BSD2513 ARTIFICIAL INTELLIGENCE

LAB REPORT 4

NAME : TEAN JIN HE

MATRIC ID : SD21063

SECTION : 02G

Questions 1 : General Knowledge

Two examples of the applications of computer vision in real-world problems are:Computer vision is a field of artificial intelligence (AI) and computer science that enables computers and systems to extract meaningful information from digital images, videos and other visual inputs for taking actions or making recommendations based on that information. Its goal is to mimic human visual perception and provide machines with the ability to analyze and make decisions based on visual data.

Two examples of the applications of computer vision in real-world problems are:

i) Autonomous Vehicles

Computer Vision plays a crucial role in enabling autonomous vehicles to perceive and navigate their environment. By using cameras and other sensors, these vehicles capture visual information and process it in real-time to make decisions such as detecting objects on the road, identifying traffic signs and signals, and understanding the overall scene. This technology is essential for ensuring the safety and efficiency of self-driving cars. One notable example is the development of autonomous vehicles by companies like Waymo (formerly the Google Self-Driving Car Project). Waymo's vehicles utilize advanced Computer Vision techniques to interpret their surroundings and make driving decisions based on visual input.

Reference: "Waymo: Waymo's Safety Drivers" - Waymo, https://waymo.com/safety/

ii) Medical Image Analysis

Computer Vision is widely used in medical applications for analyzing various types of medical images, such as X-rays, MRI scans, and histopathology slides. It aids in the diagnosis, treatment, and monitoring of diseases by automating image interpretation and providing quantitative measurements. For instance, in cancer diagnosis, Computer Vision algorithms can detect and classify tumors, segment regions of interest, and analyze tissue structures. This helps radiologists and pathologists in making more accurate and efficient diagnoses. One notable research paper in this domain is "Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis" by Coudray et al. (2018), which demonstrates the application of deep learning-based Computer Vision techniques in histopathology analysis.

```
Reference: Coudray, N., Ocampo, P. S., Sakellaropoulos, T., Narula, N., Snuderl,
M., Fenyö, D., & Moreira, A. L. (2018). Deep learning as a tool for increased
accuracy and efficiency of histopathological diagnosis. Scientific reports, 8(1),
1-11.
```

Question 2

Python: Image Recognition

In [1]:

```python
# 1. Create simple object detection and tracking code using Haar Cascades classifier

import cv2

# Load the pre-trained Haar cascade classifier for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalfac

# Initialize video capture from the webcam
video_capture = cv2.VideoCapture(0)

while True:
    # Read the current frame from the video stream
    ret, frame = video_capture.read()

    # Convert the frame to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Perform face detection
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, mir

    # Draw rectangles around the detected faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Display the resulting frame
    cv2.imshow('Video', frame)

    # Break the loop if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video capture and close the windows
video_capture.release()
cv2.destroyAllWindows()
```

In [1]:

```python
import cv2

# Define a class to handle object tracking related functionality
class ObjectTracker(object):
    def __init__(self, scaling_factor=0.5):
        # Initialize the video capture object
        self.cap = cv2.VideoCapture(0)

        # Capture the frame from the webcam
        _, self.frame = self.cap.read()

        # Scaling factor for the captured frame
        self.scaling_factor = scaling_factor

        # Resize the frame
        self.frame = cv2.resize(self.frame, None,
                fx=self.scaling_factor, fy=self.scaling_factor,
                interpolation=cv2.INTER_AREA)

        # Create a window to display the frame
        cv2.namedWindow('Object Tracker')

        # Set the mouse callback function to track the mouse
        cv2.setMouseCallback('Object Tracker', self.mouse_event)

        # Initialize variable related to rectangular region selection
        self.selection = None

        # Initialize variable related to starting position
        self.drag_start = None

        # Initialize variable related to the state of tracking
        self.tracking_state = 0

        # Load the pre-trained Haar cascade classifiers for face and eye detection
        self.face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcasca
        self.eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascad

    # Define a method to track the mouse events
    def mouse_event(self, event, x, y, flags, param):
        # Convert x and y coordinates into 16-bit numpy integers
        x, y = np.int16([x, y])

        # Check if a mouse button down event has occurred
        if event == cv2.EVENT_LBUTTONDOWN:
            self.drag_start = (x, y)
            self.tracking_state = 0

        # Check if the user has started selecting the region
        if self.drag_start:
            if flags & cv2.EVENT_FLAG_LBUTTON:
                # Extract the dimensions of the frame
                h, w = self.frame.shape[:2]

                # Get the initial position
                xi, yi = self.drag_start

                # Get the max and min values
                x0, y0 = np.maximum(0, np.minimum([xi, yi], [x, y]))
                x1, y1 = np.minimum([w, h], np.maximum([xi, yi], [x, y]))

                # Reset the selection variable
                self.selection = None

                # Finalize the rectangular selection
                if x1-x0 > 0 and y1-y0 > 0:
                    self.selection = (x0, y0, x1, y1)

            else:
                # If the selection is done, start tracking
                self.drag_start = None
                if self.selection is not None:
                    self.tracking_state = 1

    # Method to start tracking the object
    def start_tracking(self):
        # Iterate until the user presses the Esc key
        while True:
            # Capture the frame from webcam
            _, self.frame = self.cap.read()

            # Resize the input frame
            self.frame = cv2.resize(self.frame, None,
                    fx=self.scaling_factor, fy=self.scaling_factor,
                    interpolation=cv2.INTER_AREA)

            # Create a copy of the frame
            vis = self.frame.copy()

            # Convert the frame to grayscale for face and eye detection
            gray = cv2.cvtColor(vis, cv2.COLOR_BGR2GRAY)

            # Check if the user has selected the region
            if self.selection:
                # Extract the coordinates of the selected rectangle
                x0, y0, x1, y1 = self.selection

                # Extract the tracking window
                self.track_window = (x0, y0, x1-x0, y1-y0)

                # Extract the region of interest (ROI) for face detection
                roi_gray = gray[y0:y1, x0:x1]

                # Perform face detection within the ROI
                faces = self.face_cascade.detectMultiScale(roi_gray)

                # Iterate over detected faces
                for (fx, fy, fw, fh) in faces:
                    # Draw a rectangle around the face
                    cv2.rectangle(vis, (fx + x0, fy + y0), (fx + x0 + fw, fy + y0 +

                    # Extract the ROI for eye detection
                    roi_gray_face = gray[fy + y0:fy + y0 + fh, fx + x0:fx + x0 + fw

                    # Perform eye detection within the face region
                    eyes = self.eye_cascade.detectMultiScale(roi_gray_face)

                    # Iterate over detected eyes
                    for (ex, ey, ew, eh) in eyes:
                        # Draw a rectangle around the eyes (relative to the face)
                        cv2.rectangle(vis, (ex + fx + x0, ey + fy + y0), (ex + fx +

            # Check if the system is in the "tracking" mode
            if self.tracking_state == 1:
                # Reset the selection variable
                self.selection = None

                # Convert the frame to grayscale for face detection
                gray = cv2.cvtColor(vis, cv2.COLOR_BGR2GRAY)

                # Perform face detection
                faces = self.face_cascade.detectMultiScale(gray)

                # Iterate over detected faces
                for (x, y, w, h) in faces:
                    # Draw a rectangle around the face
                    cv2.rectangle(vis, (x, y), (x + w, y + h), (0, 255, 0), 2)

                    # Extract the ROI for eye detection
                    roi_gray_face = gray[y:y + h, x:x + w]

                    # Perform eye detection within the face region
                    eyes = self.eye_cascade.detectMultiScale(roi_gray_face)

                    # Iterate over detected eyes
                    for (ex, ey, ew, eh) in eyes:
                        # Draw a rectangle around the eyes (relative to the face)
                        cv2.rectangle(vis, (ex + x, ey + y), (ex + x + ew, ey + y +

            # Show the output live video
            cv2.imshow('Object Tracker', vis)

            # Stop if the user hits the 'Esc' key
            c = cv2.waitKey(5)
            if c == 27:
                break

        # Close all the windows
        cv2.destroyAllWindows()

if __name__ == '__main__':
    # Start the tracker
    ObjectTracker().start_tracking()
```

```
-----------------------------------------------------------------------
----
NameError                                 Traceback (most recent call l
ast)
~\AppData\Local\Temp\ipykernel_21276\1770577794.py in mouse_event(self,
event, x, y, flags, param)
     40     def mouse_event(self, event, x, y, flags, param):
     41         # Convert x and y coordinates into 16-bit numpy integer
s
---> 42         x, y = np.int16([x, y])
     43 
     44         # Check if a mouse button down event has occurred

NameError: name 'np' is not defined
```

In [1]:

```python
# 2. Create simple code for face and eye detection and tracking.

import cv2
import numpy as np

# Load the Haar cascade files for face and eye
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalfac
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

# Check if the face cascade file has been loaded correctly
if face_cascade.empty():
            raise IOError('Unable to load the face cascade classifier xml file')

# Check if the eye cascade file has been loaded correctly
if eye_cascade.empty():
            raise IOError('Unable to load the eye cascade classifier xml file')

# Initialize the video capture object
cap = cv2.VideoCapture(0)

# Define the scaling factor
ds_factor = 0.5

# Iterate until the user hits the 'Esc' key
while True:
    # Capture the current frame
    _, frame = cap.read()

    # Resize the frame
    frame = cv2.resize(frame, None, fx=ds_factor, fy=ds_factor, interpolation=cv2.IN

    # Convert to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Run the face detector on the grayscale image
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    # For each face that's detected, run the eye detector
    for (x,y,w,h) in faces:
        # Extract the grayscale face ROI
        roi_gray = gray[y:y+h, x:x+w]

        # Extract the color face ROI
        roi_color = frame[y:y+h, x:x+w]

        # Run the eye detector on the grayscale ROI
        eyes = eye_cascade.detectMultiScale(roi_gray)

        # Draw circles around the eyes
        for (x_eye,y_eye,w_eye,h_eye) in eyes:
            center = (int(x_eye + 0.5*w_eye), int(y_eye + 0.5*h_eye))
            radius = int(0.3 * (w_eye + h_eye))
            color = (0, 255, 0)
            thickness = 3
            cv2.circle(roi_color, center, radius, color, thickness)

    # Display the output
    cv2.imshow('Eye Detector', frame)

    # Check if the user hit the 'Esc' key
    c = cv2.waitKey(1)
    if c == 27:
        break

# Release the video capture object
cap.release()

# Close all the windows
cv2.destroyAllWindows()
```

In [1]:

```python
import cv2

# Load the pre-trained Haar cascade classifiers for face and eye detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalfac
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

# Initialize video capture from the webcam
video_capture = cv2.VideoCapture(0)

while True:
    # Read the current frame from the video stream
    ret, frame = video_capture.read()

    # Convert the frame to grayscale for face and eye detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Perform face detection
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, mir

    # Iterate over detected faces
    for (x, y, w, h) in faces:
        # Draw a rectangle around the face
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        # Extract the region of interest (ROI) for eyes within the face rectangle
        roi_gray = gray[y:y + h, x:x + w]
        roi_color = frame[y:y + h, x:x + w]

        # Perform eye detection within the face region
        eyes = eye_cascade.detectMultiScale(roi_gray)

        # Iterate over detected eyes
        for (ex, ey, ew, eh) in eyes:
            # Draw a rectangle around the eyes (relative to the face)
            cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (255, 0, 0), 2)

    # Display the resulting frame
    cv2.imshow('Video', frame)

    # Break the loop if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video capture and close the windows
video_capture.release()
cv2.destroyAllWindows()
```

In [ ]:

```python

```