1. **Affine scale:**

import cv2

import numpy as np

img = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

rows,cols,\_ = img.shape

pts1 = np.float32([[50, 50], [200, 50], [50, 200]])

pts2 = np.float32([[10,100],[200, 50], [100, 250]])

M = cv2.getAffineTransform(pts1,pts2)

dst = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow("Affine Transform", dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Background Subtraction:**

import cv2

import numpy as np

image\_path = "C://Users//mdshu//Pictures//Camera Roll//watch.webp"

image = cv2.imread(image\_path)

x = 100

y = 150

w = 300

h = 250

mask = np.zeros(image.shape[:2], np.uint8)

rect = (x, y, w, h)

cv2.grabCut(image, mask, rect, None, None, 5, cv2.GC\_INIT\_WITH\_RECT)

mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')

result = image \* mask2[:, :, np.newaxis]

cv2.imshow('Background Removed', result)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Gaussian Blur:**

import cv2

import numpy as np

kernel = np.ones((5, 5), np.uint8)

print(kernel)

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

img = cv2 .imread(path)

imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

imgBlur = cv2.GaussianBlur(imgGray, (7, 7), 0)

cv2.imshow("Img Blur", imgBlur)

cv2.waitKey(0)

1. **Canny :**

import cv2

import numpy as np

kernel = np.ones((5,5),np.uint8)

print(kernel)

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

img =cv2.imread(path)

imgGray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

imgBlur = cv2.GaussianBlur(imgGray,(7,7),0)

imgCanny = cv2.Canny(imgBlur,50,100)

cv2.imshow("Img Canny",imgCanny)

cv2.waitKey(0)

1. **Convertion of color space :**

import cv2

image = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//cat.webp")

rgb\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

gray\_image = cv2.cvtColor(rgb\_image, cv2.COLOR\_RGB2GRAY)

cv2.imshow('rgb',rgb\_image)

cv2.imshow('gray',gray\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Counting the spaces:**

import cv2

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

image = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//persons.jpg")

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray\_image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

for (x, y, w, h) in faces:

cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('Detected Faces', image)

num\_faces = len(faces)

print(f"Number of faces detected: {num\_faces}")

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Crop,copy,paste:**

import cv2

import numpy as np

image = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//cat.webp")

img2 = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//cat.webp")

print(image.shape)

cv2.imshow("original", image)

imageCopy = image.copy()

cv2.circle(imageCopy, (100, 100), 30, (255, 0, 0), -1)

cv2.imshow('image', image)

cv2.imshow('image copy', imageCopy)

cropped\_image = image[80:280, 150:330]

cv2.imshow("cropped", cropped\_image)

cv2.imwrite("Cropped Image.jpg", cropped\_image)

dst = cv2.addWeighted(image, 0.5, img2, 0.7, 0)

img\_arr = np.hstack((image, img2))

cv2.imshow('Input Images',img\_arr)

cv2.imshow('Blended Image',dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Dilate :**

import cv2

import numpy as np

kernel = np.ones((5, 5), np.uint8)

print(kernel)

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

img = cv2.imread(path)

img\_dilation = cv2.dilate(img, kernel, iterations=1)

cv2.imshow('Input', img)

cv2.imshow('Dilation', img\_dilation)

cv2.waitKey(0)

1. **Erode:**

import cv2

import numpy as np

kernel = np.ones((5, 5), np.uint8)

print(kernel)

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

img = cv2.imread(path)

img\_erosion = cv2.erode(img, kernel, iterations=1)

cv2.imshow('Input', img)

cv2.imshow('Erosion', img\_erosion)

cv2.waitKey(0)

1. **Extract img from vid :**

import cv2

video\_path = "C://Users//mdshu//Pictures//Camera Roll//Valorant .mp4"

cap = cv2.VideoCapture(video\_path)

if not cap.isOpened():

print("Error opening video file")

exit()

frame\_count = int(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))

frame\_width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

frame\_height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

fps = int(cap.get(cv2.CAP\_PROP\_FPS))

print(f"Frame count: {frame\_count}")

print(f"Frame dimensions: {frame\_width}x{frame\_height}")

print(f"Frames per second: {fps}")

for frame\_number in range(frame\_count):

ret, frame = cap.read()

if not ret:

print(f"Error reading frame {frame\_number}")

break

frame\_filename = f"frame\_{frame\_number:04d}.jpg"

cv2.imwrite(frame\_filename, frame)

print(f"Saved frame {frame\_filename}")

cap.release()

cv2.destroyAllWindows()

1. **Eye detection:**

import cv2

image\_path = "C://Users//mdshu//Pictures//Camera Roll//eye.jpg"

image = cv2.imread(image\_path)

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

eye\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_eye.xml')

eyes = eye\_cascade.detectMultiScale(gray\_image, scaleFactor=1.3, minNeighbors=5)

for (x, y, w, h) in eyes:

cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)

cv2.imshow('Eye Detection', image)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Face Detection:**

import cv2

face\_cascade = cv2.CascadeClassifier("C://Users//mdshu//PycharmProjects//pythonProject3//haarcascade\_frontalface\_default.xml")

img = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//persons.jpg")

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray,1.1,5)

for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow('img', img)

cv2.waitKey(0)

1. **Foreground Subtractor:**

import cv2

import numpy as np

image\_path = "C://Users//mdshu//Pictures//Camera Roll//watch.webp"

image = cv2.imread(image\_path)

x = 100

y = 150

w = 300

h = 250

mask = np.zeros(image.shape[:2], np.uint8)

rect = (x, y, w, h)

cv2.grabCut(image, mask, rect, None, None, 5, cv2.GC\_INIT\_WITH\_RECT)

mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')

result = image \* mask2[:, :, np.newaxis]

cv2.imshow('Foreground Removed', result)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Four corner detection:**

import cv2

import numpy as np

im\_src = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

pts\_src = np.array([[141, 131], [480, 159], [493, 630],[64, 601]])

im\_dst = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

pts\_dst = np.array([[318, 256],[534, 372],[316, 670],[73, 473]])

h, status = cv2.findHomography(pts\_src, pts\_dst)

im\_out = cv2.warpPerspective(im\_src, h, (im\_dst.shape[1],im\_dst.shape[0]))

cv2.imshow("Source Image", im\_src)

cv2.imshow("Destination Image", im\_dst)

cv2.imshow("Warped Source Image", im\_out)

cv2.waitKey(0)

1. **Gray img:**

import cv2

import numpy as np

kernel = np.ones((5,5),np.uint8)

print(kernel)

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

img =cv2.imread(path)

imgGray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

cv2.imshow("GrayScale",imgGray)

cv2.waitKey(0)

1. **Harris Corner detection:**

import cv2

import numpy as np

image\_path =("C://Users//mdshu//Pictures//Camera Roll//download.png")

image = cv2.imread(image\_path)

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

corner\_response = cv2.cornerHarris(gray\_image, blockSize=2, ksize=3, k=0.04)

corner\_response = cv2.dilate(corner\_response, None)

threshold = 0.01 \* corner\_response.max()

image[corner\_response > threshold] = [0, 0, 255]

cv2.imshow('Harris Corners', image)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Histogram equalization:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

image\_path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

equalized\_image = cv2.equalizeHist(image)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(image, cmap='gray')

plt.title('Original Image')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.imshow(equalized\_image, cmap='gray')

plt.title('Equalized Image')

plt.axis('off')

plt.tight\_layout()

plt.show()

1. **Resize img:**

import cv2

import numpy as np

kernel = np.ones((5,5),np.uint8)

img = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

img = cv2.resize(img,(600,600))

cv2.imshow("image",img)

cv2.waitKey(0)

1. **Img Segmentation:**

import cv2

import numpy as np

image = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

\_, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

kernel = np.ones((3, 3), np.uint8)

opening = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, kernel, iterations=2)

sure\_bg = cv2.dilate(opening, kernel, iterations=3)

dist\_transform = cv2.distanceTransform(opening, cv2.DIST\_L2, 5)

\_, sure\_fg = cv2.threshold(dist\_transform, 0.7 \* dist\_transform.max(), 255, 0)

sure\_fg = np.uint8(sure\_fg)

unknown = cv2.subtract(sure\_bg, sure\_fg)

\_, markers = cv2.connectedComponents(sure\_fg)

markers = markers + 1

markers[unknown == 255] = 0

markers = cv2.watershed(image, markers)

image[markers == -1] = [0, 0, 255]

cv2.imwrite('segmented\_image.jpg', image)

1. **Morphological operations:**

import cv2

import numpy as np

img = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg", cv2.IMREAD\_GRAYSCALE)

kernel = np.ones((5,5), np.uint8)

closing = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel)

cv2.imshow("Original", img)

cv2.imshow("Closing", closing)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Object detection:**

import cv2

watch\_cascade = cv2.CascadeClassifier("C://Users//mdshu//PycharmProjects//pythonProject3//watch-cascade.xml")

img = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//images.jpg")

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

watches = watch\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

for (x, y, w, h) in watches:

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.imshow('Watches Detected', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Perspective img:**

import cv2

import numpy as np

img = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

rows,cols,ch = img.shape

pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])

pts2 = np.float32([[100,50],[300,0],[0,300],[300,300]])

M = cv2.getPerspectiveTransform(pts1,pts2)

dst = cv2.warpPerspective(img,M,(cols, rows))

cv2.imshow('Transformed Image', dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Reverse video frame:**

import cv2

cap = cv2.VideoCapture("C://Users//mdshu//Pictures//Camera Roll//Valorant .mp4")

total\_frames = cap.get(cv2.CAP\_PROP\_FRAME\_COUNT)

current\_frame = total\_frames - 1

while current\_frame >= 0:

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, current\_frame)

ret, frame = cap.read()

if not ret:

break

cv2.imshow('Video in Reverse', frame)

if cv2.waitKey(25) & 0xFF == ord('q'):

break

current\_frame -= 1

cap.release()

cv2.destroyAllWindows()

1. **Rotations:**

**Rot 90:**

import cv2

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

src = cv2.imread(path)

window\_name = 'Image'

image = cv2.rotate(src, cv2.ROTATE\_90\_CLOCKWISE)

cv2.imshow(window\_name, image)

cv2.waitKey(0)

**Rot 180:**

import cv2

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

src = cv2.imread(path)

window\_name = 'Image'

image = cv2.rotate(src, cv2.ROTATE\_180)

cv2.imshow(window\_name, image)

cv2.waitKey(0)

**Rot 270:**

import cv2

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

src = cv2.imread(path)

window\_name = 'Image'

image = cv2.rotate(src, cv2.ROTATE\_90\_COUNTERCLOCKWISE)

cv2.imshow(window\_name, image)

cv2.waitKey(0)

1. **Scale img:**

import cv2

import numpy as np

kernel = np.ones((5,5),np.uint8)

path = "C://Users//mdshu//Pictures//Camera Roll//img blur.jpg"

img = cv2.imread(path)

img = cv2.resize(img,(450,750))

cv2.imshow("image",img)

cv2.waitKey(0)

1. **Shaping img:**

import cv2

img = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

x, y = 100, 100

width, height = 200, 150

roi = img[y:y+height, x:x+width]

cv2.imshow('ROI', roi)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Smile detection:**

import cv2

image\_path = "C://Users//mdshu//Pictures//Camera Roll//Dhoni.jpg."

image = cv2.imread(image\_path)

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

smile\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_smile.xml')

smiles = smile\_cascade.detectMultiScale(gray\_image, scaleFactor=1.1, minNeighbors=20)

for (x, y, w, h) in smiles:

cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.imshow('Smile Detection', image)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Text string in img :**

import cv2

image = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

text = (" SPIDY")

position = (150, 300)

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_scale = 1

font\_color = (255, 255, 0)

a=cv2.putText(image, text, position, font, font\_scale, font\_color, 2)

cv2.imshow('output\_image.jpg', a)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Water Mark:**

import cv2

img = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

wm = cv2.imread("C://Users//mdshu//Pictures//Camera Roll//img blur.jpg")

h\_wm, w\_wm = wm.shape[:2]

h\_img, w\_img = img.shape[:2]

center\_x = int(w\_img/2)

center\_y = int(h\_img/2)

top\_y = center\_y - int(h\_wm/2)

left\_x = center\_x - int(w\_wm/2)

bottom\_y = top\_y + h\_wm

right\_x = left\_x + w\_wm

roi = img[top\_y:bottom\_y, left\_x:right\_x]

result = cv2.addWeighted(roi, 1, wm, 0.3, 0)

img[top\_y:bottom\_y, left\_x:right\_x] = result

cv2.imshow("Watermarked Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()