**Programme: Higher Diploma in Artificial Intelligence and Robotics**

**Programme code: EG114728**

|  |  |  |
| --- | --- | --- |
| **Official Use** | | |
|  | Full Mark | Mark |
| **Total** | 100% |  |

**Module: AI and Programming**

**Module Code: MBS 3523**

**Assessment: Assignment 2**

**Due Date: 13 April 2021**

|  |  |  |
| --- | --- | --- |
| **Name:**  **Lee Wang Hei** | **Class:**  **EG114728-1A** | **Student no.:**  **200110289** |

**NOTES:**

* **Answer all questions.**
* **Full mark of this paper is 100.**
* **Attach your programs with this paper.**

**Submission deadline: 13 April 2021 5:00 pm**

“*I declare that this assessment is my own work and was not copied from any other person”*

*Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**NOTE:**

* For this assignment (Question 1 -3), you are required to demonstrate the operation of your programs with specific tasks/requirements that will be given during the demonstration section!
* You may bring your own notebook computer or use the computer in lab (webcam will be provided).
* You have 3 minutes for each question demonstration. That means a total of 9 minutes for the whole assignment. No extra time will be given unless there is a technical problem. Be aware and well prepare for the demonstration.
* The demonstration period is on **13 April 2021, 13:30 – 17:00**.
* If you cannot demonstrate any code, a reassessment will be arranged on 20 April 2021, 15:30 – 17:30. The maximum marks from the reassessment will be 40!

|  |
| --- |
| **Question 1 (30%)** |
| With reference to ***OpenCV-Ex5-FaceRec.py***, demonstrate:   1. your code with your own face with your name on top left corner of the bounding box; 2. how you can add a new known person (the photo of the known person will be given during demonstration time) and recognize him/her. When the code is run, the new known person’s name should be displayed on top left corner of the bounding box.   import cv2 print(cv2.\_\_version\_\_) import face\_recognition print(face\_recognition.\_\_version\_\_) import numpy as np import os  encodes = [] names = []  image\_dir='Images\_Known' for root, dirs, files in os.walk(image\_dir):  print(files)  for file in files:  fullPath = os.path.join(root, file)  print(fullPath)  name = os.path.splitext(file)[0]  print(name)  imgKnown = face\_recognition.load\_image\_file(fullPath)  encodeKnown = face\_recognition.face\_encodings(imgKnown)[0]  encodes.append(encodeKnown)  names.append(name) print(names)  capture = cv2.VideoCapture(0) capture.set(3, 640) capture.set(4, 480)  font=cv2.FONT\_HERSHEY\_SIMPLEX  while True:  success, img = capture.read()  imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  # From webcam, read and find unknown images, encode, and compare to known faces  faceLocsCam = face\_recognition.face\_locations(imgRGB)  encodesCam = face\_recognition.face\_encodings(imgRGB, faceLocsCam)   # Compare faces in webcam to encoded face  for (top, right, bottom, left), encodeCam in zip(faceLocsCam, encodesCam):  name = "unknown"  results = face\_recognition.compare\_faces(encodes, encodeCam)  faceDist = face\_recognition.face\_distance(encodes, encodeCam)  print(faceDist)  match\_index = np.argmin(faceDist)  if results[match\_index]:  name = names[match\_index]  cv2.rectangle(img, (left, top), (right, bottom), (255, 0, 0), 2)  cv2.rectangle(img, (left, top), (right, top-30), (255, 0, 0), -1)  cv2.putText(img, name, (left, top-10), font, .75, (0, 255, 255), 2)   cv2.imshow('Frame', img)  if cv2.waitKey(1) == 27:  break  capture.release() cv2.destroyAllWindows() |

|  |
| --- |
| **Question 2 (30%)** |
| With reference to ***OPENCV-17-colorTracking2.py*** and ***OpenCV-Ex6-Track2Colors.py***, demonstrate   1. the tracking result when ONE random color ball is given; 2. the tracking result when TWO random color balls are given.   import cv2 import numpy as np  def nothing(): pass  cv2.namedWindow('Trackbars')  cv2.createTrackbar('HueLow','Trackbars',24,179,nothing) cv2.createTrackbar('HueHigh','Trackbars',86,179,nothing) cv2.createTrackbar('SatLow','Trackbars',139,255,nothing) cv2.createTrackbar('SatHigh','Trackbars',255,255,nothing) cv2.createTrackbar('ValLow','Trackbars',122,255,nothing) cv2.createTrackbar('ValHigh','Trackbars',255,255,nothing)  ############################################## # Below lines are added for sencond color tracking cv2.namedWindow('Trackbars2')  cv2.createTrackbar('HueLow2','Trackbars2',24,179,nothing) cv2.createTrackbar('HueHigh2','Trackbars2',86,179,nothing) cv2.createTrackbar('SatLow2','Trackbars2',139,255,nothing) cv2.createTrackbar('SatHigh2','Trackbars2',255,255,nothing) cv2.createTrackbar('ValLow2','Trackbars2',122,255,nothing) cv2.createTrackbar('ValHigh2','Trackbars2',255,255,nothing) ###############################################  # Set up webcam cam = cv2.VideoCapture(0) # cam.set(3,640) # cam.set(4,480)  # Start capturing and show frames on window while True:  success, img = cam.read()   img2 = img.copy()   hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)  hueLow = cv2.getTrackbarPos('HueLow','Trackbars')  hueHigh = cv2.getTrackbarPos('HueHigh', 'Trackbars')  satLow = cv2.getTrackbarPos('SatLow', 'Trackbars')  satHigh = cv2.getTrackbarPos('SatHigh', 'Trackbars')  valLow = cv2.getTrackbarPos('ValLow', 'Trackbars')  valHigh = cv2.getTrackbarPos('ValHigh', 'Trackbars')   ##############################################  # Below lines are added for sencond color finding  hsv2 = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)  hueLow2 = cv2.getTrackbarPos('HueLow2','Trackbars2')  hueHigh2 = cv2.getTrackbarPos('HueHigh2', 'Trackbars2')  satLow2 = cv2.getTrackbarPos('SatLow2', 'Trackbars2')  satHigh2 = cv2.getTrackbarPos('SatHigh2', 'Trackbars2')  valLow2 = cv2.getTrackbarPos('ValLow2', 'Trackbars2')  valHigh2 = cv2.getTrackbarPos('ValHigh2', 'Trackbars2')  ##############################################    FGmask = cv2.inRange(hsv, (hueLow,satLow,valLow),(hueHigh,satHigh,valHigh))  cv2.imshow('FGmask',FGmask)  final = cv2.bitwise\_and(img,img,mask=FGmask)   ##############################################  # Below lines are added for sencond color masking  FGmask2 = cv2.inRange(hsv2, (hueLow2, satLow2, valLow2), (hueHigh2, satHigh2, valHigh2))  cv2.imshow('FGmask2', FGmask2)  final2 = cv2.bitwise\_and(img2, img2, mask=FGmask2)  ##############################################   FGmaskAdded = cv2.add(FGmask,FGmask2)   contours, hierarchy = cv2.findContours(FGmaskAdded,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_SIMPLE)  contours = sorted(contours,key=lambda x:cv2.contourArea(x),reverse=True)  for cnt in contours:  area = cv2.contourArea(cnt)  (x,y,w,h) = cv2.boundingRect(cnt)  if area > 100:  #cv2.drawContours(img,[cnt],0,(255,0,0),3)  cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),3)   cv2.imshow('Final', final)  cv2.imshow('Final2', final2)  cv2.imshow('Frame', img)  if cv2.waitKey(1) == 27:  break  # cam.release() cv2.destroyAllWindows() |

|  |
| --- |
| **Question 3 (40%)** |
| With reference to ***OpenCV-Ex8-YOLO-track2objects.py***, demonstrate:   1. the tracking result, with correct names shown, when any objects within the COCO namelist are seen from the webcam; 2. the tracking result, with correct name shown, when ONE randomly selected object is specified; 3. the tracking result, with correct names shown, when TWO randomly selected objects are specified.   import cv2 import numpy as np  confThreshold = 0.4  cap = cv2.VideoCapture(0)  # Create an empty list - classes[] and point the classesFile to 'coco80.names' classesFile = 'coco80.names' classes = [] # Load all classes in coco80.names into classes[] with open(classesFile, 'r') as f:  classes = f.read().splitlines()  print(classes)  print(len(classes))  # Load the configuration and weights file # You need to download the weights and cfg files from https://pjreddie.com/darknet/yolo/ net = cv2.dnn.readNetFromDarknet('yolov3.cfg','yolov3.weights') # Use OpenCV as backend and use CPU net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_OPENCV) net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CPU)   while True:  success, img = cap.read()  height, width, ch = img.shape   blob = cv2.dnn.blobFromImage(img, 1 / 255, (320, 320), (0, 0, 0), swapRB=True, crop=False)  net.setInput(blob)   layerNames = net.getLayerNames()  print(layerNames)   output\_layers\_names = net.getUnconnectedOutLayersNames()  print(output\_layers\_names)   LayerOutputs = net.forward(output\_layers\_names)  print(len(LayerOutputs))  # print(LayerOutputs[0].shape)  # print(LayerOutputs[1].shape)  # print(LayerOutputs[2].shape)  # print(LayerOutputs[0][0])    bboxes = [] # array for all bounding boxes of detected classes  confidences = [] # array for all confidence values of matching detected classes  class\_ids = [] # array for all class IDs of matching detected classes   for output in LayerOutputs:  for detection in output:  scores = detection[5:] # omit the first 5 values  class\_id = np.argmax(scores) # find the highest score ID out of 80 values which has the highest confidence value  confidence = scores[class\_id]  if confidence > confThreshold:  center\_x = int(detection[0]\*width) #YOLO predicts centers of image  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)   bboxes.append([x,y,w,h])  confidences.append((float(confidence)))  class\_ids.append(class\_id)  # cv2.rectangle(img, (x, y), (x + w, y + h), (255,0,0), 2)   # print(len(bboxes))  indexes = cv2.dnn.NMSBoxes(bboxes, confidences, confThreshold, 0.4) #Non-maximum suppresion  # print(indexes)  # print(indexes.flatten())   font = cv2.FONT\_HERSHEY\_PLAIN  B, G, R = 150 , 0, 0   if len(indexes) > 0:  for i in indexes.flatten():  x,y,w,h = bboxes[i]  label = str(classes[class\_ids[i]])  if label == 'person' or label == 'mouse':  confidence = str(round(confidences[i],2))  cv2.rectangle(img,(x,y),(x+w,y+h),(B,G,R),2)  cv2.putText(img,label+" "+ confidence,(x,y+20),font,1,(255,255,255),2)  G = G+250   cv2.imshow('Image', img)  key = cv2.waitKey(1)  if key == 27:  break  cap.release() cv2.destroyAllWindows() |

**~ End of Question ~**