**TASK – 1**

**Aim of the task:-**

1) Installation of required software/libraries for image processing.

2) Learn fundamental concepts related to image processing.

3) Opening an image and capturing an image through camera.

4) Video capturing through webcam.

**Inference:-**

* Python language is for image processing. It’s libraries like “numpy” and “OpenCV” are used to carry out the required operation on a image.
* Fundamentals included the functions offered by the OpenCV library, which are importing libraries, reading images, waiting for user input, exiting the program. They are used for following:
  + - **import cv2-** This command asks python to use OpenCV library.
    - **import numpy-** this command asks python to use Numpy library to manipulate matrices as images we use are stored in matrices.
    - **cv2.imread(filename,[flags])-**This command loads an image as a matrix. If we don’t provide any flag value then the image is read and returned as HxWxC matrix where H-height,W-width and C-channel.
    - **cv2.waitKey(time)-** This command waits for a period of time milliseconds for the user to press akey and if pressed then it return the ASCII code of the key pressed.
    - **cv2.destroyAllWindows()-** This command asks python to close all the open windows and exists the program.
* Fundamentals also included the functions offered by OpenCV for capturing image from a camera, which are initializing the camera, reading images from camera, releasing the camera, They are used for following:
  + - **cv2.VideoCapture(camera) –** This command tells python to set up an instance of a VideoCapture class and assigns it to the variable capture. In other words, we need to tell python where we are getting our images from, in this case the number assigned to the camera we need to use.
    - **<capture>.imread() -** This is the command we use to retrieve images from the source we have named as capture.
    - **<capture>.release()** – This command releases the camera that we had initialized when we used the cv2.VideoCapture command. In the absence of this command, we will get errors when we run the program again and try to initialize the same camera without first releasing it.
* Using all these fundamentals opening of an image , capturing an image through camera, video capturing through camera can be done.

**Bibliography:-**

1. Youtube videos for opening of image, capturing image, capturing video through webcam.
2. Various links on Google.
3. Folders and videos provided in the drive of the Task-1.

**Doubts & Technical Problems:-**

Problems were faced during installation of “OpenCV” and “Numpy” libraries, it also required installation of wheels which was unknown file which comes under Numpy and is usable for Numpy library.

**TASK-2**

**Aim of the task:-**

1. Print the rows, columns, channels of the given image.
2. Print the intensity at the beginning centre of the image.
3. Draw contour, find area and perimeter of each contour.
4. Draw contour for maximum area, print the maximum area and print its perimeter.
5. Convert an image to gray image.
6. Convert RGB image to red image.

**Inference:-**

* OpenCV provides a lot of operations to carry out on images as like accessing and modifying the pixel values, accessing image properties, image ROI, splitting and merging channels, making borders for image. They can done through following functions and methods:
  + - * We can access a pixel value by its row and column coordinates. For RGB image, it returns as array of Blue, Green, Red values. For grayscale image, just corresponding intensity is returned.
      * **<read image>.item(int,int,int)-** This command provides a particular color intensity value of R,B or G.
      * **<read image>.itemset((int,int,int),int)-** This command sets a particular color intensity value of R,B or G.
      * **<read image>.shape-** This command returns a tuple of rows, columns and channels hence it provides the shape and size of the image.
      * **<read image>.size-** This command helps in accessing total number of pixels.
      * **<read image>.dtype-** This command returns the datatype of the image.
      * Image ROI(Region Of Image)-For eye detection in images, first face detection is done over the entire image. When a face is obtained, we select the face region alone and search foe eyes inside it instead of searching the whole image. Image ROI is obtained using Numpy indexing.
      * **cv2.split()-** This command is used for splitting the channels.
      * **cv2.merge()-**This command is used for merging the channels.
      * **cv.copyMakeBorder()-** This command is used for padding or making boeder around the image. It takes certain arguments such as ‘src’, ‘top, bottom, left, right’, ‘borderType’ and ‘value’.
* Images contain contours, which are section of the image that are closed. OpenCV provides certain functions on contours to carry out operations such as finding numbers of contours, finding area and perimeter of a particular contour , finding centroid of a contour bounding box of a contour etc. Some of the useful functions are:
  + - **cv2.moments-**  This command helps to calculate some features of contours like COM, area of object etc. This command provides a dictionary of all moment values calculated.
    - **cv2.contourArea()-** This command calculates the area of a particular contour. It takes a argument which specifies the contour number.
    - **cv2.arcLength()-** This command calculates the perimeter of a particular contour. It takes two arguments one is the contour number other is the information about the contour is closed or not if passed “True” then the contour is closed.
    - **cv2.convexHull()-** This command checks a curve for convexity defects and corrects it. It takes arguments such as points(Are the contours we pass into), hull(The output, normally we avoid it), clockwise(if true then it is oriented clockwise otherwise oriented in counter-clockwise), returnPoints (bydefault true, returns the coordinates of hull points).
    - **cv2.isContourConvex()-**  This function is used to check whether the curve is convex or not. It returns True or False.
    - **cv2.boundingRect()-** This command helps in finding the area of a straight rectangle as it doesn’t consider the rotation of the object.
    - **cv2.minAreaRect()-** This function used for bounding rectangle drawn with minimum area, so it considers the rotation also.
    - **cv2.minEnclosingCircle()-** This command is used to find the circumcircle of an object. Center and radius are defined and then passed as argument to function **cv.cirlce(read image, center, radius,(B,G,R),2)**.
    - **cv2.fitEllipse()-** This command is used to fit an ellipse to an object. It returns rotated rectangle in which the ellipse is inscribed. It’s parameters are passed to a function **cv2.ellipse(read image,ellipse,(B,G,R),2).**
    - **cv2.fitLine()-** This command is used to fit a line to a set of points.
    - **cv2.approxPolyDP()-**  This command is used to approximate a contour shape to another shape with less number of vertices depending upon the precision we specifty.

**Bibliography:-**

1. <https://docs.opencv.org/master/d3/df2/tutorial_py_basic_ops.html>
2. <https://docs.opencv.org/trunk//dd/d49/tutorial_py_contour_features.html>
3. <https://drive.google.com/folderview?id=1VLaCtaEu4qQMvXlwhbelDnM3AUwQvBtq>
4. Youtube videos on image processing.

**Doubts & Technical Problems:-**

1. As the functions were new to use a lot of problems were faced during passing the parameters to those functions used.
2. Problems were also faced for choosing a suitable threshold value between o and 255 for the image.