARY)
sure_fg = sure_fg.astype(np.uint8)
imshow(sure_fg, axes[1,0])
axes[1, 0].set_title('Sure Foreground')

# unknown area
unknown = cv2.subtract(sure_bg, sure_fg)
imshow(unknown, axes[1,1])
axes[1, 1].set_title('Unknown')

plt.show()
```

Input:

Output :-




Sure Background


Distance Transform


Sure Foreground


Unknown

b.) Grabcut Algorithm

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

#original = cv2.imread('../images/statue_small.jpg')
original = cv2.imread('seminar.png')
img = original.copy()
mask = np.zeros(img.shape[:2], np.uint8)

bgdModel = np.zeros((1, 65), np.float64)
fgdModel = np.zeros((1, 65), np.float64)

rect = (100, 1, 421, 378)
cv2.grabCut(img, mask, rect, bgdModel, fgdModel, 5,
cv2.GC_INIT_WITH_RECT)

mask2 = np.where((mask==2) | (mask==0), 0, 1).astype('uint8')
img = img*mask2[:,:,np.newaxis]

plt.subplot(121)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title("grabcut")
plt.xticks([])
plt.yticks([])

plt.subplot(122)
plt.imshow(cv2.cvtColor(original, cv2.COLOR_BGR2RGB))
plt.title("original")
plt.xticks([])
plt.yticks([])

plt.show()
```

Output:

Conclusion:

Object Segmentation or Image segmentation is the process of categorizing each pixel value of an image to a particular class. Using OpenCV we can perform tasks in machine vision such as object segmentation by implementation algorithm's like Watershed algorithm and GrabCut algorithm, OpenCV has built-in function for performing image segmentation using Watershed by using the watershed() function as GrabCut by using the GrabCut function