1.Basic Image Handling and Processing video using OpenCV

```
import cv2
from google.colab.patches import cv2_imshow
image=cv2.imread('set image path')
cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2_imshow(gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
resized_image = cv2.resize(image,(200,200))
cv2_imshow(resized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
blurred_image = cv2.GaussianBlur(image, (15, 15), 0)
cv2_imshow(blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
edges = cv2.Canny(gray_image, 100, 200)
cv2_imshow(edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.rectangle(image, (50,50), (300,300), (255, 0, 0), 2)
cv2.line(image, (60,60), (300,300), (0, 0, 255), 2)
cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. Image 2D to 3D Conversion

Date: 14/06/2024

Code:

```
from PIL import Image
import numpy as np
def shift_image(img, depth_img, shift_amount=10):
    # Ensure base image has alpha
    img = img.convert("RGBA")
    data = np.array(img)
    # Ensure depth image is grayscale (for single value)
    depth_img = depth_img.convert("L")
    depth_data = np.array(depth_img)
    deltas = ((depth_data / 255.0) * float(shift_amount)).astype(int)
    # This creates the transparent resulting image.
    # For now, we're dealing with pixel data.
    shifted_data = np.zeros_like(data)
    height, width, _ = data.shape
    for y, row in enumerate(deltas):
        for x, dx in enumerate(row):
            if x + dx < width and x + dx >= 0:
                shifted_data[y, x + dx] = data[y, x]
    # Convert the pixel data to an image.
    shifted_image = Image.fromarray(shifted_data.astype(np.uint8))
    return shifted_image
img = Image.open("C:\\Users\\student\\Desktop\\cube1.jpeg")
depth_img = Image.open("C:\\Users\\student\\Desktop\\cube3.jpeg")
shifted_img = shift_image(img, depth_img, shift_amount=10)
shifted_img.show()
```

3. BASIC MOTION DETECTION AND TRACKING USING CV

```
import cv2
def detect_moving_objects(video_path):
 cap = cv2.VideoCapture(video_path)
 if not cap.isOpened():
 print("Error: Couldn't open the video file.")
 return
 bg_subtractor = cv2.createBackgroundSubtractorMOG2()
 while cap.isOpened():
 ret, frame = cap.read()
 if not ret:
 break
 fg_mask = cv2.medianBlur(bg_subtractor.apply(frame), 5)
 contours, _ = cv2.findContours(fg_mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
 for contour in contours:
 if cv2.contourArea(contour) > 100:r
 M = cv2.moments(contour)
 if M["m00"] != 0:
 cx = int(M["m10"] / M["m00"])
 cy = int(M["m01"] / M["m00"])
 # Draw a dot (small filled circle)
cv2.circle(frame, (cx, cy), 5, (255, 0, 0), 5)
 cv2.imshow('Moving Object Detection', frame)
 if cv2.waitKey(45) & 0xFF == ord('q'):
 break
 cap.release()
 cv2.destroyAllWindows()
video_path = 'input.mp4'
detect_moving_objects(video_path)
```

4. Image Captioning

```
from transformers import VisionEncoderDecoderModel, ViTFeatureExtractor, AutoTokenizer

import torch

from PIL import Image

import warnings

warnings.filterwarnings('ignore')

model = VisionEncoderDecoderModel.from_pretrained("nlpconnect/vit-gpt2-image-captioning")

feature_extractor = ViTFeatureExtractor.from_pretrained("nlpconnect/vit-gpt2-image-captioning")

tokenizer = AutoTokenizer.from_pretrained("nlpconnect/vit-gpt2-image-captioning")

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model.to(device)

max_length = 16

num_beams = 4

gen_kwargs = {"max_length": max_length, "num_beams": num_beams}

def predict_step(image_paths):
  images = []
  for image_path in image_paths:
    i_image = Image.open(image_path)
    if i_image.mode != "RGB":
      i_image = i_image.convert(mode="RGB")
    images.append(i_image)
  pixel_values = feature_extractor(images=images, return_tensors="pt").pixel_values
  pixel_values = pixel_values.to(device)
  output_ids = model.generate(pixel_values, **gen_kwargs)
  preds = tokenizer.batch_decode(output_ids, skip_special_tokens=True)
  preds = [pred.strip() for pred in preds]
  return preds
predict_step(["C:\\Users\\student\\Downloads\\ss.jpg"])
```

5. Build your own vehicle detection model

```
from PIL import
Image import cv2
import numpy as np
import requests
image_url =
'sample.jpg'
image =
Image.open(image_url)
image = image.resize((450,
250)) image.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
image_arr =
np.array(image)
grey = cv2.cvtColor(image_arr,
cv2.COLOR_BGR2GRAY) blur =
cv2.GaussianBlur(grey, (5, 5), 0)
dilated = cv2.dilate(blur, np.ones((3,
3))) dilated = cv2.dilate(blur,
np.ones((3, 3)))
kernel                                                    =
cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
(2,  2))  closing  =  cv2.morphologyEx(dilated,
cv2.MORPH_CLOSE,  kernel)  car_cascade_src  =
"vehicle.xml"
car_cascade                                  =
cv2.CascadeClassifier(car_cascade_src)
```

```
cars                                                =
car_cascade.detectMultiScale(closing,  1.1,
1)
cnt = 0
for (x, y, w, h) in cars:
   cv2.rectangle(image_arr, (x, y), (x + w, y + h), (255, 0, 0), 2)
   cnt += 1
annotated_image =
Image.fromarray(image_arr)
annotated_image.show()
cv2.waitKey(0)
cv2.destroyAllWindo
ws()
```

6. Contour based segmentation

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
image = cv2.imread('path_to_your_image.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
edged = cv2.Canny(blurred, 50, 150)
contours, _ = cv2.findContours(edged, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
contour_image = image.copy()
cv2.drawContours(contour_image, contours, -1, (0, 255, 0), 2)
plt.figure(figsize=(10, 10))
plt.subplot(1, 3, 1)
plt.title('Original Image')
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.subplot(1, 3, 2)
plt.title('Edge Detection')
plt.imshow(edged, cmap='gray')
plt.subplot(1, 3, 3)
plt.title('Contours')
plt.imshow(cv2.cvtColor(contour_image, cv2.COLOR_BGR2RGB))
plt.show()
```

7. REGION BASED SEGMENTATION

Date: 05/07/2024

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.feature import canny
from skimage import data, segmentation, morphology, filters
from skimage.color import rgb2gray, label2rgb
import scipy.ndimage as nd
plt.rcParams["figure.figsize"] = (12, 8)
%matplotlib inline
# Load image and convert to grayscale
rocket = data.rocket()
rocket_wh = rgb2gray(rocket)
# Apply Canny edge detection
edges = canny(rocket_wh)
plt.imshow(edges, interpolation='gaussian')
plt.title('Canny detector')
plt.show()
# Fill regions to perform edge segmentation
fill_im = nd.binary_fill_holes(edges)
plt.imshow(fill_im)
plt.title('Region Filling')
plt.show()
# Compute the elevation map using the Sobel filter
elevation_map = filters.sobel(rocket_wh)
plt.imshow(elevation_map)
plt.title('Elevation Map')
```

```
plt.show()
# Create markers for watershed
markers = np.zeros_like(rocket_wh)
markers[rocket_wh < 0.1171875] = 1 # 30/255
markers[rocket_wh > 0.5859375] = 2 # 150/255
plt.imshow(markers)
plt.title('Markers')
plt.show()
# Perform watershed segmentation
segments = segmentation.watershed(elevation_map, markers)
plt.imshow(segments)
plt.title('Watershed Segmentation')
plt.show()
# Fill holes in the segmented image
segments_filled = nd.binary_fill_holes(segments - 1)
label_rock, _ = nd.label(segments_filled)
# Overlay image with different labels
image_label_overlay = label2rgb(label_rock, image=rocket_wh)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(24, 16), sharey=True)
ax1.imshow(rocket_wh)
ax1.contour(segments_filled, [0.8], linewidths=1.8, colors='w')
ax2.imshow(image_label_overlay)
plt.show()
```

9.Implementation of Shape Detection using Hough Transform

```
import cv2
import numpy as np
def detect_shapes(image_path):
    image = cv2.imread(image_path)
    if image is None:
        print(f"Error: Failed to load image from {image_path}.")
        return
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray_blurred = cv2.medianBlur(gray, 5)
    edges = cv2.Canny(gray, 50, 150, apertureSize=3)
    lines = cv2.HoughLines(edges, 1, np.pi / 180, 150)
    if lines is not None:
        for rho, theta in lines[:, 0]:
            a = np.cos(theta)
            b = np.sin(theta)
            x0 = a * rho
            y0 = b * rho
            x1 = int(x0 + 1000 * (-b))
            y1 = int(y0 + 1000 * (a))
            x2 = int(x0 - 1000 * (-b))
            y2 = int(y0 - 1000 * (a))
            cv2.line(image, (x1, y1), (x2, y2), (0, 0, 255), 2)
    circles = cv2.HoughCircles(gray_blurred, cv2.HOUGH_GRADIENT, 1,
20,
                               param1=50, param2=30, minRadius=1,
maxRadius=40)
```

```
    if circles is not None:
        circles = np.uint16(np.around(circles))
        for i in circles[0, :]:
            cv2.circle(image, (i[0], i[1]), i[2], (0, 255, 0), 2)
            cv2.circle(image, (i[0], i[1]), 2, (0, 0, 255), 3)
    contours, _ = cv2.findContours(edges, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
        epsilon = 0.02 * cv2.arcLength(contour, True)
        approx = cv2.approxPolyDP(contour, epsilon, True)
        if len(approx) == 3:
            # Triangle
            cv2.drawContours(image, [approx], 0, (0, 255, 255), 2)
        elif len(approx) == 4:
            # Rectangle or Square
            cv2.drawContours(image, [approx], 0, (255, 0, 0), 2)
        elif len(approx) > 4:
            # Circle or Ellipse
            area = cv2.contourArea(contour)
            if area > 100:
                cv2.drawContours(image, [approx], 0, (255, 255, 0), 2)
    # Display the result
    cv2.imshow('Shape Detection', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
# Example usage
detect_shapes("C:\\Users\\student\\Downloads\\circle.png")
```

10. Face Detection using Photos

Code:

```
import cv2
import matplotlib.pyplot as plt
image_path = 'C:/Users/student/Downloads/perfect-family-photo-session-by-rebecca-danzenbaker.webp' # Replace with your image path
image = cv2.imread(image_path)
if image is None:
 print(f"Error: Could not load image from {image_path}")
else:
 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, minSize=(30, 30))
 for (x, y, w, h) in faces:
 cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
 image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
 plt.imshow(image_rgb)
 plt.axis('off')
 plt.show()
```

11. Scene Text Detection

Code:

```
!pip install opencv-python pytesseract

import cv2

import pytesseract

pytesseract.pytesseract.tesseract_cmd =

'c:\\Users\\online\\AppData\\Local\\Programs\\Tesseract-
OCR\\tesseract.exe'

def extract_text_from_image(image_path):

image = cv2.imread(image_path)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

text = pytesseract.image_to_string(gray)

return text

image_path = './sample.jpeg'

extracted_text = extract_text_from_image(image_path)

print("Extracted Text:")

print(extracted_text)
```

12 - Road Lane Detection in Autonomous Vehicles

Date: 19/07/2024

Code:

```
import cv2
import numpy as np
def region_of_interest(img, vertices):
 mask = np.zeros_like(img)
 cv2.fillPoly(mask, vertices, 255)
 masked_image = cv2.bitwise_and(img, mask)
 return masked_image
def draw_lines(img, lines):
 if lines is not None:
 for line in lines:
 for x1, y1, x2, y2 in line:
 cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 5)
def process_image(image):
 height, width = image.shape[:2]
 # Convert the image to grayscale
 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
 # Apply Gaussian blur
 blur = cv2.GaussianBlur(gray, (5, 5), 0)
 # Apply Canny edge detector
 edges = cv2.Canny(blur, 50, 150)
 # Define the region of interest
 vertices = np.array([[(50, height), (width//2 - 50, height//2 + 50), (width//2
+ 50,
```

```
height//2 + 50), (width - 50, height)]], dtype=np.int32)
 masked_edges = region_of_interest(edges, vertices)
lines = cv2.HoughLinesP(masked_edges, rho=1, theta=np.pi/180,
threshold=50,
minLineLength=50, maxLineGap=200)
 # Draw the lines on the original image
 line_image = np.zeros_like(image)
 draw_lines(line_image, lines)
 result = cv2.addWeighted(image, 0.8, line_image, 1, 0)
 return result
def main():
 cap =
cv2.VideoCapture("C:\\Users\\student.SCASA1\\Downloads\\travel_road.
mp4") #
Use your own video file or 0 for webcam
 while(cap.isOpened()):
 ret, frame = cap.read()
 if ret:
 result = process_image(frame)
 cv2.imshow('Lane Detection', result)
 if cv2.waitKey(1) & 0xFF == ord('q'):
 break
 else:
 break
 cap.release()
 cv2.destroyAllWindows()
if __name__ == '__main__':
 main()
```

13 - Emotional Recognition through Facial Expressions

Code:

```
pip install deepface

import cv2

import matplotlib.pyplot as plt
from deepface import DeepFace

img = cv2.imread(r'/content/321.jpg')
plt.imshow(img[:, :, : : -1]) plt.show()

result = DeepFace.analyze(img,
                  actions = ['emotion'])


print(result)
```

15 – People Counting

```
import cv2

import numpy as np

net =
cv2.dnn.readNetFromCaffe('C:\\Users\\sudha\\Desktop\\deploy.prototxt',
'C:\\Users\\sudha\\Desktop\\MobileNet-SSD-master\\MobileNet-SSD-
master\mobilenet_iter_73000.caffemodel')

cap = cv2.VideoCapture(0)

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat", "bottle",
"bus",

          "car", "cat", "chair", "cow", "diningtable", "dog", "horse",

          "motorbike", "person", "pottedplant", "sheep", "sofa", "train",
"tvmonitor"]COLORS = np.random.uniform(0, 255, size=(len(CLASSES),
3))

people_count = 0

while True:

    ret, frame = cap.read()

    if not ret:

        break

    (h, w) = frame.shape[:2]

    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
0.007843, (300, 300), 127.5)

    net.setInput(blob)

    detections = net.forward()

    current_count = 0

    for i in np.arange(0, detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence > 0.2:
```

```
            idx = int(detections[0, 0, i, 1])
            if CLASSES[idx] == "person":
                current_count += 1

                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")

                label = "{}: {:.2f}%".format(CLASSES[idx], confidence * 100)
                cv2.rectangle(frame, (startX, startY), (endX, endY),
COLORS[idx], 2)
                y = startY - 15 if startY - 15 > 15 else startY + 15
                cv2.putText(frame, label, (startX, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
    if current_count != people_count:
        people_count = current_count
    cv2.putText(frame, f"People Count: {people_count}", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    cv2.imshow("Frame", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

15 - Count Vehicles in images and video

Code:

```
import cv2
from pyzbar.pyzbar import decode
def decode_qr_from_image(image_path):
    frame = cv2.imread(image_path)
    if frame is None:
        print(f"Failed to load image from {image_path}")
        return
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    decoded_objects = decode(gray_frame)
    for obj in decoded_objects:
        qr_data = obj.data.decode('utf-8')
        print('Data:', qr_data)
        points = obj.polygon
        if len(points) > 4:
            hull = cv2.convexHull(np.array([point for point in points],
dtype=np.float32))
            hull = list(map(tuple, np.squeeze(hull)))
        else:
            hull = points
        n = len(hull)
        for j in range(0, n):
            cv2.line(frame, hull[j], hull[(j + 1) % n], (255, 0, 0), 3)
    cv2.imshow('QR Code Scanner', frame)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
def main():
    image_path = "C:\\Users\\student.SCASA1\\Downloads\\OIP.jpg"  #
Replace with your image path
    decode_qr_from_image(image_path)
if __name__ == "__main__":
    main()
```

16 - qrcode

Code:

```
import cv2
import numpy as np
from pyzbar.pyzbar import decode
from google.colab.patches import cv2_imshow
# Function to decode QR codes
def decode_qr(frame):
    # Convert frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Decode QR codes
    qr_codes = decode(gray)
    return qr_codes

# Load the image file
image_path = 'qr.png'  # Replace with your image path
frame = cv2.imread(image_path)

if frame is None:
    print("Error: Could not read the image file")
    exit()

# Detect QR codes
qr_codes = decode_qr(frame)
for qr_code in qr_codes:
    qr_data = qr_code.data.decode('utf-8')
    points = qr_code.polygon
    if len(points) > 4:
        hull = cv2.convexHull(np.array([point for
point in points],dtype = np.float32))
        cv2.polylines(frame, [hull.astype(np.int32)],
True, (255, 0, 0),3)
    else:
        cv2.polylines(frame, [np.array(points,
dtype=np.int32)], True, (255, 0, 0), 3)
    print("QR Code detected:", qr_data)
cv2_imshow(frame)
cv2.waitKey(0)
cv2.destroyAllWindows()
```