**AI-based Proctoring System for Online Examination (Video Proctoring)**

**TASK 1**

* Read and display digital images and also perform image arithmetic operations and logical operations on the given image and display the result.

To read and view digital - steps:

Import the OpenCV library: First import the OpenCV library into your project. This can be done by adding the appropriate header files and linking the appropriate libraries.

Read digital images: Use the image read functions provided by OpenCV such as cv2.imread() to read digital images from disk files. Specify the path of the image file as input.

Image display: After reading the image, it can be viewed using tools like cv2.imshow() in OpenCV. This will open a window showing the image.

steps to perform arithmetic and arithmetic operations on the given image:

Decide what you want: Select numbers Mathematics or mathematical operations to complete the picture. For example, you can add two images, subtract one image from another, do some operations, and more.

OpenCV uses cv2.add(), cv2.subtract(), cv2.bitwise\_and(), cv2.bitwise\_or(), etc. to do this. It give many functions such as

Request Action: Apply the selected action to the input image or images depending on the specific action you want to perform. Don't forget to pass the appropriate parameters to the OpenCV functions.

Content Presentation: After using the function, you can view the image used by the same function as cv2.imshow(). This will open a new window showing the resulting image.

Make sure you handle any necessary errors and clean up resources properly.

import cv2

img = cv2.imread("sample.jpg")

cv2.imshow("sample”, img)

cv2.waitKey(0)

import cv2

frameWidth = 640

frameHeight = 480

cap = cv2.VideoCapture("test\_ video.mp4")

while True:

success, img = cap.read()

img = cv2.resize(img, (frameWidth, frameHeight))

cv2.imshow("Result", img)

break

import cv2

import numpy as np

image1 = cv2.imread(‘sample1.jpg')

image2 = cv2.imread(‘sample2.jpg')

if image1 is None or image2 is None:

print("Error: Could not open or find the images")

exit()

addition = cv2.add(image1, image2)

subtraction = cv2.subtract(image1, image2)

multiplication = cv2.multiply(image1, image2)

division = cv2.divide(image1, image2)

bitwise\_and = cv2.bitwise\_and(image1, image2)

bitwise\_or = cv2.bitwise\_or(image1, image2)

bitwise\_xor = cv2.bitwise\_xor(image1, image2)

bitwise\_not = cv2.bitwise\_not(image1)

cv2.imshow("Addition", addition)

cv2.imshow("Subtraction", subtraction)

cv2.imshow("Multiplication", multiplication)

cv2.imshow("Division", division)

cv2.imshow("Bitwise AND", bitwise\_and)

cv2.imshow("Bitwise OR", bitwise\_or)

cv2.imshow("Bitwise XOR", bitwise\_xor)

cv2.imshow("Bitwise NOT", bitwise\_not)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Applying geometric transformation for the image.

2. Read image: Use `cv2.imread()` function to read image from a file on disk.

Specify the path of the image file as input.

3. Select Transform: Specify the geometric transformation to be applied to the image. Some of the most common transformations include scaling, rotation, translation, and perspective transformation.

4. Define the transformation matrix: Depending on the selected transformation, you must define the transformation matrix. For example:

- Scaling: Use cv2.getAffineTransform()' function to generate a 2x3 transform matrix.

- Rotation: Use cv2.getRotationMatrix2D()' to generate a 2x3 rotation matrix.

- Translation: Manually create a 2x3 transformation matrix.

- Perspective Transform: Use cv2.getPerspectiveTransform()' function to generate a 3x3 transformation matrix.

5. Request a change: use `cv2'.The warpAffine()` or `cv2.warpPerspective()` function applies a transformation to an image. Pass the input image and transformation matrix to the object as arguments.

6. Screen conversion: Use `cv2'.

The imshow()` function shows the display changes in the window.

7. Manage user input: Use the "cv2.waitKey()" function to keep the window open and let the user see the converted image. This function will wait for the pressed time before closing the window.

Make sure you handle any necessary errors and clean up resources properly.

scaling:

import numpy as np

import cv2 as cv

img = [cv.imread](https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)('sample.jpg')

assert img is not None, "file could not be read, check with os.path.exists()"

res = [cv.resize](https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d)(img,None,fx=2, fy=2, interpolation = cv.INTER\_CUBIC)

#OR

height, width = img.shape[:2]

res = [cv.resize](https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d)(img,(2\*width, 2\*height), interpolation = cv.INTER\_CUBIC)

translation:

import numpy as np

import cv2 as cv

img = [cv.imread](https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)('messi5.jpg', cv.IMREAD\_GRAYSCALE)

assert img is not None, "file could not be read, check with os.path.exists()"

rows,cols = img.shape

M = np.float32([[1,0,100],[0,1,50]])

dst = [cv.warpAffine](https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#ga0203d9ee5fcd28d40dbc4a1ea4451983)(img,M,(cols,rows))

[cv.imshow](https://docs.opencv.org/3.4/df/d24/group__highgui__opengl.html#gaae7e90aa3415c68dba22a5ff2cefc25d)('img',dst)

[cv.waitKey](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ga5628525ad33f52eab17feebcfba38bd7)(0)

[cv.destroyAllWindows](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ga6b7fc1c1a8960438156912027b38f481)()

rotation:

img = [cv.imread](https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)('messi5.jpg', cv.IMREAD\_GRAYSCALE)

assert img is not None, "file could not be read, check with os.path.exists()"

rows,cols = img.shape

# cols-1 and rows-1 are the coordinate limits.

M = [cv.getRotationMatrix2D](https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#gafbbc470ce83812914a70abfb604f4326)(((cols-1)/2.0,(rows-1)/2.0),90,1)

dst = [cv.warpAffine](https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#ga0203d9ee5fcd28d40dbc4a1ea4451983)(img,M,(cols,rows))

* Operations to blur and deblur an image.

import cv2

import numpy as np

image = cv2.imread('sample.jpg')

cv2.imshow('Original Image', image)

cv2.waitKey(0)

Gaussian = cv2.GaussianBlur(image, (7, 7), 0)

cv2.imshow('Gaussian Blurring', Gaussian)

cv2.waitKey(0)

median = cv2.medianBlur(image, 5)

cv2.imshow('Median Blurring', median)

cv2.waitKey(0)

bilateral = cv2.bilateralFilter(image, 9, 75, 75)

cv2.imshow('Bilateral Blurring', bilateral)

cv2.waitKey(0)

cv2.destroyAllWindows()

2. Read image: Use `cv2.imread()` function to read image from a file on disk.

Specify the path of the image file as input.

3. Blur the image: You can use various visualization methods from OpenCV to blur the image. Some visual blurring methods include Gaussian blur, average blur, and binary filtering. Choose the appropriate blurring action according to your needs.

- Gaussian Blur: Gaussian blur is applied using the 'cv2.GaussianBlur()' function. You must specify the kernel size and standard deviation as parameters.

- Median Blur: Use the 'cv2.medianBlur()' function to apply median blur.

Specify the size as input.

- Binary filter: Use the 'cv2.bilateralFilter()' function to apply a binary filter. Specify the area, sigma color, and sigma position as arguments.

4. Show invisible images: Use the 'cv2.imshow()' function to display invisible images in the window.

5. Removing Image Blur: Removing image blur is a difficult task and there is no one-size-fits-all method. However, there are some tricks you can try to make the image visible.

- Deconvolution: You can try deconvolution using functions like "cv2.deconvolve()" or use inverse filters for invisible images.

- Restore Algorithms: OpenCV, Wiener filter, Lucy-Richardson deconvolution etc. It provides many restore algorithms such as You can try algorithms to extract your images.

6. Show deblurred image: Use 'cv2.imshow()' function to display deblurred image in window.

7. Manage user input: Use the "cv2.waitKey()" function to keep the window open and allow the user to see transparent and transparent images.

This function will wait for the pressed time before closing the window.

* Edge Detection – explore various techniques

import cv2

import numpy as np

cap = cv2.VideoCapture(0)

while(1):

ret, frame = cap.read()

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

lower\_red = np.array([30,150,50])

upper\_red = np.array([255,255,180])

mask = cv2.inRange(hsv, lower\_red, upper\_red)

res = cv2.bitwise\_and(frame,frame, mask= mask)

cv2.imshow('Original',frame)

edges = cv2.Canny(frame,100,200)

cv2.imshow('Edges',edges)

k = cv2.waitKey(5) & 0xFF

if k == 27:

break

cap.release()

cv2.destroyAllWindows()

1. Canny Edge Detection:

- The Canny edge detection algorithm is widely used and provides good results.

- Use the `cv2.Canny()` function in OpenCV to apply Canny edge detection. You need to specify the lower and upper threshold values as parameters.

- The algorithm involves steps such as noise reduction, gradient calculation, non-maximum suppression, and hysteresis thresholding.

2. Sobel Operator:

- The Sobel operator is a simple and widely used edge detection technique.

- OpenCV provides the `cv2.Sobel()` function to apply the Sobel operator for edge detection. You can specify the image, the desired depth of the output image, and the order of the derivative in x and y directions as parameters.

- The Sobel operator performs convolution to calculate gradients and identifies edges based on the gradient magnitudes.

3. Scharr Operator:

- The Scharr operator is an improvement over the Sobel operator and provides more accurate results.

- Similar to the Sobel operator, you can use the `cv2.Scharr()` function in OpenCV to apply the Scharr operator for edge detection. Specify the image and the desired depth of the output image as parameters.

4. Laplacian Operator:

- The Laplacian operator is another commonly used technique for edge detection.

- Use the `cv2.Laplacian()` function in OpenCV to apply the Laplacian operator. Specify the image and the desired depth of the output image as parameters.

- The Laplacian operator calculates the second derivative of the image and identifies regions of rapid intensity changes.

5. Gradient Magnitude and Gradient Direction:

- OpenCV provides functions such as `cv2.magnitude()` and `cv2.phase()` to calculate the gradient magnitude and gradient direction of an image, respectively.

- You can apply these functions to the output of the Sobel or Scharr operators to obtain the gradient magnitude and direction, which can provide additional information about the edges.