**Experiment-1**

import cv2

image = cv2.imread(r"C:\Users\M.Giri babu\OneDrive\Pictures\Screenshots\Screenshot 2025-06-30 101322.png") # Replace 'sample.jpg' with your image path

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

cv2.imshow('Original Image', image)

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-2**

import cv2

image = cv2.imread(r"C:\Users\M.Giri babu\OneDrive\Pictures\Screenshots\Screenshot 2025-06-30 101322.png")

blurred\_image = cv2.GaussianBlur(image, (15, 15), 0)

cv2.imshow('Original Image', image)

cv2.imshow('Blurred Image', blurred\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-3**

import cv2

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Error: Image not found. Check the file path.")

else:

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

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

cv2.imshow("Original Image", image)

cv2.imshow("Canny Edge Detection", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-4**

import cv2

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Error: Image not found. Check the file path.")

else:

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

equalized = cv2.equalizeHist(gray)

cv2.imshow("Original Grayscale Image", gray)

cv2.imshow("Histogram Equalized Image", equalized)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-5**

import cv2

import matplotlib.pyplot as plt

def analyze\_color\_histogram(image\_path):

image = cv2.imread(image\_path)

if image is None:

print("Error: Image not found.")

return

color = ('b', 'g', 'r')

plt.figure(figsize=(8, 4))

plt.title("Color Histogram")

plt.xlabel("Pixel Intensity")

plt.ylabel("Frequency")

for i, col in enumerate(color):

hist = cv2.calcHist([image], [i], None, [256], [0, 256])

plt.plot(hist, color=col)

plt.xlim([0, 256])

plt.tight\_layout()

plt.show()

# 🔁 Call the function with image path

analyze\_color\_histogram(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

**Experiment-6**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Error: Image not found.")

else:

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

eroded\_image = cv2.erode(image, kernel, iterations=1)

cv2.imshow("Original Image", image)

cv2.imshow("Eroded Image", eroded\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-7**

import cv2

cap = cv2.VideoCapture(r"C:\Users\M.Giri babu\Videos\@Team\_HDT - Demonte Colony 2 (2024) Tamil 720p x264 DD5.1.mkv")

if not cap.isOpened():

print("Error: Cannot open video.")

else:

while True:

ret, frame = cap.read()

if not ret:

break

cv2.imshow("Video", frame)

if cv2.waitKey(30) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**Experiment-8**

import cv2

import numpy as np

image = cv2.imread( r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Error: Image not found.")

else:

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

dilated = cv2.dilate(image, kernel, iterations=1)

cv2.imshow("Original Image", image)

cv2.imshow("Dilated Image", dilated)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-9**

import cv2

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Error: Image not found.")

else:

smaller = cv2.resize(image, None, fx=0.5, fy=0.5, interpolation=cv2.INTER\_AREA)

bigger = cv2.resize(image, None, fx=2.0, fy=2.0, interpolation=cv2.INTER\_LINEAR)

cv2.imshow("Original Image", image)

cv2.imshow("Smaller Image", smaller)

cv2.imshow("Bigger Image", bigger)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-10**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Error: Image not found.")

else:

h, w = image.shape[:2]

src\_points = np.float32([[0,0], [w,0], [0,h], [w,h]])

dst\_points = np.float32([[w//2,0], [w,0], [w//2,h], [w,h]])

matrix = cv2.getPerspectiveTransform(src\_points, dst\_points)

rotated = cv2.warpPerspective(image, matrix, (w, h))

cv2.imshow("Original Image", image)

cv2.imshow("Y-Axis Rotated Image", rotated)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-11**

import cv2

image = cv2.imread("your\_image.jpg") # Replace with your actual image path

if image is None:

print("Image not loaded. Check the path.")

exit()

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

cv2.imshow('Rotated', rotated\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-12**

import cv2

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Image not found.")

else:

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

rotated\_image = cv2.flip(image, 1)

cv2.imwrite('rotated\_image.jpg', rotated\_image)

cv2.imshow('Rotated Image', rotated\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-13**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

if image is None:

print("Image not found. Check the file path.")

exit()

rows, cols = image.shape[:2]

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

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

matrix = cv2.getAffineTransform(pts1, pts2)

result = cv2.warpAffine(image, matrix, (cols, rows))

cv2.imwrite('affine\_transformed.jpg', result)

cv2.imshow('Original', image)

cv2.imshow('Affine Transformed', result)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-14**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Image not loaded. Check the path.")

else:

rows, cols = image.shape[:2]

pts1 = np.float32([[0,0], [cols-1,0], [0,rows-1], [cols-1,rows-1]])

pts2 = np.float32([[50,50], [cols-100,50], [100,rows-100], [cols-150,rows-150]])

matrix = cv2.getPerspectiveTransform(pts1, pts2)

result = cv2.warpPerspective(image, matrix, (cols, rows))

cv2.imshow("Perspective Transform", result)

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.imwrite(r"C:\Users\M.Giri babu\Downloads\perspective.jpg", result)

**Experiment-15**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Image not found. Check the path.")

else:

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

gray = np.float32(gray)

dst = cv2.cornerHarris(gray, 2, 3, 0.04)

dst = cv2.dilate(dst, None)

image[dst > 0.01 \* dst.max()] = [0, 0, 255]

cv2.imshow("Harris Corners", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.imwrite(r"C:\Users\M.Giri babu\Downloads\harris\_corners.jpg", image)

**Experiment-16**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg", 0)

if image is None:

print("Image not found or cannot be opened. Check the path.")

exit()

sobelx = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=3)

sobely = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=3)

sobel = cv2.magnitude(sobelx, sobely)

sobel = cv2.convertScaleAbs(sobel)

success = cv2.imwrite(r"C:\Users\M.Giri babu\Downloads\sobel.jpg", sobel)

if success:

print("Sobel image saved successfully.")

else:

print("Failed to save Sobel image.")

cv2.imshow("Sobel", sobel)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-17**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

watermark = cv2.imread(r"C:\Users\M.Giri babu\Downloads\watermark.png", cv2.IMREAD\_UNCHANGED)

if image is None or watermark is None:

print("Image or watermark not found.")

exit()

wm = cv2.resize(watermark, (100, 100))

if wm.shape[2] == 4:

overlay = wm[:, :, :3]

mask = wm[:, :, 3] / 255.0

h, w = overlay.shape[:2]

x\_offset, y\_offset = 50, 50

for c in range(3):

image[y\_offset:y\_offset+h, x\_offset:x\_offset+w, c] = (

image[y\_offset:y\_offset+h, x\_offset:x\_offset+w, c] \* (1 - mask) +

overlay[:, :, c] \* mask

).astype(np.uint8)

cv2.imwrite(r"C:\Users\M.Giri babu\Downloads\watermarked.jpg", image)

**Experiment-18**

import cv2

image = cv2.imread(r"C:\Users\M.Giri babu\Downloads\Insbright.jpeg")

if image is None:

print("Image not found.")

exit()

roi = image[50:150, 100:200]

image[200:300, 300:400] = roi

cv2.imshow('ROI Copy-Paste', image)

cv2.imwrite('roi\_result.jpg', image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Experiment-19**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", 0)

if image is None:

print("Image not found.")

exit()

\_, binary = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

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

eroded = cv2.erode(binary, kernel, iterations=1)

cv2.imshow("Original", binary)

cv2.imshow("Eroded Image", eroded)

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.imwrite(r"C:\Users\P S PAVAN KUMAR\Downloads\eroded.jpg", eroded)

**Experiment-20**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", 0)

if image is None:

print("Image not found.")

exit()

\_, binary = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

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

dilated = cv2.dilate(binary, kernel, iterations=1)

cv2.imshow("Original", binary)

cv2.imshow("Dilated Image", dilated)

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.imwrite(r"C:\Users\P S PAVAN KUMAR\Downloads\dilated.jpg", dilated)

**Experiment-21**

import cv2

import numpy as np

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", 0)

if image is None:

print("Image not found.")

exit()

\_, binary = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

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

opening = cv2.morphologyEx(binary, cv2.MORPH\_OPEN, kernel)

cv2.imshow("Original", binary)

cv2.imshow("Opened Image", opening)

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.imwrite(r"C:\Users\P S PAVAN KUMAR\Downloads\opened.jpg", opening)

**Experiment-22**

import cv2

import numpy as np

def morphological\_closing(image\_path):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", cv2.IMREAD\_GRAYSCALE)

if image is None:

return

\_, binary = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (5, 5))

closed\_image = cv2.morphologyEx(binary, cv2.MORPH\_CLOSE, kernel)

cv2.imshow("Original Image", image)

cv2.imshow("Binary Image", binary)

cv2.imshow("After Closing", closed\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

morphological\_closing(r"C:\Users\P S PAVAN KUMAR\Downloads\your-image.jpg")

**Experiment-23**

import cv2

import numpy as np

def morphological\_tophat(image\_path):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", cv2.IMREAD\_GRAYSCALE)

if image is None:

return

\_, binary = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (5, 5))

tophat\_image = cv2.morphologyEx(binary, cv2.MORPH\_TOPHAT, kernel)

cv2.imshow("Original Image", image)

cv2.imshow("Binary Image", binary)

cv2.imshow("Top Hat Result", tophat\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

morphological\_tophat(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

**Experiment-24**

import cv2

import numpy as np

def morphological\_blackhat(image\_path):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", cv2.IMREAD\_GRAYSCALE)

if image is None:

return

\_, binary = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (5, 5))

blackhat\_image = cv2.morphologyEx(binary, cv2.MORPH\_BLACKHAT, kernel)

cv2.imshow("Original Image", image)

cv2.imshow("Binary Image", binary)

cv2.imshow("Black Hat Result", blackhat\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

morphological\_blackhat(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

**Experiment-25**

import cv2

def recognize\_watch(template\_path, scene\_path):

template = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", cv2.IMREAD\_GRAYSCALE)

scene = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", cv2.IMREAD\_GRAYSCALE)

if template is None or scene is None:

return

orb = cv2.ORB\_create()

kp1, des1 = orb.detectAndCompute(template, None)

kp2, des2 = orb.detectAndCompute(scene, None)

bf = cv2.BFMatcher(cv2.NORM\_HAMMING, crossCheck=True)

matches = bf.match(des1, des2)

matches = sorted(matches, key=lambda x: x.distance)

result = cv2.drawMatches(template, kp1, scene, kp2, matches[:20], None, flags=2)

cv2.imshow("Watch Recognition", result)

cv2.waitKey(0)

cv2.destroyAllWindows()

recognize\_watch(

r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg",

r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg"

)

**Experiment-26**

import cv2

def reverse\_video(input\_path, output\_path):

cap = cv2.VideoCapture(r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\f9f59bfa2cd14338bb0a3416b7d35569.mp4")

if not cap.isOpened():

print("Error: Cannot open video file.")

return

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

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

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

frames = []

while True:

ret, frame = cap.read()

if not ret:

break

frames.append(frame)

fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')

out = cv2.VideoWriter(output\_path, fourcc, fps, (frame\_width, frame\_height))

for frame in reversed(frames):

out.write(frame)

cap.release()

out.release()

reverse\_video(

r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\f9f59bfa2cd14338bb0a3416b7d35569.mp4",

r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\f9f59bfa2cd14338bb0a3416b7d35569.mp4"

)

**Experiment-27**

import cv2

def detect\_faces(image\_path):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

if image is None:

print("Error: Unable to load image.")

return

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

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

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

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)

cv2.waitKey(0)

cv2.destroyAllWindows()

detect\_faces(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

**Experiment-28**

import cv2

def detect\_vehicles(video\_path):

cap = cv2.VideoCapture(r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\a213910f443248cf80471584c8c4af1c.mp4")

car\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_car.xml")

if not cap.isOpened():

print("Error: Cannot open video file.")

return

while True:

ret, frame = cap.read()

if not ret:

break

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

cars = car\_cascade.detectMultiScale(gray, 1.1, 3)

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

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

cv2.imshow("Vehicle Detection", frame)

if cv2.waitKey(1) == 27:

break

cap.release()

cv2.destroyAllWindows()

**Experiment-29**

import cv2

def detect\_vehicles(video\_path):

vehicle\_cascade = cv2.CascadeClassifier('cars.xml') # Make sure this file exists

if vehicle\_cascade.empty():

print("Failed to load cascade classifier.")

return

cap = cv2.VideoCapture(video\_path)

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

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

vehicles = vehicle\_cascade.detectMultiScale(gray, 1.1, 2)

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

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

cv2.imshow('Vehicle Detection', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

# Call function with video path

detect\_vehicles(r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\a213910f443248cf80471584c8c4af1c.mp4")

**Experiment-30**

import cv2

def detect\_smile(image\_path):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

if image is None:

print("Error: Cannot load image.")

return

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

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

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

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

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

roi\_gray = gray[y:y + h, x:x + w]

roi\_color = image[y:y + h, x:x + w]

smiles = smile\_cascade.detectMultiScale(roi\_gray, 1.8, 20)

for (sx, sy, sw, sh) in smiles:

cv2.rectangle(roi\_color, (sx, sy), (sx + sw, sy + sh), (0, 255, 0), 2)

cv2.imshow("Smile Detection", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

detect\_smile(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

Experiment-31

import cv2

def segment\_image(image\_path, threshold\_value=127):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", cv2.IMREAD\_GRAYSCALE)

if image is None:

print("Error: Unable to load image.")

return

\_, segmented = cv2.threshold(image, threshold\_value, 255, cv2.THRESH\_BINARY)

cv2.imshow("Original Image", image)

cv2.imshow("Segmented Image", segmented)

cv2.waitKey(0)

cv2.destroyAllWindows()

segment\_image(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg", threshold\_value=120)

**Experiment-32**

import numpy as np

import cv2

def create\_colored\_corner\_boxes(width, height):

image = np.ones((25, 15, 3), dtype=np.uint8) \* 255

box\_w, box\_h = width // 10, height // 10

image[0:box\_h, 0:box\_w] = [0, 0, 0] # Top-left: Black

image[0:box\_h, -box\_w:] = [255, 0, 0] # Top-right: Blue

image[-box\_h:, 0:box\_w] = [0, 255, 0] # Bottom-left: Green

image[-box\_h:, -box\_w:] = [0, 0, 255] # Bottom-right: Red

cv2.imshow("Corner Boxes", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Example usage

create\_colored\_corner\_boxes(500, 400)

**Experiment-33**

import cv2

import numpy as np

def create\_image\_with\_rectangle():

width = 612

height = 408

image = np.ones((height, width, 3), dtype=np.uint8) \* 255

top\_left = (width // 4, height // 4)

bottom\_right = (width \* 3 // 4, height \* 3 // 4)

cv2.rectangle(image, top\_left, bottom\_right, (0, 0, 255), 3)

cv2.imshow("Rectangle on White Image", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

create\_image\_with\_rectangle()

**Experiment-34**

import cv2

import numpy as np

def create\_image\_with\_circle():

width = 612

height = 408

image = np.ones((height, width, 3), dtype=np.uint8) \* 255

center = (width // 2, height // 2)

radius = min(width, height) // 4

cv2.circle(image, center, radius, (0, 255, 0), 3)

cv2.imshow("Circle on White Image", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

create\_image\_with\_circle()

**Experiment-35**

import cv2

import numpy as np

def draw\_text\_on\_image(image\_path=None):

text = input("hello")

if image\_path:

image = cv2.imread(image\_path)

if image is None:

print("Error: Cannot load image.")

return

else:

image = np.ones((400, 600, 3), dtype=np.uint8) \* 255

position = (50, 200)

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_scale = 1

color = (0, 0, 255)

thickness = 2

cv2.putText(image, text, position, font, font\_scale, color, thickness)

cv2.imshow("Image with Text", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

draw\_text\_on\_image()

**Experiment-36**

import cv2

import numpy as np

def subtract\_background(image\_path):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

if image is None:

print("Error: Unable to load image.")

return

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

lower\_bound = np.array([35, 40, 40]) # Example: green screen lower HSV

upper\_bound = np.array([85, 255, 255]) # Example: green screen upper HSV

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

mask\_inv = cv2.bitwise\_not(mask)

result = cv2.bitwise\_and(image, image, mask=mask\_inv)

cv2.imshow("Original Image", image)

cv2.imshow("Mask (Background)", mask)

cv2.imshow("Foreground (Background Removed)", result)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Example usage

subtract\_background(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

**Experiment-37**

import cv2

import numpy as np

def subtract\_foreground(image\_path):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

if image is None:

print("Error: Cannot load image.")

return

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

lower\_color = np.array([0, 50, 50]) # Example: Red foreground lower HSV

upper\_color = np.array([10, 255, 255]) # Example: Red foreground upper HSV

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

mask\_inv = cv2.bitwise\_not(mask)

background\_only = cv2.bitwise\_and(image, image, mask=mask\_inv)

cv2.imshow("Original Image", image)

cv2.imshow("Foreground Mask", mask)

cv2.imshow("Image without Foreground", background\_only)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Example usage

subtract\_foreground(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

**Experiment-38**

import cv2

def count\_faces(image\_path):

image = cv2.imread(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

if image is None:

print("Error: Unable to load image.")

return

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

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

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

face\_count = len(faces)

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

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

print(f"Number of faces detected: {face\_count}")

cv2.imshow("Detected Faces", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Example usage

count\_faces(r"C:\Users\BHASKAR GOWD\Downloads\natureimg.jpg")

**Experiment-39**

import cv2

def play\_reverse\_slow(video\_path, slow\_factor=2):

cap = cv2.VideoCapture(r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\a213910f443248cf80471584c8c4af1c.mp4")

if not cap.isOpened():

print("Error: Cannot open video.")

return

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

frames = []

for \_ in range(frame\_count):

ret, frame = cap.read()

if not ret:

break

frames.append(frame)

for frame in reversed(frames):

cv2.imshow("Reverse Slow Motion", frame)

if cv2.waitKey(40 \* slow\_factor) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

# Example usage

play\_reverse\_slow(r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\a213910f443248cf80471584c8c4af1c.mp4", slow\_factor=3)

**Experiment-40**

import cv2

import pytesseract

pytesseract.pytesseract.tesseract\_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

def extract\_text\_from\_video(video\_path):

cap = cv2.VideoCapture(r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\a213910f443248cf80471584c8c4af1c.mp4")

if not cap.isOpened():

print("Error: Cannot open video.")

return

frame\_number = 0

while True:

ret, frame = cap.read()

if not ret:

break

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

text = pytesseract.image\_to\_string(gray)

print(f"Frame {frame\_number}:")

print(text.strip())

print("-" \* 50)

frame\_number += 1

cap.release()

# Call the function

extract\_text\_from\_video(r"C:\Users\BHASKAR GOWD\Downloads\Bhaskar Mobile\Videos\a213910f443248cf80471584c8c4af1c.mp4")