9 Stage 3 – Facial Recognition

The focus of Stage 3 is to build a subsystem that can detect and recognise a person's face. If recognised, the person will be granted access to the property, otherwise access will be denied[10].

9.1 Requirements

The subsystem developed in this stage aims to meet the following minimum requirements:

- The Serving Unit must…
  - Capture images of a person
  - Process/prepare images for facial detection and recognition
  - Scan images in order to detect a person's face
  - Store the relevant information required to recognise the person
  - Recognise the faces of people that are enrolled with the system
  - Not recognise the faces of people that are not enrolled with the system

- Users must be able to…
  - Add a person to the system (in order to be recognised)
  - View a list of all people currently enrolled with the system
  - Remove a person who is enrolled with the system
  - Be recognised by the system (if enrolled)

[10] Access to the property will be simulated.

9.2 Research

9.2.1 Facial Recognition Overview

Facial recognition is a term used to describe the task of authenticating the identity of an individual's face (Marcel *et al.*, 2007; Oxford English Dictionary, 2015). For more than 20 years it has attracted the attention of psychophysicists, neuroscientists, and engineers, and has drawn from various disciplines such as digital image processing, machine learning, neural networks, pattern recognition and data mining (Chellappa *et al.*, 1995; Yang *et al.*, 2002; Carrera, 2010).

Due to their continual decline in price to performance ratio, computer vision system's can now be deployed in desktop and embedded systems for the purpose of facial recognition (Yang *et al.*, 2002). Commonly used for security purposes, these systems typically capture and store specific data of a person's face known as a faceprint, which is then compared with data captured from an image (Jones, 2012).

Initially, computer-aided facial recognition systems must verify whether a face is present. Referred to as face detection, this process aims to calculate the location of faces within an image, if present. Researchers have proposed numerous methods for face detection, many of which are described in detail by Yang et al. (Yang *et al.*, 2002).

There are many different approaches to machine-based facial recognition, though typically a 3-step process is used involving detection, extraction and recognition (Carrera, 2010; Oscós *et al.*, 2014). All approaches begin with an image; this could be either a static photograph or a still taken from a video. Firstly, the image is processed to determine whether a face is present. If present, the features of the face are extracted to obtain the relevant data that identifies the face. When identifying a face, the data is passed to a classification algorithm that performs a comparison with existing facial data, for which the identity of the person is known. A prediction is made a result of the comparison, and a level of accuracy/confidence is also provided.

9.2.2 Facial Recognition Algorithms

The 3 most widely used algorithms for facial recognition are Fisherfaces, Eigenfaces and Local Binary Patterns Histogram (LBPH) (Ahmed and Amin, 2015; Wagner, 2012). A comprehensive explanation of each can found in Liu et al. (Liu *et al.*, 1991). There is much debate in the academic community regarding the pros and cons of said algorithms. However, general comparisons have been made (Lobdell, 2014; Marcel *et al.*, 2007), the most relevant of which are summarised below:

- LBPH is significantly faster than both Fisherfaces and Eighenfaces
- LBPH requires less image preparation than both Fisherfaces and Eigenfaces
- Fisherfaces and Eighenfaces require re-training when new images are added
- LBPH can be updated without re-training the dataset
- LBPH is less sensitive to changes in lighting conditions

9.2.3 Facial Recognition in Python

As alluded to in Section 7.2.4, OpenCV has a *FaceRecognizer* class that enables Python users to implement and experiment with facial recognition systems (OpenCV Dev Team, 2015a). Example 25, Example 26 and Example 27 show the fundamental operations required to perform facial detection and recognition using OpenCV.

```
# Create the cascade classifier for face detection.
cascade = cv2.CascadeClassifier("lbpcascade_frontalface.xml")

# Load an image from file in grayscale format.
img = cv2.imread("image_file.jpg", cv2.COLOR_BGR2GRAY)

# Detect faces in the image.
faces = cascade.detectMultiScale(img)

# Do something with the faces detected.
for (x, y, w, h) in faces:
   print 'Face detected at location:', 'x =', x, 'y =', y, 'width =', w, 'height =', h
```

Example 25: Basic use of OpenCV's *CascadeClassifier* class and its *detectMultiScale* function to perform face detection in OpenCV.

```
# Create the LBPH face recognizer object for face recognition.
recognizer = cv2.createLBPHFaceRecognizer()

# images = the facial images; loaded in grayscale format and converted to a NumPy array.
# labels = a NumPy array of ints that identify who each image belongs to.
images, labels = loadImagesAndLabels()

# Train the recognizer using the list of images and labels.
recognizer.train(images, labels)
# Or, if the recognizer has already been trained then...
recognizer.update(images, labels)
```

Example 26: Basic use of OpenCV's *FaceRecognizer* class and its *train/update* functions. The resulting *recognizer* object can be used to predict the identity of a person's face, as demonstrated in Example 27.

```
# Prediction needs a cropped, grayscale, NumPy image containing only the detected face.
face_img = scan_webcam_for_face_image()

# Use the recognizer to make a prediction.
predicted_label, confidence = recognizer.predict(face_img)

# Report the results of the prediction.
print 'Recognized as label', predicted_label, 'with a confidence of', confidence
```

Example 27: Using OpenCV's *FaceRecognizer* class to perform face recognition/prediciton. The *recognizer* object that was created and trained in Example 26 can be used to predict the identity of a person's face.

9.3 Design

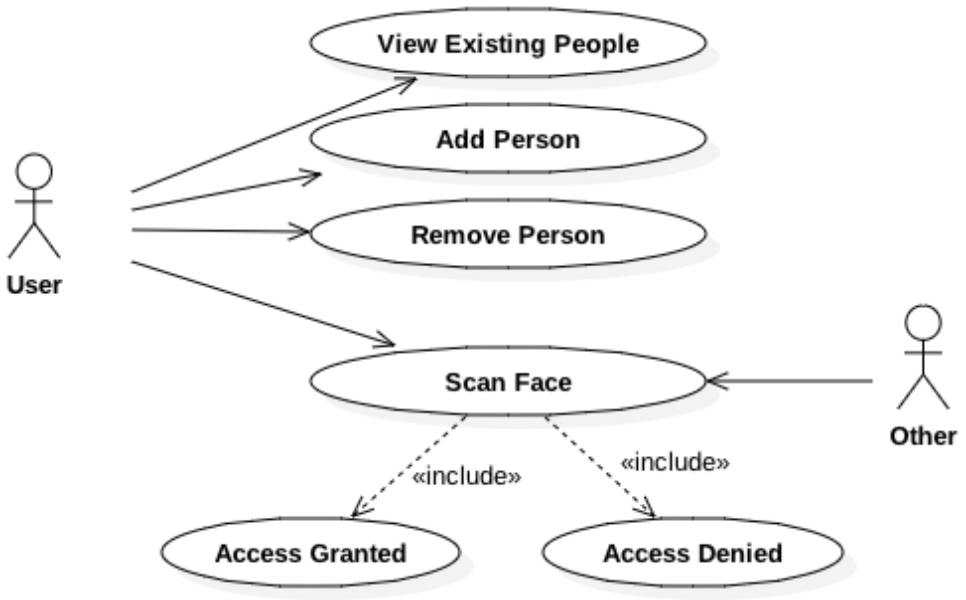

Figure 29: Use case diagram for Stage 4.

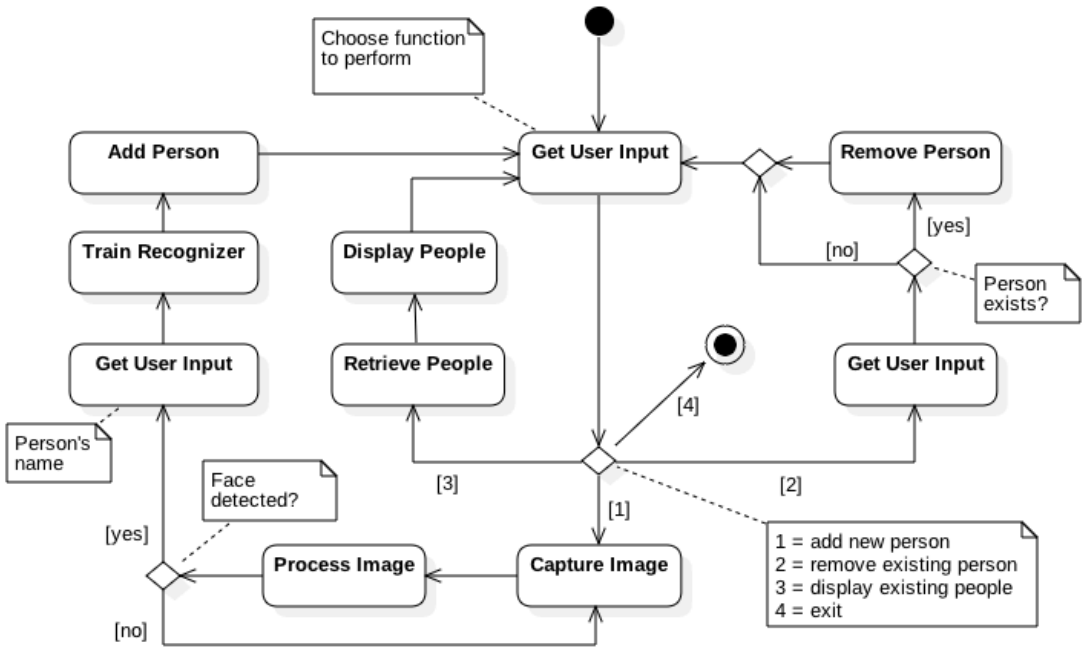

Figure 30: Activity diagram for Stage 3.

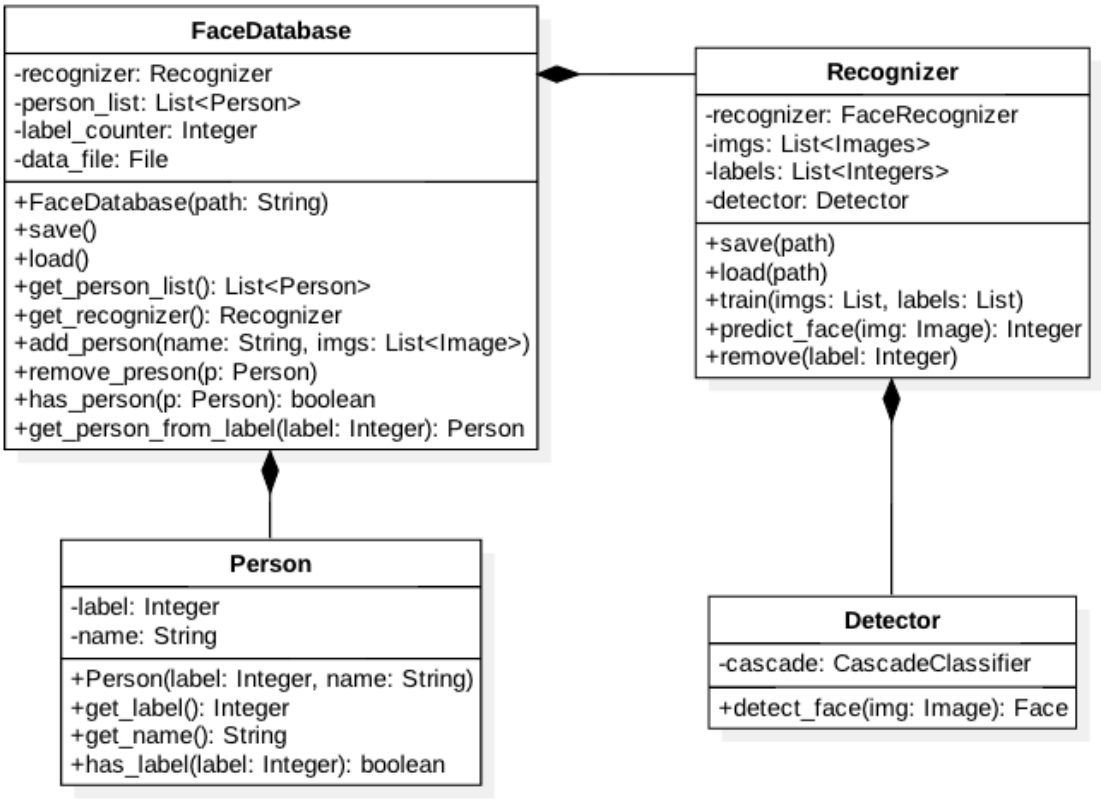

Figure 31: Class diagram for Stage 3.

9.4 Implementation

9.4.1 Creating, Saving and Loading the Face Database

Upon first launch, the application instantiates a *FaceDatabase* object that is responsible for managing all data. It stores a reference to the *Recognizer* object, a list of the people registered with the system and the name of the csv file used for storing the labels and names of people registered with the system. If the path provided in the *__init__* function already exists, it is assumed that a database is present in this location, and is loaded via the *self.load* function. Alternatively, a new database is created. This process is shown in Example 28.

When loading an existing database, the labels and names are read from the csv file. Any changes to the database are saved to the csv file during runtime. The actual data used to train/update *recognizer* object is located in a separate file with the .yml extension, which is saved/loaded using its corresponding save/*load* functions. The operations associated with loading and saving the database can be viewed in Example 29.

```
CSV_FILE = 'labels&names.csv'
…
def __init__(self, path):
        self.recognizer = Recognizer()
        self.person_list = []
        self.label_counter = 0
        self.subdir = os.path.join(ROOT, path)
        self.csv_file = os.path.join(self.subdir, CSV_FILE)

        # Check if database exists already
        if os.path.isdir(self.subdir):
                self.load()
        else:  # Create the database files
                os.makedirs(self.subdir)
                self.recognizer.save(self.subdir)
                self.create_csv()
            app.log(TAG, 'New database created')
```

Example 28: The *FaceDatabase's __init__* function, taken from the user-defined *face* module.

```
FIELD = ['label', 'name']
...
def load(self):
        try:
                self.recognizer.load(self.subdir)
                with open(self.csv_file) as c:
                        data = csv.DictReader(c)
                        for row in data:
                                label = int(row[FIELD[0]])
                                name = row[FIELD[1]]
                                p = Person(label, name)
                                self.person_list.append(p)
        except IOError as e:
                app.log(TAG, 'Error loading data -', e)
        if self.person_list:  # Set label counter
                self.label_counter = self.person_list[-1].get_label()
        app.log(TAG, 'Existing data loaded')

def save(self):
  self.recognizer.save(self.subdir)
```

Example 29: The *FaceDatabase's load* and *save* functions, taken from the user-defined *face* module.

9.4.2 Scanning the Webcam for a Person's Face

Generally, the first operation performed when using a facial recognition system is to train it to recognize a person. In order to achieve this, images containing the person's face must be obtained and processed into a format suitable for the *recognizer* object. The function *scan_webcam_for_face* (in the user-defined module *face)* performs this task, the implementation of which can be viewed in Example 30.

Put simply, the function enters a loop that performs 4 key operations. Firstly, an image is captured using the webcam device; this is accomplished using the *camera* module introduced in section 7.4.1. Secondly, the image is passed to the function *detect_face* to determine whether or not the image contains a face and, if it does, where its location is in the image (*detect_face* uses a method similar to that shown in Example 25). Thirdly, if a face is detected, the image is cropped (via the user-defined function *crop_image)* around the face area. Finally, the resulting cropped image containing the face is added to a list. When the length of this list becomes equal to 10, a number often used for training and testing face recognition systems (AT&T Laboratories, 2002; Jain *et al.*, 2008), the loop is exited and the list is returned.

```
def scan_webcam_for_face():
  TAG = 'FaceScanner'

  face_img_list = []
  while True:
    success, img = camera.get_raw_image()
    if success:  # Try to detect a face
      gray_img = image.to_grayscale(img)
      has_face, face_area = detect_face(gray_img)
      if has_face:  # Crop the image around face area
        face_img = image.crop_image(gray_img, face_area)
        face_img_list.append(face_img)
        app.log(TAG, 'Face added', len(face_img_list))
        if len(face_img_list) == 10:  # Scan complete
          break  # Exit the loop and return

  return face_img_list
```

Example 30: The user-defined function *scan_webcam_for_face*, taken from the *face* module.

9.4.3 Adding a Person to the Database

Once the list of images is returned by *scan_webcam_for_face*, the user is prompted to enter their name. This information is then passed to the *add_person* function shown in Example 31. The person is assigned a label, which is appended to the csv file along with the person's name. Finally, the list of images is passed to the *recognizer* object for training, before returning the newly created *Person* object.

```
def add_person(self, name, imgs):
        self.label_counter += 1
        label = self.label_counter
        p = Person(label, name)  # Create the person
        # Add the person to the database
        self.person_list.append(p)
        with open(self.csv_file, 'a') as c:
                w = csv.DictWriter(c, fieldnames=FIELD)
                w.writerow({'label': p.get_label(), 'name': p.get_name()})
        app.log(TAG, 'Added -', p.get_name())

        # Lastly, use the images to train/update the recognizer
        labels = [p.get_label()] * len(imgs)
        self.recognizer.train(imgs, labels)

      return p
```

Example 31: The user-defined function *add_person*, taken from the *FaceDatabase* class of the *face* module.

9.4.4 Listing Existing People in the Database

A list of existing people in the database can be obtained by calling the *get_person_list* function of the *FaceDatabase* object. Example 32 demonstrates the information that can be accessed via the list, such as the amount of people in the database and each person's label and name.

```
def list_all_persons():
    app.log(TAG, 'Existing people:', len(data.get_person_list()))
    for p in data.get_person_list():
        app.log(TAG, '>', p.get_label(), '-', p.get_name())
```

Example 32: The *get_person_list* function can be used to access the names and labels of people in the database.

9.4.5 Removing an Existing Person from the Database

As demonstrated in Example 33, a person can be removed from the database by calling the *remove_person* function with the *person* object to be removed. Since, the *recognizer* is not directly capable of removing images and labels, a helper function, called *remove_label*, was written to simulate this operation. In short, the *recognizer* object keeps lists of all images and labels it is trained/updated with. When a call is made to remove a label, it and all associated images are deleted from the lists, which are then passed to the *recognizer* for retraining. Similarly, the csv file used to store the labels and names of people is not directly capable of deleting specific entries. Hence, a new file is created to reflect the changes made.

```
def remove_person(self, person):
        self.recognizer.remove_label(person.get_label())
        self.person_list.remove(person)

        # Can't delete specific entries. Create new file and override the old one
        self.create_csv()
        with open(self.csv_file, 'a') as c:
                for p in self.person_list:
                        w = csv.DictWriter(c, fieldnames=FIELD)
                        w.writerow({'label': p.get_label(), 'name': p.get_name()})
        app.log(TAG, 'Removed -', person.get_name())
```

Example 33: The user-defined function *remove_person*, taken from the *FaceDatabase* class of the *face* module.

9.4.6 Recognizing/Predicting People

A similar method to that shown in Example 27 is used to predict the identity of a person. The LBPH recognition algorithm was selected for the reasons specified in Section 9.2.2. Firstly, the *scan_webcam_for_face* function is employed to obtain images of the person's face. These are then passed to the function shown in Example 34, which iterates the list of images making a prediction for each. The most commonly predicted label is returned, unless it is equal to -1, which indicates that the face was not recognised.

```
def predict_face_from_list(self, imgs):
        pred_labels = []
        for img in imgs:
                label, conf = self.predict_face(img)
                app.log(self.TAG, 'Label: {} Conf: {}'.format(label, conf))
                pred_labels.append(label)
        predicted_label = Counter(pred_labels).most_common(1)[0][0]
        if predicted_label == -1:  # Face wasn't recognised
                return None
        return predicted_label
```

Example 34: The user-defined function *predict_face_from_list*, taken from the *Recognizer* class of the *face* module.

9.4.7 Practical Demonstration

```
    FaceRecDemo:  Loading data.... ................................ 03:45:53PM on 27-Sep-2015
   FaceDatabase:  New database created. ........................... 03:45:53PM on 27-Sep-2015

************************************************************
************  Face Recognition Demonstration  ************
************************************************************

Usage: a  -  add new person to the dataset
       l  -  list all existing people in the dataset
       r  -  remove an existing person from the dataset
       p  -  predict a face by scanning the webcam
       e  -  exit.

->
```

Figure 32: The application upon first launch. Note that an existing database is not found, thus a new database is created.

```
-> a
   FaceDetector:  Face detected. ................................. 03:46:27PM on 27-Sep-2015
    FaceScanner:  Face added 1. .................................. 03:46:27PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:27PM on 27-Sep-2015
    FaceScanner:  Face added 2. .................................. 03:46:27PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:27PM on 27-Sep-2015
    FaceScanner:  Face added 3. .................................. 03:46:27PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:27PM on 27-Sep-2015
    FaceScanner:  Face added 4. .................................. 03:46:27PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:27PM on 27-Sep-2015
    FaceScanner:  Face added 5. .................................. 03:46:27PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:28PM on 27-Sep-2015
    FaceScanner:  Face added 6. .................................. 03:46:28PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:28PM on 27-Sep-2015
    FaceScanner:  Face added 7. .................................. 03:46:28PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:28PM on 27-Sep-2015
    FaceScanner:  Face added 8. .................................. 03:46:28PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:28PM on 27-Sep-2015
    FaceScanner:  Face added 9. .................................. 03:46:28PM on 27-Sep-2015
   FaceDetector:  Face detected. ................................. 03:46:28PM on 27-Sep-2015
    FaceScanner:  Face added 10. ................................. 03:46:28PM on 27-Sep-2015

Please enter your name: Matthew
   FaceDatabase:  Added - Matthew. ............................... 03:46:38PM on 27-Sep-2015
     Recognizer:  Update completed. ............................... 03:46:38PM on 27-Sep-2015
```

Figure 33: The user has selected the option to add a new person to the database. The facial images are obtained from the webcam, a name is entered and both the database and recognizer update accordingly.

```
-> l
    FaceRecDemo:  Existing people: 1. ............................. 03:47:06PM on 27-Sep-2015
    FaceRecDemo:  > 1 - Matthew. .................................. 03:47:06PM on 27-Sep-2015
->
```

Figure 34: The user has selected the option to list all existing people in the database.

```
-> p
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 1. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 2. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 3. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 4. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 5. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 6. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 7. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 8. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 9. ..................................... 03:47:43PM on 27-Sep-2015
  FaceDetector:  Face detected. .................................... 03:47:43PM on 27-Sep-2015
   FaceScanner:  Face added 10. .................................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 14.2723351323. ..................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 12.7001858912. ..................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 12.622996678. ...................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 12.8787796918. ..................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 13.3494515005. ..................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 13.2936960126. ..................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 13.5054865675. ..................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 13.1345461011. ..................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 13.4138011475. ..................... 03:47:43PM on 27-Sep-2015
    Recognizer:  Label: 1 Conf: 14.1536954725. ..................... 03:47:43PM on 27-Sep-2015
   FaceRecDemo:  Access granted. Recognised as Matthew. ............ 03:47:43PM on 27-Sep-2015
->
```

Figure 35: The user has selected the option to predict the face of the person in the webcam. In this instance, the person is the user added in Figure 33, hence access is granted.

```
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
    Recognizer:  Label: -1 Conf: 1.79769313486e+308. ............... 03:49:44PM on 27-Sep-2015
   FaceRecDemo:  Access denied. .................................... 03:49:44PM on 27-Sep-2015
->
```

Figure 36: The user has selected the option to predict the face of the person in the webcam. In this instance, the person is NOT the user added in Figure 33, hence access is denied.

```
-> e
   FaceRecDemo:  Data saved. ....................................... 03:50:48PM on 27-Sep-2015
   FaceRecDemo:  Goodbye. .......................................... 03:50:48PM on 27-Sep-2015
```

Figure 37: The user has selected the option to exit the application; existing data is saved.

```
   FaceRecDemo:  Loading data.... .................................. 03:51:21PM on 27-Sep-2015
  FaceDatabase:  Existing data loaded. ............................. 03:51:21PM on 27-Sep-2015

************************************************************
************  Face Recognition Demonstration  ************
************************************************************
```

Figure 38: The application upon first launch. This time, the database saved in Figure 37 is loaded.

Figure 39: The user has selected the option to remove an existing person from the database. The user enters the label to be removed and both the database and recognizer update accordingly. The removal is confirmed by the following command to list all existing people.

13 Bibliography

Ahmed, M.T., Amin, S.H.M. (2015) Comparison of Face Recognition Algorithms for Human - Robot Interactions. *Jurnal Teknologi*. **72**(2), 1–6.

Andrews, J.G., Buzzi, S., Choi, W., Hanly, S. V., Lozano, A., Soong, A.C.K., Zhang, J.C. (2014) What will 5G be? *IEEE Journal on Selected Areas in Communications*. **32**(6), 1065–1082.

Android (2015a) NetworkOnMainThreadException. *Android 6.0 r1*. [online]. Available from: http://developer.android.com/reference/android/os/NetworkOnMainThreadException.html [Accessed September 5, 2015].

Android (2012) platform_development/samples/SipDemo. *Github*. [online]. Available from: https://github.com/android/platform_development/tree/master/samples/SipDemo [Accessed September 23, 2015].

Android (2015b) Session Initiation Protocol. *Android API Guide*. [online]. Available from: http://developer.android.com/guide/topics/connectivity/sip.html [Accessed September 23, 2015].

Ashraf, Q.M., Habaebi, M.H. (2015) Autonomic schemes for threat mitigation in Internet of Things. *Journal of Network and Computer Applications*. **49**, 112–127.

AT&T (2005) *Making the Case for VoIP*.

AT&T Laboratories (2002) The Database of Faces. [online]. Available from: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html [Accessed September 24, 2015].

Bacon, J. (2006) Getting started with GStreamer with Python. [online]. Available from: http://www.jonobacon.org/2006/08/28/getting-started-with-gstreamer-with-python/ [Accessed September 19, 2015].

Bambino (2015) PyAudio playback slow on linux. [online]. Available from: http://stackoverflow.com/questions/28649207/pyaudio-playback-slow-on-linux [Accessed September 25, 2015].

Bird Home Automation (2015) DoorBird. [online]. Available from: https://www.doorbird.com [Accessed May 17, 2015].

Bot Home Automation (2015) ring. [online]. Available from: https://ring.com [Accessed May 17, 2015].

Bradley, J., Barbier, J., Handler, D. (2013) *Embracing the Internet of Everything To Capture Your Share of $14.4 Trillion*. San Jose, CA.

Brooke, J. (1996) *SUS - A quick and dirty usability scale*. Reading.

Carrera, P. De (2010) Face Recognition Algorithms. *Master's thesis in Computer Science, Universidad*

Cavanaugh, E. (2006) *Web services: Benefits, challenges, and a unique, visual development solution*.

Chang, S., Kirsch, A., Lyons, M. (2007) *Energy and Storage Reduction in Data Intensive Wireless Sensor Network Applications*. Cambridge, Massachusetts.

Chellappa, R., Wilson, C.L., Sirohey, S. (1995) Human and machine recognition of faces: a survey. *Proceedings of the IEEE*. **83**(5), 705–740.

Chong, G., Zhihao, L., Yifeng, Y. (2011) The research and implement of smart home system based on Internet of Things. *2011 International Conference on Electronics, Communications and Control (ICECC)*, 2944–2947.

Chou, S., Chiu, T., Yu, Y., Pang, A. (2014) Mobile Small Cell Deployment for Next Generation Cellular Networks. , 4852–4857.

Chui (2015) Chui. [online]. Available from: https://www.getchui.com [Accessed May 17, 2015].

Covey, C., Friesz, A. (2003) *Remote Intercom with Internet Video*. The University of Nebraska-Lincoln, Omaha Campus.

Croston, B. (2015) RPi.GPIO 0.5.11. [online]. Available from: https://pypi.python.org/pypi/RPi.GPIO [Accessed September 25, 2015].

Ernest, J. (2015) Audio problem with snd_bcm2835 + PyAudio (sound stuttering) #994. [online]. Available from: https://github.com/raspberrypi/linux/issues/994 [Accessed September 25, 2015].

Garden, D. (2014) PebblyPi Smart Doorbell. [online]. Available from: http://www.muacksandglomps.com/blog/2014/04/29/pebblypi-smart-doorbell/ [Accessed September 2, 2015].

Gartner (2014) *Hype Cycle for the Internet of Things, 2014*. Stamford.

GStreamer Team (2015) GStreamer Python Bindings Supplement. [online]. Available from: http://gstreamer.freedesktop.org/modules/gst-python.html [Accessed September 19, 2015].

Haller, M. (2015) PyFFmpeg - Python FFmpeg wrapper. [online]. Available from: https://github.com/mhaller/pyffmpeg [Accessed September 19, 2015].

Harold, E.R. (2004) *Java Network Programming*. 3rd ed. Sebastopol , CA: O'Reilly.

Harper, R. (2003) *Inside the Smart Home*. London: Springer.

Hassan, M., Nayandoro, A., Atiquzzaman, M. (2000) Internet telephony: services, technical challenges, and products. *IEEE Communications Magazine*. **38**(4), 96–103.

Hendryx-Parker, C. (2013) Why We Choose Python. [online]. Available from: http://www.sixfeetup.com/blog/why-we-choose-python [Accessed September 15, 2015].

i-Bell (2015) i-Bell. [online]. Available from: http://www.i-bell.co.uk [Accessed May 17, 2015].

International Data Corporation (2014) Smartphone OS Market Share, Q4 2014. [online]. Available from: http://www.idc.com/prodserv/smartphone-os-market-share.jsp [Accessed April 27, 2015].

Itseez (2015a) About OpenCV. [online]. Available from: http://opencv.org/about.html [Accessed September 21, 2015].

Itseez (2015b) OpenCV Homepage. [online]. Available from: http://opencv.org [Accessed September 20, 2015].

Jain, A., Flynn, P., Ross, A.A. (2008) *Handbook of Biometrics*. New York: Springer Science & Business Media.

Jarvis, M. (2015) *Towards the Development of a Low-Cost Smart Entry System*. The University of Northampton.

Jones, M. (2012) Facial recognition. [online]. Available from: http://whatis.techtarget.com/definition/facial-recognition [Accessed September 20, 2015].

Jones, T. (2005) Sockets programming in Python. , 1–25.

Kellmereit, D., Obodovski, D. (2013) *The Silent Intelligence: The Internet of Things*. San Francisco: DND Ventures.

Leonard, M. (2014) How to Choose the Right Platform: Raspberry Pi or BeagleBone Black? [online]. Available from: http://makezine.com/2014/02/25/how-to-choose-the-right-platform-raspberry-pi-or-beaglebone-black/ [Accessed September 25, 2015].

Liao, W.L.W. (1999) Mobile Internet telephony protocol: an application layer protocol for mobile Internet telephony services. *1999 IEEE International Conference on Communications (Cat. No. 99CH36311)*. **1**, 339–343.

Liu, Y., Ng, W.-S., Liu, C. (1991) *A Comparison of Different Face Recognition Algorithms*. Taiwan.

Lobdell, S. (2014) Facial Recognition in Python. [online]. Available from: http://scottlobdell.me/2014/01/facial-recognition-in-python/ [Accessed September 20, 2015].

Lynn, H. (2015) RASPBERRY PI INTERNET DOORBELL. [online]. Available from: https://www.raspberrypi.org/blog/internet-doorbell/ [Accessed September 2, 2015].

Maculan, I. (2015) Simple Python Motion Jpeg. [online]. Available from: https://gist.github.com/n3wtron/4624820 [Accessed September 21, 2015].

Marcel, S., Rodriguez, Y., Heusch, G. (2007) On the Recent Use of Local Binary Patterns for Face Authentication. *International Journal on Image and Video Processing Special Issue on Facial Image Processing*, 1–9.

McEwen, A., Cassimally, H. (2014) *Designing the Internet of Things*. Chichester: John Wiley and Sons.

McMillan, G. (2015) Socket Programming HOWTO. [online]. Available from: https://docs.python.org/2/howto/sockets.html [Accessed September 3, 2015].

Meunier, F., Wood, A., Weiss, K., Huberty, K., Flannery, S., Moore, J., Hettenbach, C., Lu, B. (2014) *The 'Internet of Things' is Now: Connecting the Real Economy*. New York.

Michigan, T.L. of (1999) Chapter 8: Overview of Streaming Audio and Video. In *Building a Community Information Network: A Guidebook Chapter*. Michigan.

Nematbakhsh, S. (2005) *External Interfaces and Software Tools for Electronic Blocks*. University of California Riverside.

Nest Labs (2015) nest.com. [online]. Available from: https://nest.com/uk/ [Accessed May 16, 2015].

OpenCV Dev Team (2015a) Facial Recognition with OpenCV. *OpenCV 2.4.11.0 documentation*. [online]. Available from: http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html#face-recognition-with-opencv [Accessed September 20, 2015].

OpenCV Dev Team (2015b) Reading and Writing Images and Video. *OpenCV API Reference*. [online]. Available from: http://docs.opencv.org/modules/highgui/doc/reading_and_writing_images_and_video.html [Accessed September 21, 2015].

Oracle (2014a) Class BufferedReader. *Java™ Platform, Standard Edition 7 API Specification*. [online]. Available from: http://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html [Accessed September 15, 2015].

Oracle (2014b) Class BufferedWriter. *Java™ Platform, Standard Edition 7 API Specification*. [online]. Available from: http://docs.oracle.com/javase/7/docs/api/java/io/BufferedWriter.html [Accessed September 15, 2015].

Ortiz, S. (2004) Internet telephony jumps off the wires. *Computer*. **37**(12), 16–19.

Oscós, G.C., Khoshgoftaar, T.M., Wald, R. (2014) Rotation Invariant Face Recognition Survey. , 835–840.

Oxford English Dictionary (2015) Facial recognition. *Oxford English Dictionary*. [online]. Available from: http://www.oxforddictionaries.com/definition/english/facial-recognition [Accessed September 20, 2015].

Park, Y.T., Sthapit, P., Pyun, J.Y. (2009) Smart digital door lock for the home automation. In *IEEE Region 10 Annual International Conference, Proceedings/TENCON*. Singapore, pp. 1–6.

Pham, H. (2014) PyAudio. [online]. Available from: https://people.csail.mit.edu/hubert/pyaudio/ [Accessed September 23, 2015].

Piyare, R. (2013) Internet of Things : Ubiquitous Home Control and Monitoring System using Android based Smart Phone. *international Journal of Internet of Things*. **2**(1), 5–11.

Prijono, B. (2008) Your Python SIP 'Hello World!' Application. *PJSUA Python Module Tutorial*. [online]. Available from: https://trac.pjsip.org/repos/wiki/Python_SIP/Hello_World [Accessed September 23, 2015].

Python (2015) 5.9. File Objects. *Python 2.7.10 documentation*. [online]. Available from: https://docs.python.org/2/library/stdtypes.html#file-objects [Accessed September 15, 2015].

Python Software Foundation (2015a) 20.17. SocketServer — A framework for network servers. *Python 2.7.10 documentation*. [online]. Available from: https://docs.python.org/2/library/socketserver.html [Accessed September 22, 2015].

Python Software Foundation (2015b) Applications for Python. [online]. Available from: https://www.python.org/about/apps/ [Accessed September 26, 2015].

Raspberry Pi Forum (2015) Pyaudio outputs slow/crackling/garbled audio. [online]. Available from: https://www.raspberrypi.org/forums/viewtopic.php?f=32&t=115308 [Accessed September 25, 2015].

Robertson, G., Craw, I. (1993) Testing Face Recognition Systems. *Procedings of the British Machine Vision Conference 1993*, 3.1–3.10.

Rose, D. (2014) *Enchanted Objects: Design, Human Desire, and the Internet of Things*. New York: Scribner.

Sauro, J. (2011) SUStisfied? Little-Known System Usability Scale Facts. *User Experience Magazine*. [online]. Available from: http://uxpamagazine.org/sustified/ [Accessed September 24, 2015].

Sicari, S., Rizzardi, a., Grieco, L. a., Coen-Porisini, a. (2015) Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*. **76**, 146–164.

Simpson, W. (2008) *Voice Over IP*. 2nd ed. Oxford: Focal Press.

SkyBell (2015) SkyBell™. [online]. Available from: http://www.skybell.com [Accessed May 17, 2015].

Soliman, M., Abiodun, T., Hamouda, T., Zhou, J., Lung, C.-H. (2013) Smart Home: Integrating Internet of Things with Web Services and Cloud Computing. *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, 317–320.

Song, Y., Han, B., Zhang, X., Yang, D. (2012) Modeling and simulation of smart home scenarios based on Internet of Things. , 596–600.

Storck, C. (2013) Notifying Doorbell with PushingBox. [online]. Available from: http://makezine.com/projects/notifying-doorbell-with-pushingbox/ [Accessed September 2, 2015].

Teluu Ltd. (2015) About PJSIP. [online]. Available from: http://www.pjsip.org [Accessed September 23, 2015].

Thompson, J. (2015) Installing FFMPEG for Raspberry Pi. [online]. Available from: http://www.jeffreythompson.org/blog/2014/11/13/installing-ffmpeg-for-raspberry-pi/ [Accessed September 19, 2015].

Tschalär, R. (2006) Full-Duplex Channel over HTTP. [online]. Available from: http://www.innovation.ch/java/HTTPClient/fullduplex.html [Accessed September 24, 2015].

Unique Media TV (2012) Streaming vs. Progressive Download. [online]. Available from: https://www.unique-media.tv/support/28/Introduction/Streaming_vs_Progressive_Download [Accessed September 17, 2015].

Vermesan, O., Friess, P. (2013) *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. Aalborg: River Publishers.

Wagner, P. (2012) *Face Recognition with Python*.

Walport, M. (2014) *The Internet of Things: making the most of the Second Digital Revolution*. London.

Willetts, D. (2013) *Eight great technologies*. London.

Wilstrup, C., Immisch, L. (2009) What is ALSA. [online]. Available from: http://larsimmisch.github.io/pyalsaaudio/pyalsaaudio.html#what-is-alsa [Accessed September 25, 2015].

Wingerd, M. (2014) Using GStreamer with Python. [online]. Available from: https://markwingerd.wordpress.com/2014/11/19/using-gstreamer-with-python/ [Accessed September 19, 2015].

Yang, M.H., Kriegman, D.J., Ahuja, N. (2002) Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **24**(1), 34–58.