Intelligent Face Recognition in Ecommerce Product Delivery System

Jacob Kurien
PG Scholar
Amal *Jyothi College Of Engineering*
Kanjirappally,Kottayam,Kerala
jacobkurien@mca.ajce.in

Ankhitha Philip
Asst. Proffessor Department of
Computer Application
Amal *Jyothi College Of Engineering*
Kanjirappally,Kottayam,Kerala
ankithaphilip@amaljyothi.ac.in

***Abstract*— This paper is written to enhance the security implementation in ecommerce product delivery system. In recent days of the new era over millions of products are being bought and sold on ecommerce website across the globe. Products from all around the world are imported and exported across countries. Key holes for fraud activities also arise due to this. Hundreds of products especially in the field of mobiles and laptops are bought from ecommerce websites during the offers and sales period and are being delivered to people with fake addresses. The aim of this paper is to ensure and enhance face authorization in ecommerce websites and applications. Face Authentication is done using Local Binary Patterns Histograms (LBPH). Which in turn increases the security of delivering products to authorized users only.**

I. INTRODUCTION

The purpose of this article is to ensure that the system recognizes the proper individual, we employ LBP (Local Binary Pattern) to recognize the face from various angles and emotions. LBP is a basic yet effective texture operator that labels pixels in an image by thresholding each pixel's neighborhood and treating the result as a binary number. It was also discovered that when LBP is paired with histograms of oriented gradient descriptors, the detection performance on specific datasets is improved. We can represent the face image with a simple data vector using LBP and histograms.

Users of the ecommerce or any substantial website should register their faces in their profile itself along with the addresses of delivery. The data of the users is stored in cloud along with credentials that match the LBPH algorithm to train the model. The dataset which is to be trained is the user registered face itself. Each and every user should register the face of the authenticated person to whom the product should be delivered to along with two other faces or accounts which can be shared to receive the delivered product in the absence of the buyer or the person who wants to receive the product .

By implementing this technique as intended into the field of ecommerce, the operators of the website can ensure and summarize the sale with the digitally saved receiver information. Thereby which increases the security in the online shopping portals.

II. SYSTEM ACTIVITY

A. *Training the model with datasets*

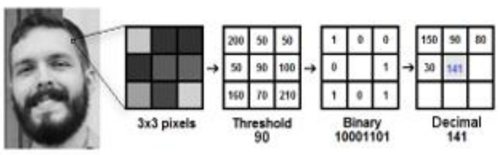
The main aim of the system is to ensure enhanced face recognition using the LBPH algorithm. So in order to accomplish this we need to train our model with the sufficient and necessary datasets. The datasets is referred to as in the sense the images of the individuals which is trained in the developed system itself. Users of the system should train or save the face of the users in their respective account. Where the algorithm identifies the face in real time and saves specific information which are relevant to recognize the person as datasets in cloud. The working of recognition is done by comparing the decimal values and histogram of the datasets and the current data. The system calculate plot the histogram of the current data and search for the same in the database. By analyzing those values, the system identifies the data.

B. *Recognizing the person*

The system compares the live person to the saved datasets in the cloud with the algorithm and its code once the algorithm identifies the person as an individual which is registered in the cloud database it returns back a confirmed Boolean value.

III. LOCAL BINARY PATTERN HISTOGRAM (LBPH)

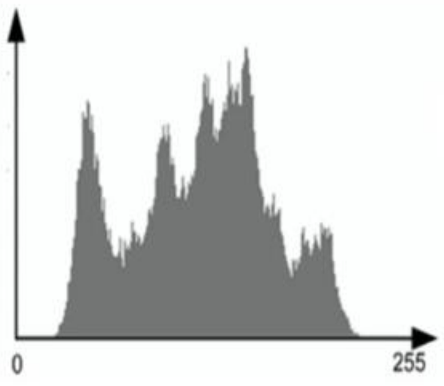
We need to train the system with different angles, emotions and perspective of the user itself from the datasets of saved images. By using the algorithm, it trains the system with 64 different images or shots of the person with the look if the face of different emotions and expressions they give to get the result. It is best advised to show different set of looks in face to ensure the best use of the system.



(Fig 1.1)

Based on the value we generate from the code a histogram is generated.

This value which is generated from the code is used to identify faces of individuals binary data, decimal codes and histogram data is used to compare with the saved result of the data from the datasets that we save.

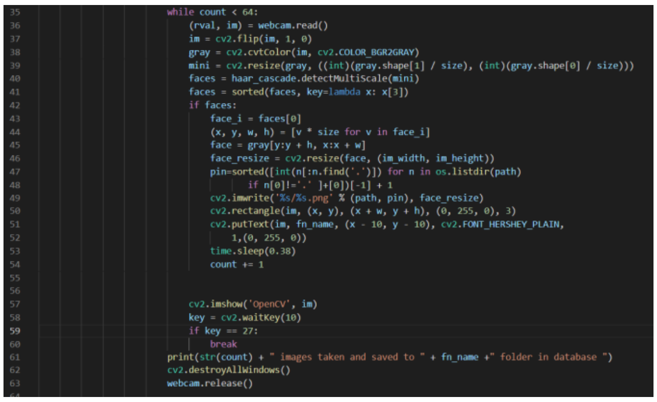


(Fig 1.2)

.

A. The Algorithm Code

1. CAPTURING THE IMAGE

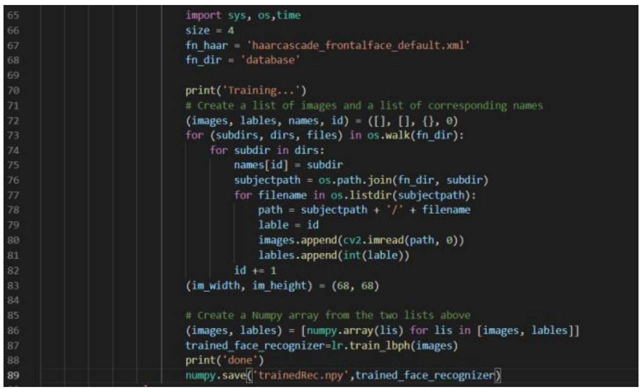


```
while count < 64:
    (rval, im) = webcam.read()
    im = cv2.flip(im, 1, 0)
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    mini = cv2.resize(gray, ((int)(gray.shape[1] / size), (int)(gray.shape[0] / size)))
    faces = haar_cascade.detectMultiScale(mini)
    faces = sorted(faces, key=lambda x: x[3])
    if faces:
        face_i = faces[0]
        (x, y, w, h) = [v * size for v in face_i]
        face = gray[y:y + h, x:x + w]
        face_resize = cv2.resize(face, (im_width, im_height))
        pin=sorted([int(n[:n.find('.')]) for n in os.listdir(path)
                if n[0]!='.' ]+[0])[-1] + 1
        cv2.imwrite('%s/%s.png' % (path, pin), face_resize)
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)
        cv2.putText(im, fn_name, (x - 10, y - 10), cv2.FONT_HERSHEY_PLAIN,
            1,(0, 255, 0))
        time.sleep(0.38)
        count += 1


    cv2.imshow('OpenCV', im)
    key = cv2.waitKey(10)
    if key == 27:
        break
print(str(count) + " images taken and saved to " + fn_name +" folder in database ")
cv2.destroyAllWindows()
webcam.release()
```

(Fig 1.3)

This is the code that is used to capture the image of the user using the webcam of the system.

2. Training Code

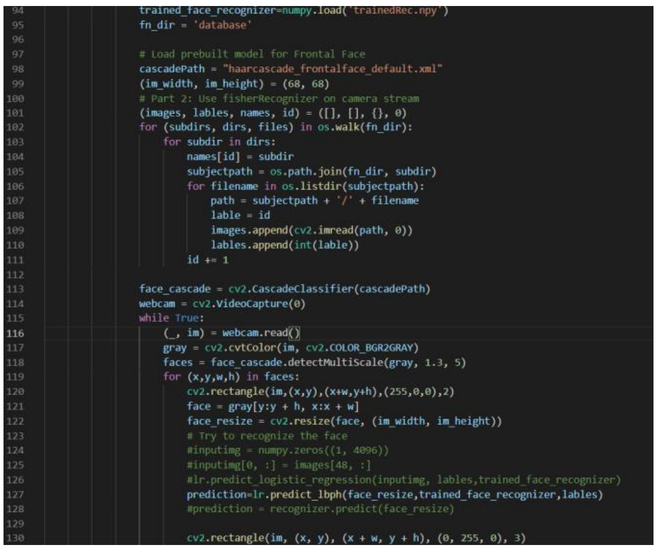


```
import sys, os,time
size = 4
fn_haar = 'haarcascade_frontalface_default.xml'
fn_dir = 'database'

print('Training...')
# Create a list of images and a list of corresponding names
(images, lables, names, id) = ([], [], {}, 0)
for (subdirs, dirs, files) in os.walk(fn_dir):
    for subdir in dirs:
        names[id] = subdir
        subjectpath = os.path.join(fn_dir, subdir)
        for filename in os.listdir(subjectpath):
            path = subjectpath + '/' + filename
            lable = id
            images.append(cv2.imread(path, 0))
            lables.append(int(lable))
        id += 1
(im_width, im_height) = (68, 68)

# Create a Numpy array from the two lists above
(images, lables) = [numpy.array(lis) for lis in [images, lables]]
trained_face_recognizer=lr.train_lbph(images)
print('done')
numpy.save('trainedRec.npy',trained_face_recognizer)
```

(Fig 1.4)

Here the images fetched from the camera is passed into the function for training.

3. Recognizing Code



```
trained_face_recognizer=numpy.load('trainedRec.npy')
fn_dir = 'database'

# Load prebuilt model for Frontal Face
cascadePath = "haarcascade_frontalface_default.xml"
(im_width, im_height) = (68, 68)
# Part 2: Use fisherRecognizer on camera stream
(images, lables, names, id) = ([], [], {}, 0)
for (subdirs, dirs, files) in os.walk(fn_dir):
    for subdir in dirs:
        names[id] = subdir
        subjectpath = os.path.join(fn_dir, subdir)
        for filename in os.listdir(subjectpath):
            path = subjectpath + '/' + filename
            lable = id
            images.append(cv2.imread(path, 0))
            lables.append(int(lable))
        id += 1

face_cascade = cv2.CascadeClassifier(cascadePath)
webcam = cv2.VideoCapture(0)
while True:
    (_, im) = webcam.read()
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(im,(x,y),(x+w,y+h),(255,0,0),2)
        face = gray[y:y + h, x:x + w]
        face_resize = cv2.resize(face, (im_width, im_height))
        # Try to recognize the face
        #inputimg = numpy.zeros((1, 4096))
        #inputimg[0, :] = images[48, :]
        #lr.predict_logistic_regression(inputimg, lables,trained_face_recognizer)
        prediction=lr.predict_lbph(face_resize,trained_face_recognizer,lables)
        #prediction = recognizer.predict(face_resize)

        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)
```

(Fig 1.5)

IV. SYSTEM WORKFLOW IN REAL TIME APPLICATIONS

A. Incorporate system with the application or website

Incorporate the intended system with the ecommerce application or website with delivery. Add in multiple fields at the delivery point of the website where the unit or the product has to be delivered to so that users can scan their images to save into our dataset. Add facility to use the camera of the device to save the dataset in cloud.

B. Train in Realtime

Collect sample images from the user and train the dataset in real time and save the necessary photos in cloud. With the authorized user of the web Application.

C. Store the data securely in cloud

The collected photos is very sensitive and it has to be stored in the cloud securely with all the privacy privileges of the user.

D. The delivery agent should be allowed to login to the site as a role which is a delivery agent. Before the delivery of the product the delivery boy has to scan the image using the camera of the device if the user is authenticated then the product can be delivered.

REFERENCES

[1] Towards Data Science, Face recognition with LBPH
https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b

[2] Analytics Vidhya, Understanding Face Recognition with LBPH
https://www.analyticsvidhya.com/blog/2021/07/understanding-face-recognition-using-lbph-algorithm/

[3] Section.io, Facial Pattern Recognition
https://www.section.io/engineering-education/understanding-facial-recognition-using-local-binary-pattern-histogram-algorithm/

[4] Devgenius, Blog, LBPH algorithm
https://blog.devgenius.io/face-recognition-based-on-lbph-algorithm-17acd65ca5f7