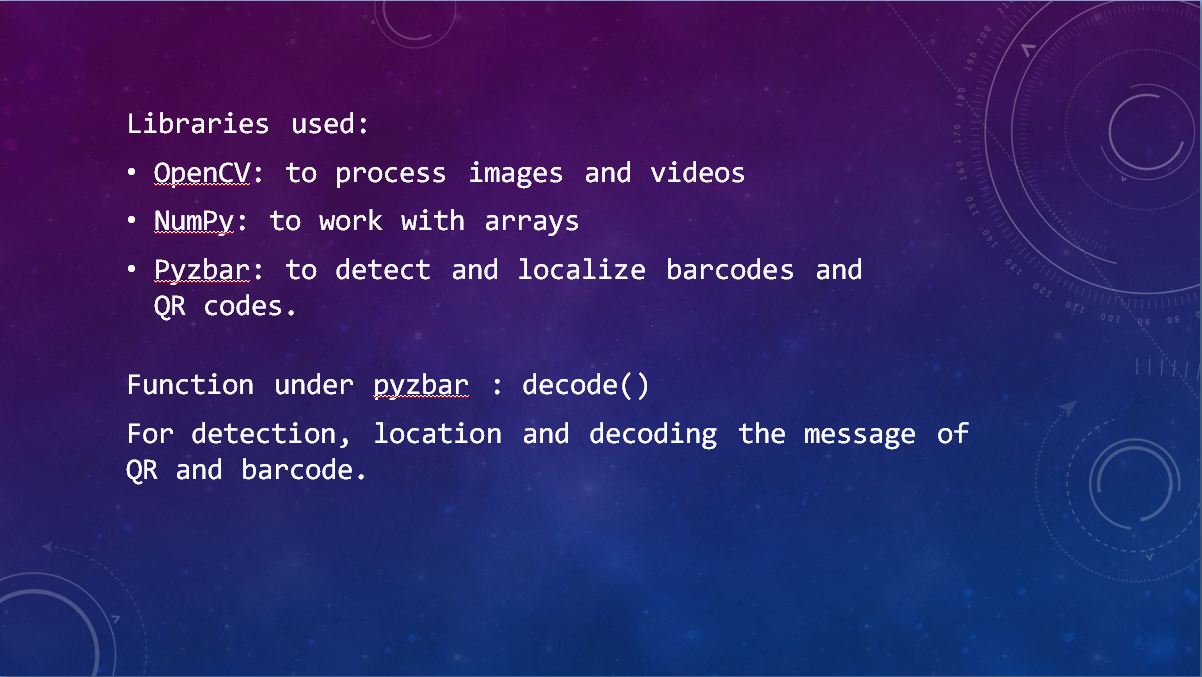
**IPV PROJECT Date: 21/06/2022**

**QR and Barcode Scanner**



****

**QR and Barcode Scanner Code:**

# QR and Bar Code Scanner

import cv2 # since we're dealing with images

import numpy as np # since we're working with arrays

from pyzbar.pyzbar import decode # to detect and localize barcodes and QR code

cap = cv2.VideoCapture(0) # for webcam, 0 is the id

cap.set(3, 640) # defines width, first value is width id

cap.set(4, 480) # defines height, first value is height id

while True:

    success, img = cap.read()

    for code in decode(img): # for detection, location and decoding the message of QR and barcode.

        myData = code.data.decode('utf-8') # from all the information, we select only the data part

        print(myData) # prints the decoded data

        pts = np.array([code.polygon],np.int32) # making a polygon instead of a rectangle, so even if the QR code is rotated it can detect properly and converting the polygon to an array

        pts = pts.reshape((-1, 1, 2)) # reshaping the array and sending it to polygon function

        cv2.polylines(img, [pts], True, (255, 0, 255), 5) # True since we want a closed figure, (255,0,255) is magenta i.e. the color we have specified, 5 is the thickness

        pts2 = code.rect # since we don't want the data to rotate

        cv2.putText(img, myData, (pts2[0], pts2[1]), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (255, 0, 255), 2) # 0.9 is the scale, 2 is the thickness

    cv2.imshow('Result', img)

    cv2.waitKey(1)

**OUTPUT:-** (a) for QR code (b) for barcode

****

**QR Code Authorization Code:**

# QR Code Authorization

import cv2 # since we're dealing with images

import numpy as np # since we're working with arrays

from pyzbar.pyzbar import decode # to detect and localize barcodes and QR code

cap = cv2.VideoCapture(0) # for webcam, 0 is the id

cap.set(3, 640) # defines width, first value is width id

cap.set(4, 480) # defines height, first value is height id

# reads all data from the text file and store it in the list

with open('myData.txt') as f:

    myDataList = f.read().splitlines() # reads all data and based on the lines it will add one item to the list, every line is a new item

print(myDataList) # prints data from our database i.e. the myData.txt file

while True:

    success, img = cap.read()

    for code in decode(img): # for detection, location and decoding the message of QR and barcode.

        myData = code.data.decode('utf-8') # from all the information, we select only the data part

        print(myData) # prints the decoded data

        if myData in myDataList: # compares the decoded data with all the items in our database i.e. the myData.txt file

            myOutput = 'Authorized'

            myColor = (0, 255, 0) # green

        else:

            myOutput = 'Un-Authorized'

            myColor = (0, 0, 255) # red

        pts = np.array([code.polygon], np.int32) # making a polygon instead of a rectangle, so even if the QR code is rotated it can detect properly and converting the polygon to an array

        pts = pts.reshape((-1, 1, 2)) # reshaping the array and sending it to polygon function

        cv2.polylines(img, [pts], True, myColor, 5) # True since we want a closed figure, changing (255,0,255) to myColor, 5 is the thickness

        pts2 = code.rect # since we don't want the data to rotate

        # myData replaced myData with myOutput, so instead of showing the actual message, it will print authorized or unauthorized

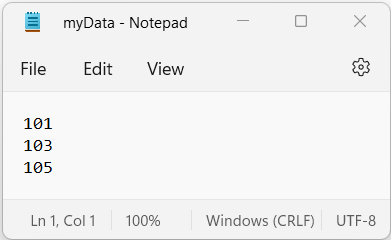
        # adding myColor variable instead of 255,0,255

        cv2.putText(img, myOutput, (pts2[0], pts2[1]), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, myColor, 2) # 0.9 is the scale, 2 is the thickness

    cv2.imshow('Result', img)

    cv2.waitKey(1)

**myData.txt file:-**

****

**OUTPUT:-**

When the decoded message is 101, it is authorized.

When the decoded message is 104, it is unauthorized.

****

**CONCLUSION:**

The mini-project on QR and Barcode Scanner using Python was implemented successfully.