PCV Lab-8

190030485

G G Sanjana

PRELAB

1.What is Image Segmentation?

A. Image segmentation is a method in which a digital image is broken down into various subgroups called Image segments which helps in reducing the complexity of the image to make further processing or analysis of the image simpler. Segmentation in easy words is assigning labels to pixels. All picture elements or pixels belonging to the same category have a common label assigned to them. For example: Let's assume a problem where the picture has to be provided as input for object detection. Rather than processing the whole image, the detector can be inputted with a region selected by a segmentation algorithm. This will prevent the detector from processing the whole image thereby reducing inference time.

2. Why is image masking used?

A. Image masking is a process of graphics software like Photoshop to hide some portions of an image and to reveal some portions. It is a non-destructive process of image editing. Most of the time it enables you to adjust and tweak the mask later if necessary. Very often, it is an efficient and more creative way of image manipulation services.

Masking is a non-destructive process. We can make changes later or fine-tune to our masks whenever we need to. But if we erase the unwanted areas it is difficult for bringing those if we need those areas later in the process of image editing. The same is true if we cut an object or portion from the image-making path on them. We will not be able to include more adjacent areas easily. Just if we want to

hide some area we've cut out, it may be possible by using the masking technique.

3. What are the different types of Image segmentation techniques?

A.

Image Segmentation Techniques:

1. Threshold Method
2. Edge Based Segmentation
3. Region Based Segmentation
4. Clustering Based Segmentation
5. Watershed Based Method
6. Artificial Neural Network Based Segmentation

4. Describe and explain the watershed algorithm?

A. The watershed algorithm is a classic algorithm used for segmentation and is especially useful when extracting touching or overlapping objects in images, such as the coins in the figure above.

Using traditional image processing methods such as thresholding and contour detection, we would be unable to extract each individual coin from the image — but by leveraging the watershed algorithm, we are able to detect and extract each coin without a problem.

When utilizing the watershed algorithm we must start with user-defined markers. These markers can be either manually defined via point-and-click, or we can automatically or heuristically define them using methods such as thresholding and/or morphological operations.

Based on these markers, the watershed algorithm treats pixels in our input image as local elevation (called a topography) — the method "floods" valleys, starting from the markers and moving outwards, until the valleys of different markers meet each other. In order to

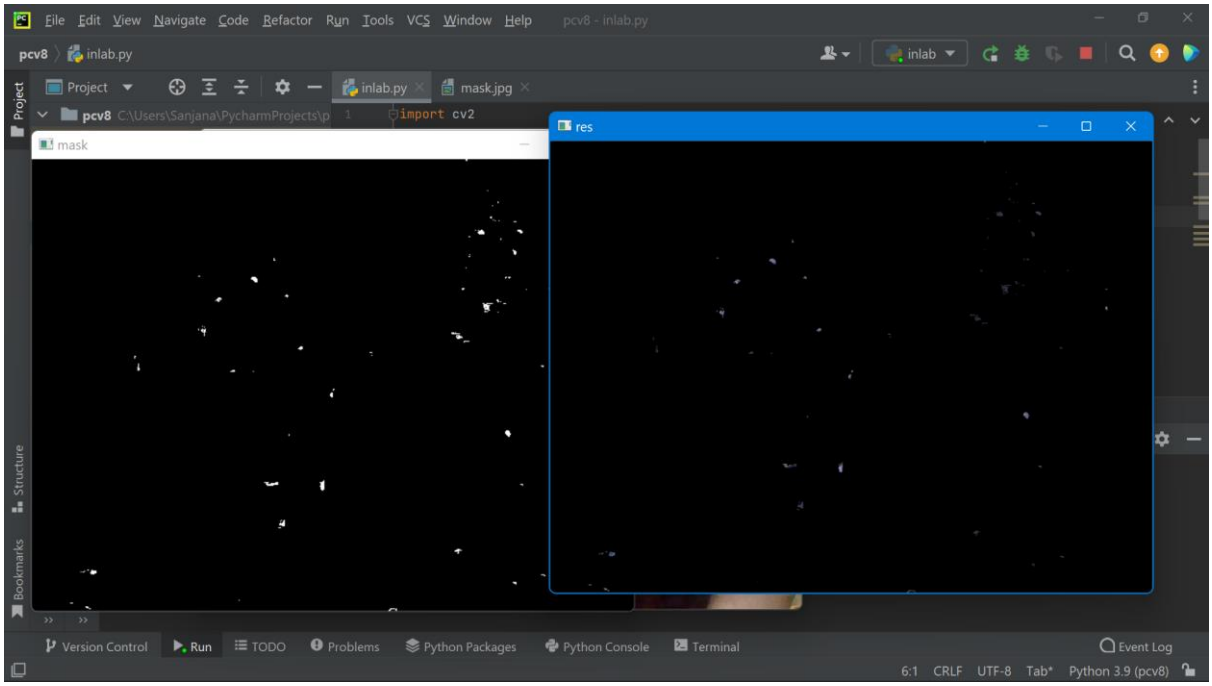
obtain an accurate watershed segmentation, the markers must be correctly placed.

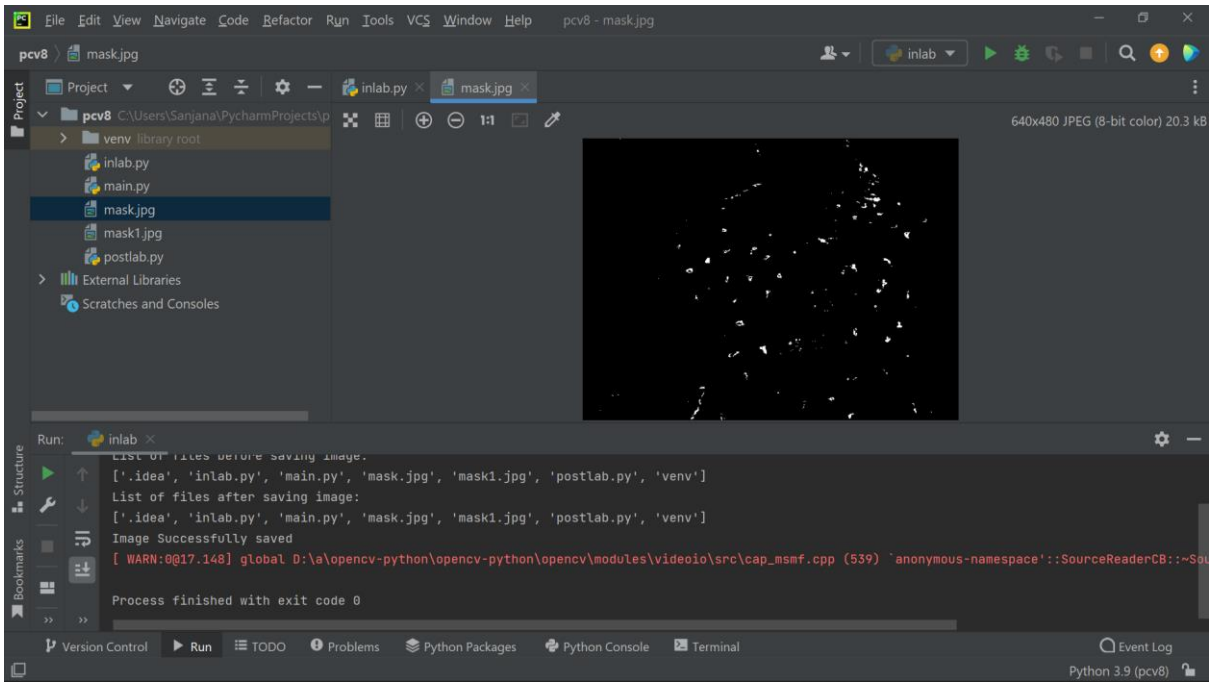
INLAB

Kishore and varun are students of K L University they were willing to implement masking technique that is to detect the blue color object using Webcam. Help Kishore and Varun to implement image masking using python and openCV?

A.

```
import cv2
import os
import numpy as np

cap = cv2.VideoCapture(0)

while(1):
   _, frame = cap.read()

   hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
   lower_blue = np.array([110,50,50])
   upper_blue = np.array([130,255,255])

   mask = cv2.inRange(hsv, lower_blue, upper_blue)

   res = cv2.bitwise_and(frame,frame, mask= mask)
   cv2.imshow('frame',frame)
   cv2.imshow('mask',mask)
   cv2.imshow('res',res)

   k = cv2.waitKey(5) & 0xFF
   if k == 27:
      print("Pressed escape key")
      print("Saving Image....")
      path = r'C:\Users\Sanjana\PycharmProjects\pcv8'
      os.chdir(path)
      print("List of files before saving image:")
      print(os.listdir(path))
      filename = 'mask.jpg'
      cv2.imwrite(filename, mask)
      print("List of files after saving image:")
      print(os.listdir(path))
      print('Image Successfully saved')
      break

cv2.destroyAllWindows()

cap.release()
```



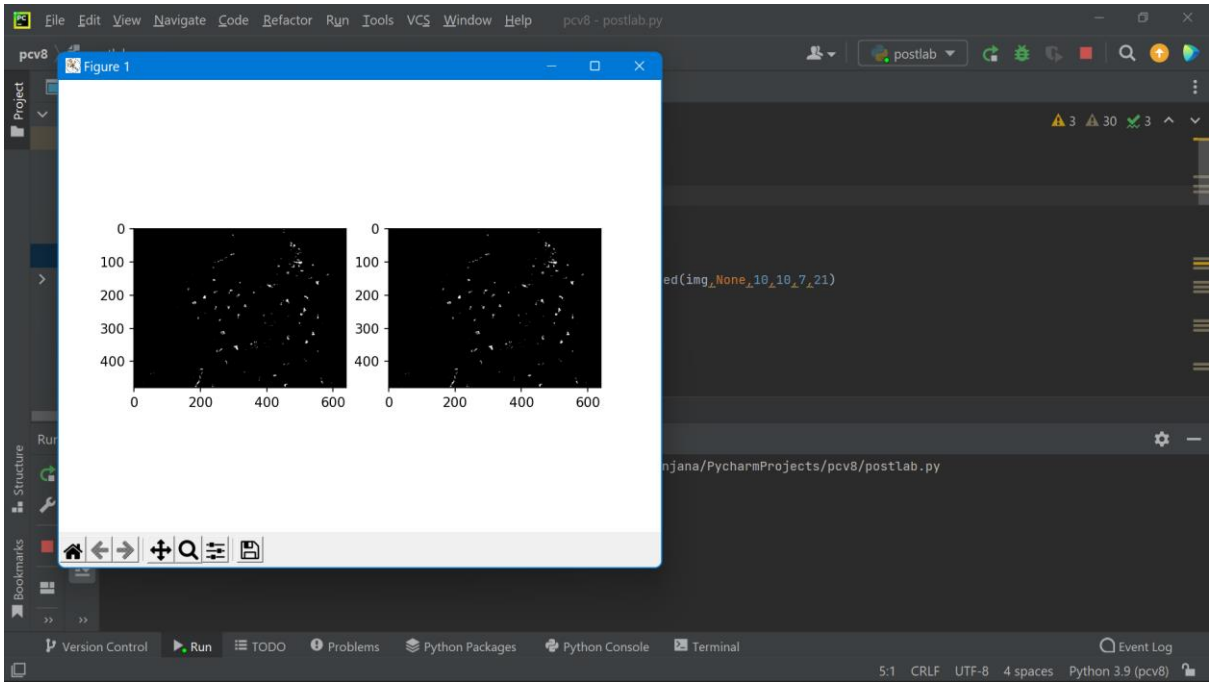


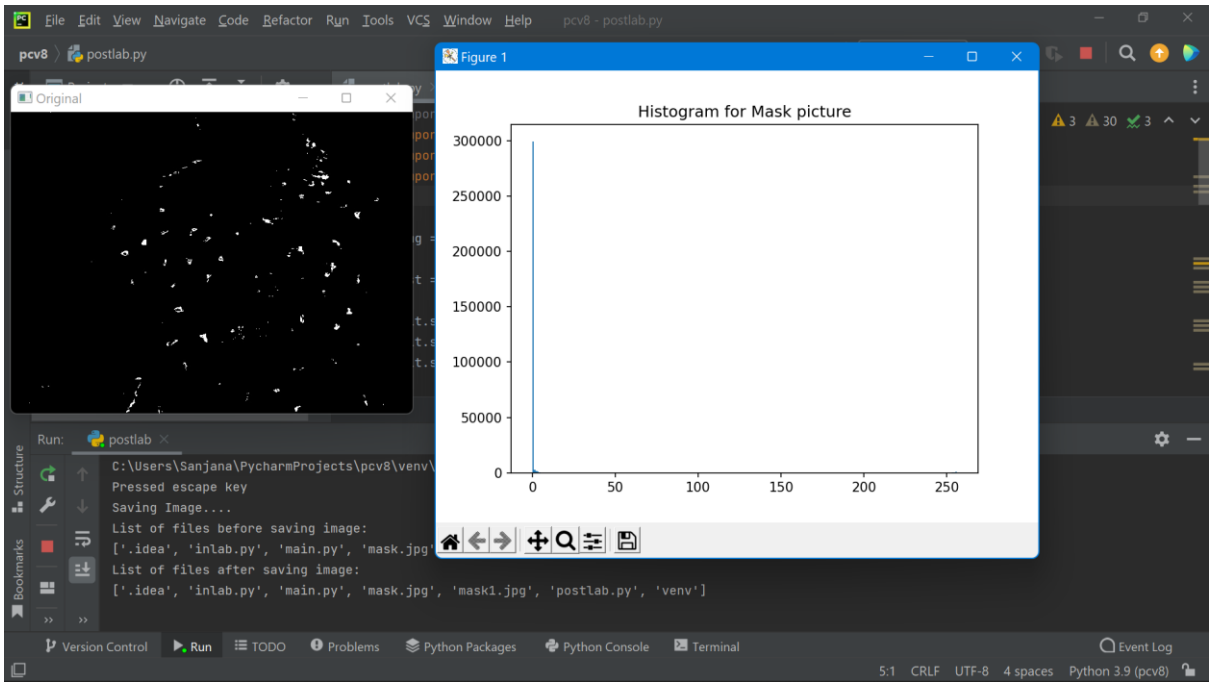
POSTLAB

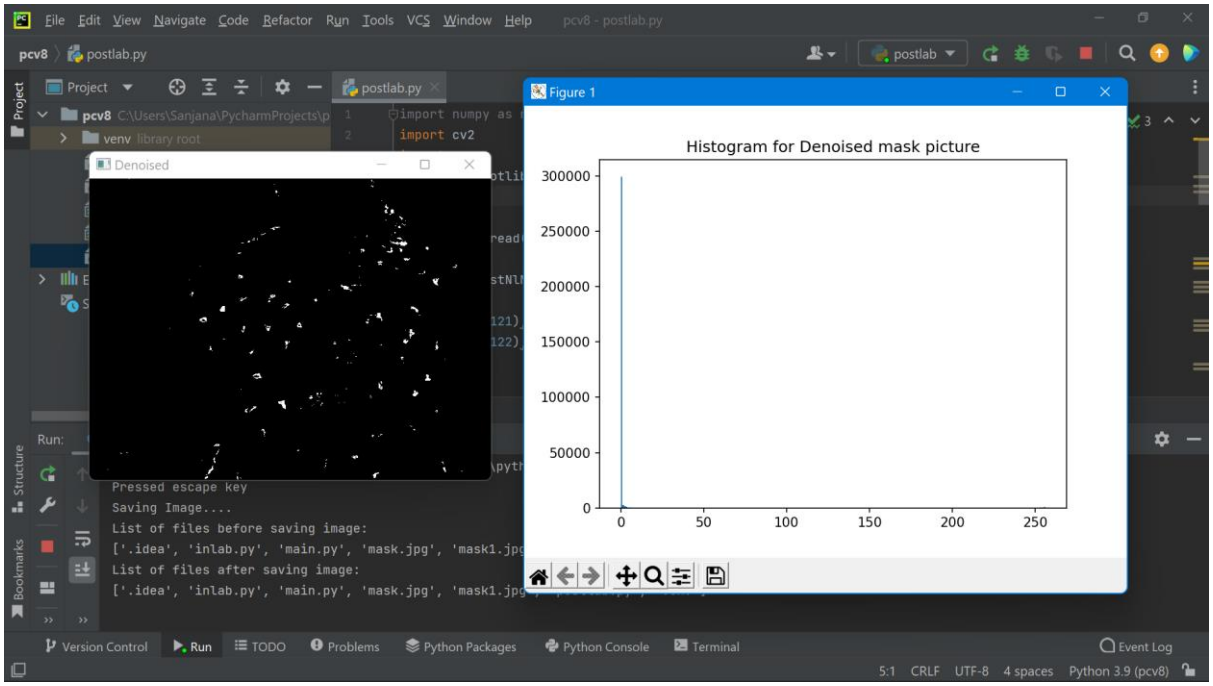
1.Apply image denoising and histogram equalization to the resulting masked image. Is there any noticeable change? Explain.

A.

```
import numpy as np
import cv2
import os
import matplotlib.pyplot as plt


img = cv2.imread('mask.jpg')

dst = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21)

plt.subplot(121),plt.imshow(img)
plt.subplot(122),plt.imshow(dst)
plt.show()

print("Pressed escape key")
print("Saving Image....")
path = r'C:\Users\Sanjana\PycharmProjects\pcv8'
os.chdir(path)
print("List of files before saving image:")
print(os.listdir(path))
filename = 'mask1.jpg'
cv2.imwrite(filename, dst)
print("List of files after saving image:")
print(os.listdir(path))

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

gray_img = cv2.imread('mask.jpg', cv2.IMREAD_GRAYSCALE)
cv2.imshow('Original',gray_img)
hist = cv2.calcHist([gray_img],[0],None,[256],[0,256])
plt.hist(gray_img.ravel(),256,[0,256])

plt.title('Histogram for Mask picture')
plt.show()

gray_img1 = cv2.imread('mask1.jpg', cv2.IMREAD_GRAYSCALE)
cv2.imshow('Denoised',gray_img1)
hist1 = cv2.calcHist([gray_img1],[0],None,[256],[0,256])
plt.hist(gray_img1.ravel(),256,[0,256])

plt.title('Histogram for Denoised mask picture')
plt.show()

while True:
    k = cv2.waitKey(0) & 0xFF
    if k == 27: break
cv2.destroyAllWindows()
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pcv8 - postlab.py
pcv8
postlab
Figure 1
0
100
200
300
400
0
200
400
600
njana/PycharmProjects/pcv8/postlab.py
Version Control
Run
TODO
Problems
Python Packages
Python Console
Terminal
Event Log
5:1
CRLF
UTF-8
4 spaces
Python 3.9 (pcv8)


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pcv8 - postlab.py
pcv8
postlab.py
Original
Figure 1
Histogram for Mask picture
300000
250000
200000
150000
100000
50000
0
0
50
100
150
200
250
Run:
postlab
C:\Users\Sanjana\PycharmProjects\pcv8\venv\
Pressed escape key
Saving Image....
List of files before saving image:
['.idea', 'inlab.py', 'main.py', 'mask.jpg'
List of files after saving image:
['.idea', 'inlab.py', 'main.py', 'mask.jpg', 'mask1.jpg', 'postlab.py', 'venv']
Version Control
Run
TODO
Problems
Python Packages
Python Console
Terminal
Event Log
5:1
CRLF
UTF-8
4 spaces
Python 3.9 (pcv8)



2. Compare and contrast the different methods available for image segmentation.

A.

Techniques	**Description**	**Advantages**	**Disadvantages**
Thresholding Method	Focuses on finding peak values based on the histogram of the image to find similar pixels	Doesn't require complicated pre-processing, simple	Many details can get omitted, threshold errors are common
Edge Based Method	based on discontinuity detection unlike similarity detection	Good for images having better contrast between objects.	Not suitable for noisy images

Region-Based Method	based on partitioning an image into homogeneous regions	Works really well for images with a considerate amount of noise, can take user markers for fasted evaluation	Time and memory consuming
Traditional Segmentation Algorithms	Divides image into k number of homogenous, mutually exclusive clusters – hence obtaining objects	Proven methods, reinforced with fuzzy logic and more useful for real-time application.	Determining cost function for minimization can be difficult.
Watershed Method	based on topological interpretation of image boundaries	segments obtained are more stable, detected boundaries are distinct	Gradient calculation for ridges is complex.
Neural Networks	based on deep learning algorithms – Convolutional Neural Networks	easy implementation, no need for following any complicated algorithms, ready-made libraries available in Python, more practical applications	Training the model for custom and business images is time consuming and resource costly.