Aim: To study object segmentation.

Objective: Object segmentation using the Watershed and GrabCut algorithms, Example of foreground detection with GrabCut, Image segmentation with the Watershed algorithm

Theory:

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. It is an algorithm that correctly determines the "outline of an object".

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.

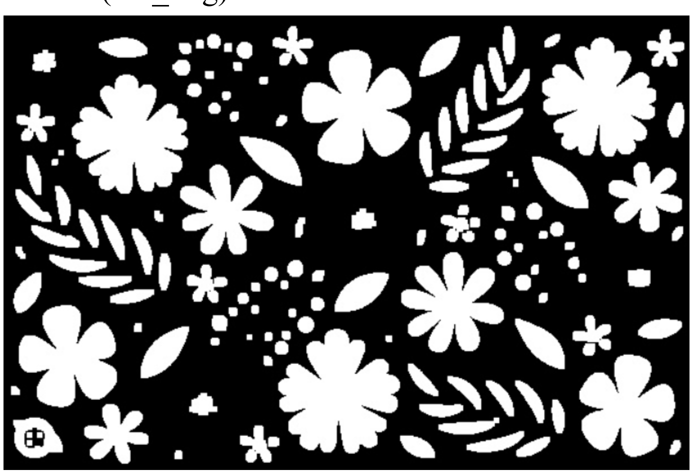
- Catchment basin formation: As the color spreads, the catchment basins are gradually filled, creating a segmentation of the image. The resulting segments or regions are assigned unique colors, which can then be used to identify different objects or features in the image.
- Boundary identification: The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

Code and Output:

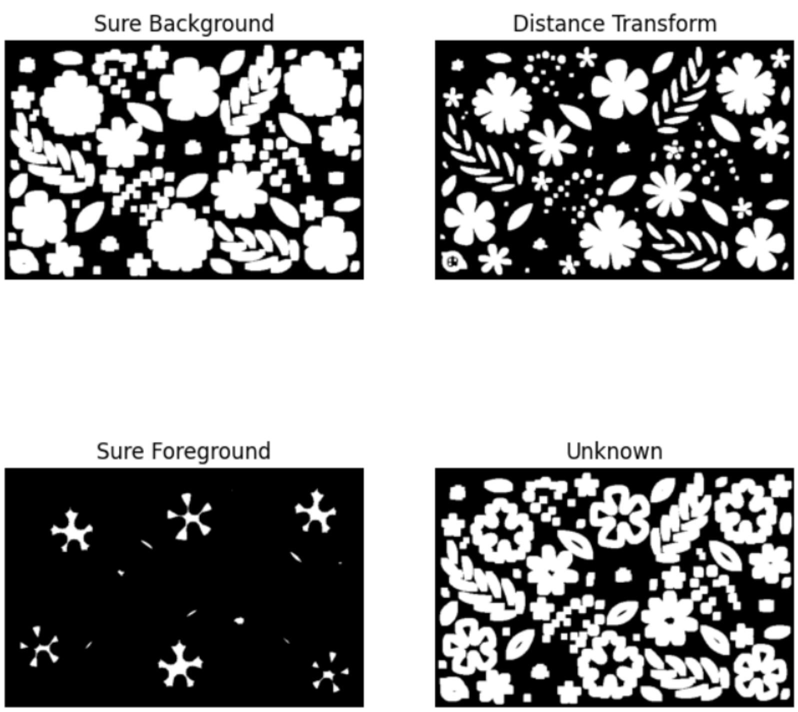
```
import cv2
import numpy as np
from IPython.display import Image, display
from matplotlib import pyplot as plt
# Plot the image
def imshow(img, ax=None):
  if ax is None:
    ret, encoded = cv2.imencode(".jpg", img)
    display(Image(encoded))
  else:
    ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    ax.axis('off')
img = cv2.imread("/content/image.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imshow(img)
```

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

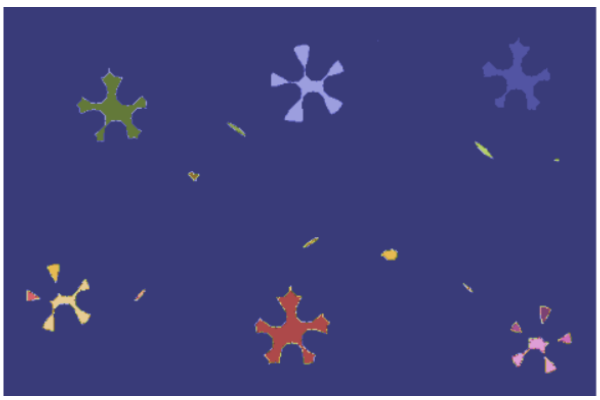
```
# noise removal
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
bin_img = cv2.morphologyEx(bin_img,
          cv2.MORPH_OPEN,
          kernel,
          iterations=2)
imshow(bin_img)
```



```
# 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()
```

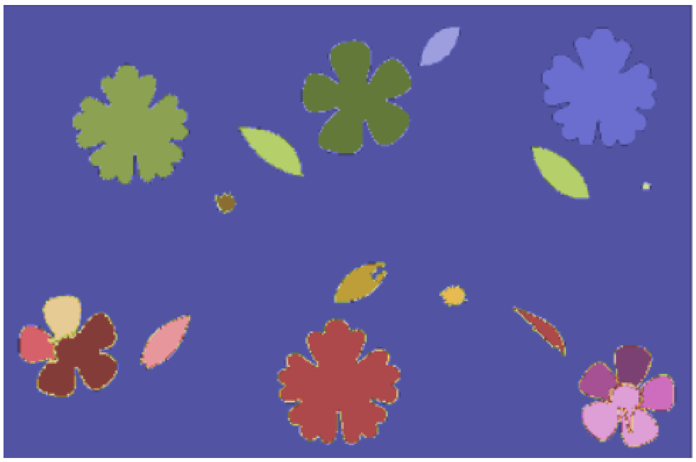


```
# Marker labelling
# sure foreground
ret, markers = cv2.connectedComponents(sure_fg)
# Add one to all labels so that background is not 0, but 1
markers += 1
# mark the region of unknown with zero
markers[unknown == 255] = 0
fig, ax = plt.subplots(figsize=(6, 6))
ax.imshow(markers, cmap="tab20b")
ax.axis('off')
plt.show()
```



```
# watershed Algorithm
markers = cv2.watershed(img, markers)
fig, ax = plt.subplots(figsize=(5, 5))
ax.imshow(markers, cmap="tab20b")
ax.axis('off')
plt.show()
labels = np.unique(markers)
coins = []
for label in labels[2:]:
# Create a binary image in which only the area of the label is in the foreground
#and the rest of the image is in the background
  target = np.where(markers == label, 255, 0).astype(np.uint8)
# Perform contour extraction on the created binary image
  contours, hierarchy = cv2.findContours(
    target, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE
  )
```

```
  coins.append(contours[0])
# Draw the outline
img = cv2.drawContours(img, coins, -1, color=(0, 23, 223), thickness=2)
imshow(img)
```



Conclusion:

The Watershed algorithm is a powerful tool for segmenting objects in an image, especially when there are clear intensity differences between objects and the background. The code implements the watershed algorithm using OpenCV to segment an image into separate objects or regions. The code first loads the image and converts it to grayscale, performs some preprocessing steps, places markers on the local minima, floods the image with different colors, and finally identifies the boundaries between the regions. The resulting segmented image is then displayed.