EXPT. NO:5

DATE: 26.02.2024

IMAGE PROCESSING TECHNIQUES USING OPEN CV IN COMPUTER VISION

AIM:

To implement the following filtering operations using PyCharm:

1. Image segmentation
2. Corner detection
3. Face and Eye detection
4. Contour detection
5. Template matching

SOFTWARE USED:

Google Colab.

THEORY:

1. IMAGE SEGMENTATION:

Image segmentation is the process of partitioning an image into multiple segments or regions based on certain characteristics such as color, intensity, or texture. The goal is to simplify the representation of an image and make it more meaningful and easier to analyze. Common techniques for image segmentation include thresholding, clustering (such as K-means clustering), region growing, and edge-based segmentation.

2. CORNER DETECTION:

Corner detection is a fundamental operation in computer vision used to identify distinct features or interest points in an image. Corners are points in an image where the intensity gradient exhibits significant changes in multiple directions. Common corner detection algorithms include Harris Corner Detection, Shi-Tomasi Corner Detection, and FAST (Features from Accelerated Segment Test).

3. Face and Eye Detection:

Face and eye detection are essential tasks in computer vision applications, particularly in face recognition and biometric systems. Face detection involves locating and identifying human faces within an image or video frame, while eye detection focuses specifically on detecting the eyes within those faces. Various methods for face and eye detection include Haar Cascade Classifiers, Histogram of Oriented Gradients (HOG), and deep learning-based approaches like Convolutional Neural Networks (CNNs).

4. Contour Detection:

Contour detection involves identifying and extracting the boundaries of objects or regions within an image. Contours are outlines that represent the edges of objects and are useful for tasks such as object recognition, shape analysis, and image understanding. Common techniques for contour detection include edge detection algorithms like Canny Edge Detection, Sobel Edge Detection, and contour tracing algorithms like the Douglas-Peucker algorithm.

5. Template Matching:

Template matching is a technique used to find instances of a template (or a small image patch) within a larger image. It involves sliding the template over the image and computing a similarity measure at each position to determine the best match. Common similarity measures include sum of squared differences (SSD) and normalized cross-correlation (NCC). Template matching is often used in object detection, pattern recognition, and image registration applications.

Other examples are:

1. *Image Denoising:*

Image denoising is the process of removing noise from an image while preserving important details and structures. Noise can degrade the quality of an image and hinder further processing or analysis. Common denoising techniques include Gaussian filtering, median filtering, bilateral filtering, and more advanced methods such as non-local means denoising and wavelet denoising.

2. Image Thresholding:

Image thresholding is a method used to segment an image into regions based on pixel intensity values. It involves selecting a threshold value and classifying pixels as foreground or background depending on whether their intensity is above or below the threshold. Thresholding is widely used in image segmentation, object detection, and feature extraction tasks.

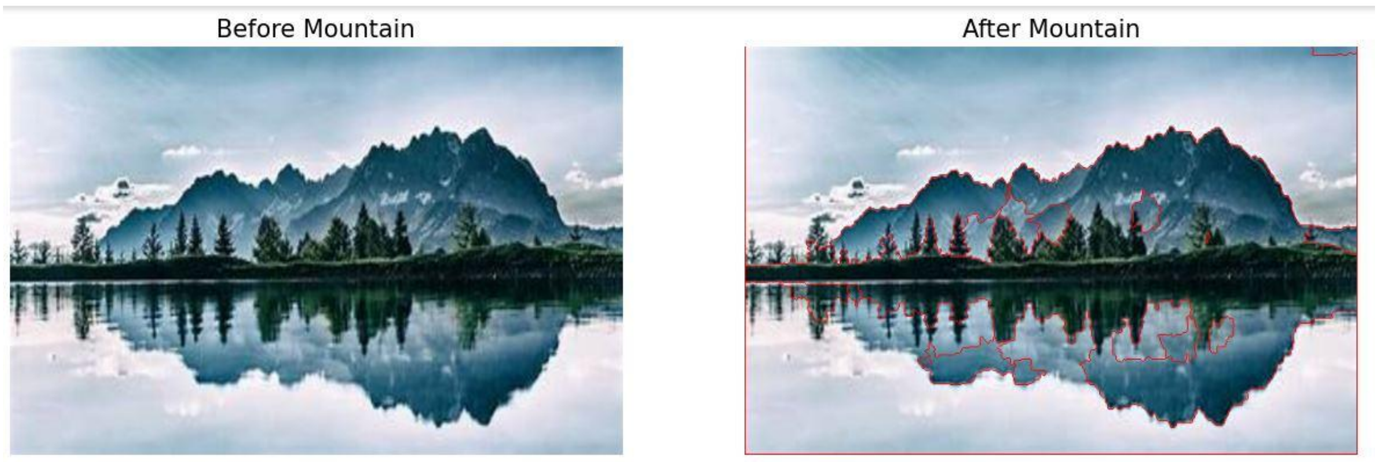
PROGRAM:

1. IMAGE SEGMENTATION:

```
img = cv2.imread('lymphocytes-min.jpg', cv2.IMREAD_COLOR)
resized_img = cv2.resize(img, (height, width))
img_copy = resized_img.copy()
gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray, thresh = 0, maxval = 255, type =
cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
opening = cv2.morphologyEx(thresh, op = cv2.MORPH_OPEN, kernel = kernel, iterations
= 2)
background = cv2.dilate(opening, kernel = kernel, iterations = 5)
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, foreground = cv2.threshold(dist_transform, thresh = 0.2 * dist_transform.max(),
maxval = 255, type = cv2.THRESH_BINARY)
foreground = np.uint8(foreground)
unknown = cv2.subtract(background, foreground)
ret, markers = cv2.connectedComponents(foreground)
markers = markers + 1
markers[unknown == 255] = 0
markers = cv2.watershed(resized_img, markers)
resized_img[markers == -1] = [0, 0, 255]
plt.figure(figsize=(15, 8))
plt.subplot(1, 2, 1).set_title('Lymphocytes Image', fontsize = font_size); plt.axis('off')
plt.imshow(cv2.cvtColor(img_copy, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2).set_title('After Watershed Algorithm', fontsize = font_size); plt.axis('off')
plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))
plt.show()
```

OUTPUT:

[Image type: RGB   Image Format: .jpg]

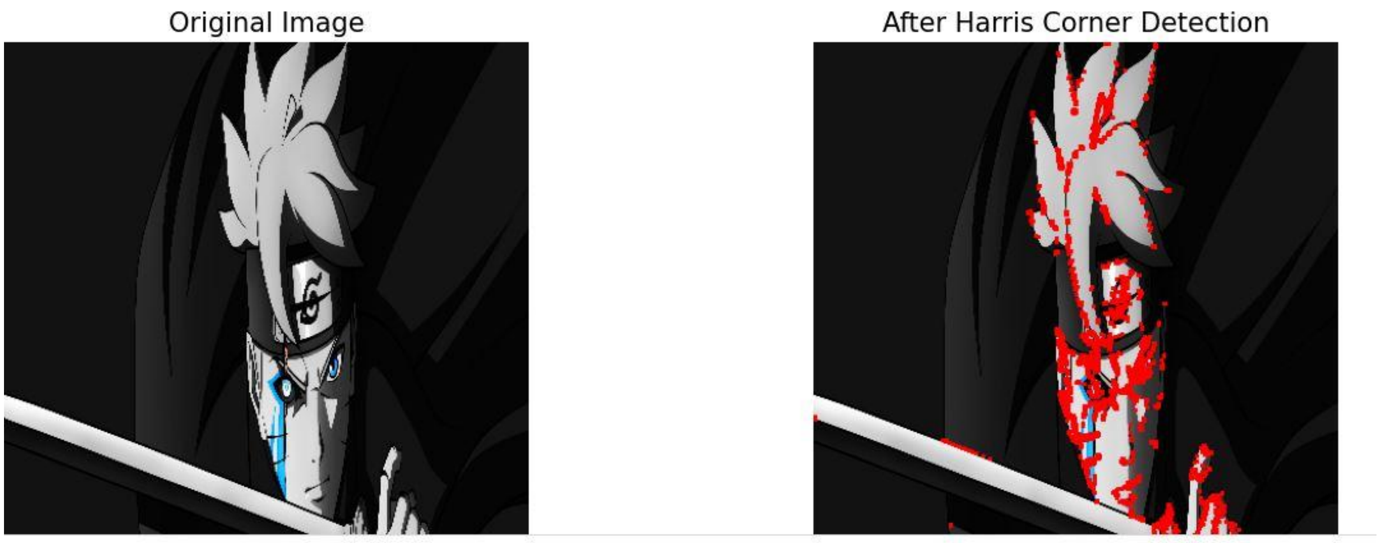


2. CORNER DETECTION:

```
img = cv2.imread('../input/cv-images/corners-min.jpg')
resized_img = cv2.resize(img, (height, width))
img_copy = resized_img.copy()
gray = cv2.cvtColor(resized_img,cv2.COLOR_BGR2GRAY)
gray = np.float32(gray)
corners = cv2.cornerHarris(gray, blockSize = 2, ksize = 3, k = 0.04)
corners = cv2.dilate(corners, None)
resized_img[corners > 0.0001 * corners.max()] = [0, 0, 255]
plt.figure(figsize=(15, 8))
plt.subplot(1, 2, 1).set_title('Original Image', fontsize = font_size); plt.axis('off')
plt.imshow(cv2.cvtColor(img_copy, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2).set_title('After Harris Corner Detection', fontsize = font_size);
plt.axis('off')
plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))
plt.show()
```

OUTPUT:

[Image type: RGB   Image Format: .jpg]

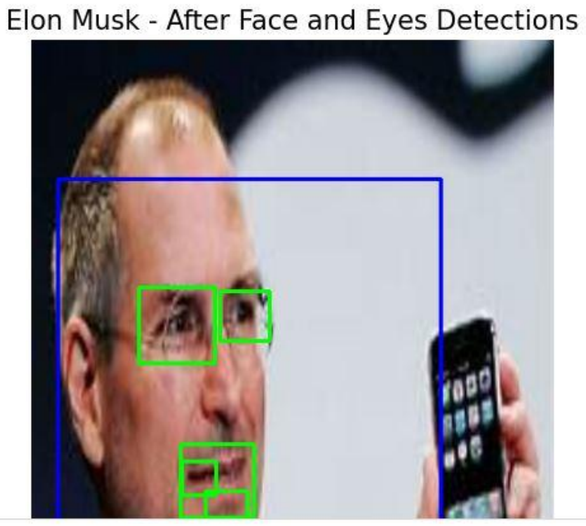


3. FACE AND EYE DETECTION:

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
img = cv2.imread('../input/cv-images/elon-min.jpg')
img = cv2.resize(img, (height, width))
img_copy = img.copy()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.3, minNeighbors = 5)
for (fx, fy, fw, fh) in faces:
img = cv2.rectangle(img, (fx, fy), (fx + fw, fy + fh), (255, 0, 0), 2)
roi_gray = gray[fy:fy+fh, fx:fx+fw]
roi_color = img[fy:fy+fh, fx:fx+fw]
eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex, ey, ew, eh) in eyes:
cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)
plt.figure(figsize=(15, 8))
plt.subplot(1, 2, 1).set_title('Elon Musk', fontsize = font_size); plt.axis('off')
plt.imshow(cv2.cvtColor(img_copy, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2).set_title('Elon Musk - After Face and Eyes Detections', fontsize =
font_size); plt.axis('off')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

OUTPUT:

[Image type: RGB Image Format: .jpg]

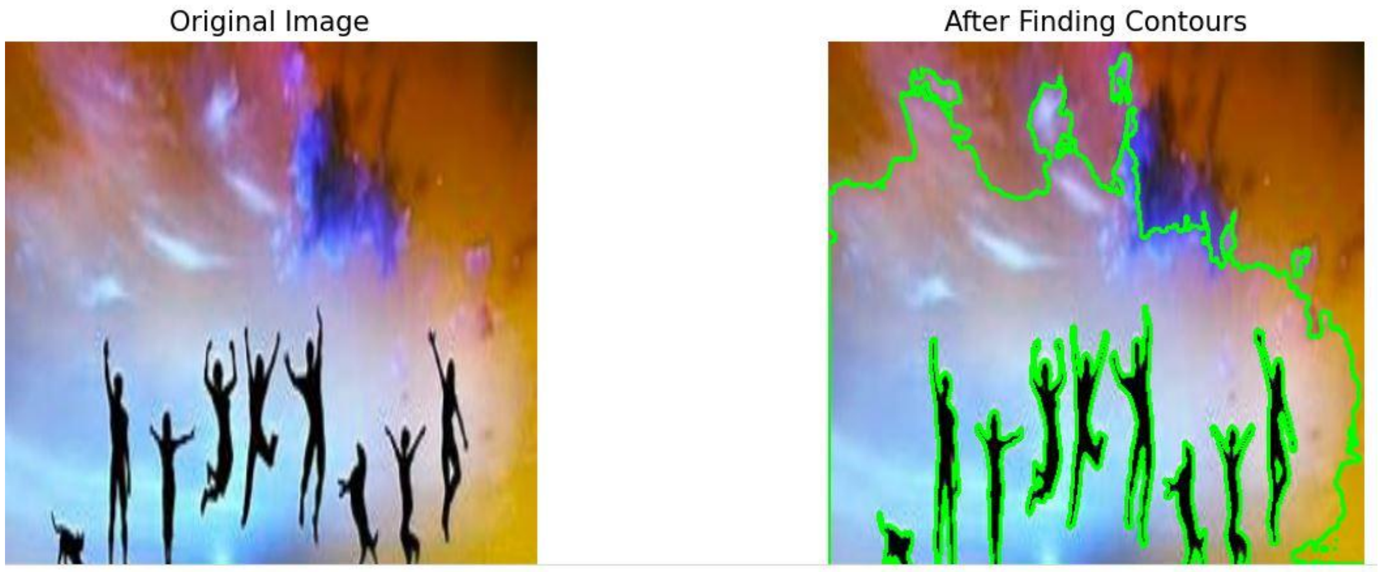


4. CONTOURS

```
plt.figure(figsize=(15,  8))
img = cv2.imread('contours-min.jpg', cv2.IMREAD_COLOR)
resized_img = cv2.resize(img, (height, width))
contours_img = resized_img.copy()
img_gray = cv2.cvtColor(resized_img,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(img_gray, thresh = 127, maxval = 255, type =
cv2.THRESH_BINARY)
contours, hierarchy = cv2.findContours(thresh, mode = cv2.RETR_TREE,
method = cv2.CHAIN_APPROX_NONE)
cv2.drawContours(contours_img, contours, contourIdx = -1, color = (0, 255,
0), thickness = 2)
plt.subplot(1,2,1).set_title('Original Image', fontsize = font_size); plt.axis('off')
plt.imshow(resized_img)
plt.subplot(1,2,2).set_title('After Finding Contours', fontsize = font_size);
plt.axis('off')
plt.imshow(contours_img)
plt.show()
```

OUTPUT:

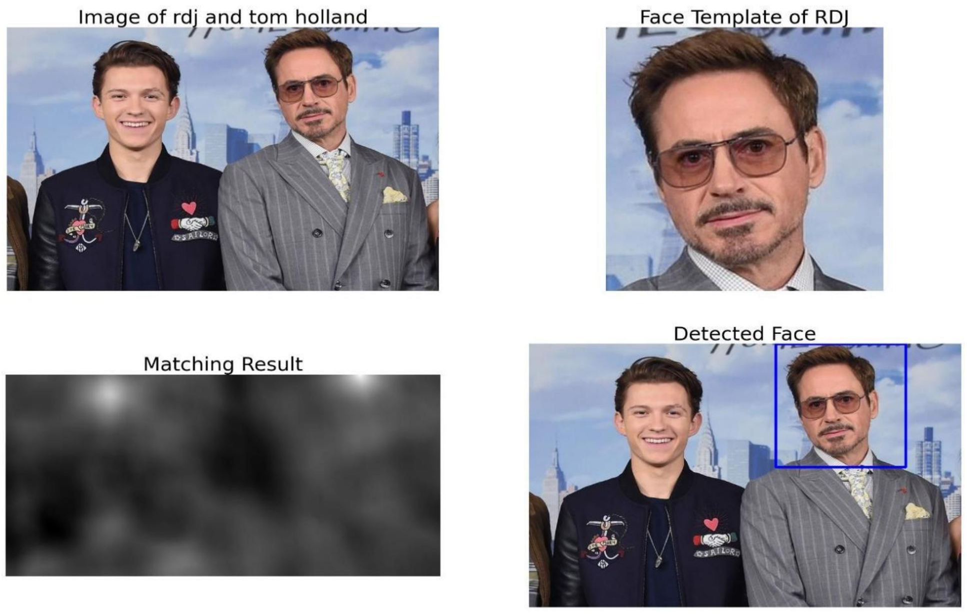
[Image type: RGB   Image Format: .jpg]



5. TEMPLATE MATCHING:

```
w, h, c = template.shape
method = eval('cv2.TM_CCOEFF')
result = cv2.matchTemplate(img, templ = template, method = method)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img, top_left, bottom_right, color = (255, 0, 0), thickness = 3)
plt.figure(figsize=(30, 20))
plt.subplot(2, 2, 1).set_title('Image of Selena Gomez and Taylor Swift',
fontsize = 35); plt.axis('off')
plt.imshow(cv2.cvtColor(img_copy, cv2.COLOR_BGR2RGB))
plt.subplot(2, 2, 2).set_title('Face Template of Selena Gomez', fontsize = 35);
plt.axis('off')
plt.imshow(cv2.cvtColor(template, cv2.COLOR_BGR2RGB))
plt.subplot(2, 2, 3).set_title('Matching Result', fontsize = 35); plt.axis('off')
plt.imshow(result, cmap = 'gray')
plt.subplot(2, 2, 4).set_title('Detected Face', fontsize = 35); plt.axis('off')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

OUTPUT:



RESULT:

Thus the image processing techniques using open cv in computer vision in performed and executed in matlab successfully and verified.

: