**1.Basic Image Handling and Processing video using OpenCV** (idle)

import cv2

# Load the image from the specified file path

image = cv2.imread("C:\\Users\\GUNTI MOHAN DHANUSH\\Desktop\\pic.jpg")

# Check if the image was loaded correctly

if image is None:

print("Error: Image not found at the specified path.")

else:

# Display the original image

cv2.imshow("Original Image", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Convert the image to grayscale

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

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

cv2.waitKey(0)

cv2.destroyAllWindows()

# Resize the image to 200x200 pixels

resized\_image = cv2.resize(image, (200, 200))

cv2.imshow("Resized Image", resized\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Apply Gaussian blur to the image

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

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

cv2.waitKey(0)

cv2.destroyAllWindows()

# Detect edges using the Canny edge detector

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

cv2.imshow("Edges", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Draw a rectangle on the original image

cv2.rectangle(image, (50, 50), (300, 300), (255, 0, 0), 2)

# Draw a line on the original image

cv2.line(image, (60, 60), (300, 300), (0, 0, 255), 2)

# Display the image with the rectangle and line

cv2.imshow("Image with Rectangle and Line", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**2. Image 2D to 3D Conversion** Date: 14/06/2024 (idle)

**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\\GUNTI MOHAN DHANUSH\\Desktop\\cube3.jpeg")

depth\_img = Image.open("C:\\Users\\GUNTI MOHAN DHANUSH\\Desktop\\cube1.jpeg")

shifted\_img = shift\_image(img, depth\_img, shift\_amount=10)

shifted\_img.show()

**3. BASIC MOTION DETECTION AND TRACKING USING CV**

import cv2

import numpy as np

# 1. Capture video from a file instead of the camera

cap = cv2.VideoCapture("C:\\Users\\GUNTI MOHAN DHANUSH\\Pictures\\DSC\_0794.MOV") # Replace with your video file path

# 2. Get the first frame as reference

ret, frame1 = cap.read()

if not ret:

print("Error: Couldn't read the video file.")

cap.release()

cv2.destroyAllWindows()

exit()

gray1 = cv2.cvtColor(frame1, cv2.COLOR\_BGR2GRAY)

gray1 = cv2.GaussianBlur(gray1, (21, 21), 0)

while cap.isOpened():

# 3. Capture current frame

ret, frame2 = cap.read()

if not ret:

break

gray2 = cv2.cvtColor(frame2, cv2.COLOR\_BGR2GRAY)

gray2 = cv2.GaussianBlur(gray2, (21, 21), 0)

# 4. Calculate the difference between frames

diff = cv2.absdiff(gray1, gray2)

# 5. Threshold the difference to get the motion mask

thresh = cv2.threshold(diff, 25, 255, cv2.THRESH\_BINARY)[1]

thresh = cv2.dilate(thresh, None, iterations=2)

# 6. Find contours of moving objects

contours, \_ = cv2.findContours(thresh.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# 7. Draw bounding boxes around moving objects

for contour in contours:

if cv2.contourArea(contour) < 1000: # Ignore small movements

continue

(x, y, w, h) = cv2.boundingRect(contour)

cv2.rectangle(frame2, (x, y), (x + w, y + h), (0, 255, 0), 2)

# 8. Display the result

cv2.imshow("Motion Detection", frame2)

# 9. Update reference frame

gray1 = gray2

# 10. Exit on 'q' key press

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

break

# Release the video capture object and close windows

cap.release()

cv2.destroyAllWindows()

**4. Image Captioning (collab)**

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(["image.jpg"])

**5. Build your own vehicle detection model (collab)(drive)**

import cv2

import time

import numpy as np

from google.colab.patches import cv2\_imshow

car\_classifier = cv2.CascadeClassifier('haarcascade\_car.xml')

cap = cv2.VideoCapture('cars.mp4')

while cap.isOpened():

time.sleep(.05)

ret, frame = cap.read()

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

cars = car\_classifier.detectMultiScale(gray, 1.4, 2)

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

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

cv2\_imshow(frame)

if cv2.waitKey(1) ==13:

break

cap.release()

cv2.destroyAllWindows()

**6.** **Contour based segmentation (**idle)

import cv2

import numpy as np

import matplotlib.pyplot as plt

image = cv2.imread("C:\\Users\\GUNTI MOHAN DHANUSH\\Desktop\\pic.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()

**. REGION BASED SEGMENTATION (idle)**

import cv2

import numpy as np

from collections import deque

def region\_growing(image, seed\_point, threshold):

# Initialize the output segmented image

segmented\_image = np.zeros\_like(image)

# Get the dimensions of the image

height, width = image.shape[:2]

# Initialize the list of pixels to visit

to\_visit = deque([seed\_point])

# Initialize a list to keep track of visited pixels

visited = np.zeros((height, width), dtype=bool)

# Get the intensity value of the seed point

seed\_intensity = image[seed\_point[1], seed\_point[0]]

while to\_visit:

x, y = to\_visit.popleft()

# If the pixel has already been visited, skip it

if visited[y, x]:

continue

# Mark the pixel as visited

visited[y, x] = True

# Calculate the intensity difference

intensity = image[y, x]

intensity\_diff = np.abs(intensity - seed\_intensity)

# If the intensity difference is within the threshold, add the pixel to the region

if intensity\_diff < threshold:

segmented\_image[y, x] = 255

# Add neighboring pixels to the list to visit

if x > 0: to\_visit.append((x - 1, y))

if x < width - 1: to\_visit.append((x + 1, y))

if y > 0: to\_visit.append((x, y - 1))

if y < height - 1: to\_visit.append((x, y + 1))

return segmented\_image

def main(image\_path, seed\_point, threshold):

# Load the image in grayscale

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

# Apply region growing segmentation

segmented\_image = region\_growing(image, seed\_point, threshold)

# Display the original and segmented images

cv2.imshow("Original Image", image)

cv2.imshow("Segmented Image", segmented\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Example usage

main("C:\\Users\\GUNTI MOHAN DHANUSH\\Desktop\\pic.jpg", seed\_point=(100, 100), threshold=15)

**Implementation of Shape Detection using Hough Transform (idle)**

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\\GUNTI MOHAN DHANUSH\\Desktop\\apple.png")

**10. Face Detection using Photos (idle)**

import cv2

from PIL import Image, ImageDraw

import matplotlib.pyplot as plt

import numpy as np

image\_path = "C:\\Users\\GUNTI MOHAN DHANUSH\\Desktop\\pic.jpg"

img = cv2.imread(image\_path)

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

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

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

pil\_image = Image.open(image\_path)

draw = ImageDraw.Draw(pil\_image)

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

draw.rectangle([(x, y), (x+w, y+h)], outline="red", width=2)

plt.figure(figsize=(8, 6))

plt.imshow(pil\_image)

plt.axis('off')

plt.show()

**11. Scene Text Detection (collab)**

!apt-get install tesseract-ocr

!pip install pytesseract

import cv2

import pytesseract

image\_path = "/content/text.png"

image = cv2.imread(image\_path)

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

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

print("Extracted Text:")

print(text)

**12 - Road Lane Detection in Autonomous Vehicles (collab)**

#Road lane

import cv2

from google.colab.patches import cv2\_imshow

import numpy as np

import time

cap = cv2.VideoCapture('/content/road\_lane.mp4')

while cap.isOpened():

time.sleep(0.05)

ret, frame = cap.read()

if not ret:

break

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

edges = cv2.Canny(gray, 50, 150)

lines = cv2.HoughLinesP(edges, 1, np.pi/180, 50, minLineLength=300, maxLineGap=10)

if lines is not None:

for line in lines:

x1, y1, x2, y2 = line[0]

cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 5)

cv2\_imshow(frame)

cap.release()

cv2.destroyAllWindows()

**13 - Emotional Recognition through Facial Expressions (collab)**

import cv2

from deepface import DeepFace

from google.colab.patches import cv2\_imshow

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

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

cv2\_imshow(image)

print(emotion)

**16- qrcode (collab)**

!pip install pyqrcode

!pip install pypng

!pip install IPython

import pyqrcode

import png

from pyqrcode import QRCode

from IPython.display import Image

s = "Mr Programmer github Link - https://github.com/dashboard"

url = pyqrcode.create(s)

url.png('download.png', scale=6)

Image(filename='download.png')

**15 Count Vehicles in images and video (idle with files)**

**import cv2**

**import numpy as np**

**# Load YOLO**

**net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")**

**layer\_names = net.getLayerNames()**

**output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]**

**# Load Image**

**image = cv2.imread("cars.png")**

**height, width, channels = image.shape**

**# Prepare the image for YOLO**

**blob = cv2.dnn.blobFromImage(image, 0.00392, (416, 416), (0, 0, 0), True, crop=False)**

**net.setInput(blob)**

**outs = net.forward(output\_layers)**

**# Initialize lists**

**class\_ids = []**

**confidences = []**

**boxes = []**

**# Process detection results**

**for out in outs:**

**for detection in out:**

**scores = detection[5:]**

**class\_id = np.argmax(scores)**

**confidence = scores[class\_id]**

**if confidence > 0.5: # Adjust the confidence threshold as needed**

**center\_x = int(detection[0] \* width)**

**center\_y = int(detection[1] \* height)**

**w = int(detection[2] \* width)**

**h = int(detection[3] \* height)**

**x = int(center\_x - w / 2)**

**y = int(center\_y - h / 2)**

**boxes.append([x, y, w, h])**

**confidences.append(float(confidence))**

**class\_ids.append(class\_id)**

**# Apply Non-Max Suppression**

**indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)**

**# Draw bounding boxes**

**vehicle\_count = 0**

**for i in range(len(boxes)):**

**if i in indexes:**

**x, y, w, h = boxes[i]**

**label = str(class\_id)**

**confidence = confidences[i]**

**color = (0, 255, 0) # Green color for bounding box**

**cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)**

**vehicle\_count += 1**

**# Show the image**

**cv2.putText(image, f"Vehicles: {vehicle\_count}", (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2)**

**cv2.imshow("Image", image)**

**cv2.waitKey(0)**

**cv2.destroyAllWindows()**