**Smart Surveillance System using Raspberry Pi and Face Recognition.**

**Description-** A small project which does face detection using OpenCV library on RaspberryPi.

**Hardware used in project :**

1. Raspberry pi
2. PiCam

**Software used in project :**

1. OpenCV Library
2. Python3
3. TextEditor

**Project Structure**

1. Face Detection and gathering
2. Train the system
3. Face Recognition

**Procedure**

**Step 1 : Installing OpenCV library and other libraries if not installed**

* Install libraries:  
  Numpy: pip install numpy  
  cv2: pip install opencv-python  
  skimage: pip install scikit-image  
  imutils: pip install imutils
* Run the below command each time you open up a new terminal to ensure your system variables have been set up correctly and ensure you are in the directory where we want to run our code.
* source ~/.profile
* workon cv
* If we see the text (cv) preceding your prompt, then we are in the cv virtual environment.

**Step 2: Face Detection**

* The most common way to detect a face (or any objects), is using the "Haar Cascade classifier".
* Copy the file ‘haarcascade\_frontalface\_default.xml’ from opencv-3.3.0/data/haarcascades.
* Download the file: [FaceDetection.py](https://github.com/kunalyelne/Face-Recognition-using-Raspberry-Pi/blob/master/FaceDetection.py)
* Now, run the above python Script on your python environment, using the Raspberry Pi Terminal: python FaceDetection.py
* After executing the above code we will be able to see a window popping which includes your face.

**Step 3**: Data Gathering

* Create a directory where we develop our project : mkdir FaceRecognition
* In this directory, besides the 3 python scripts that we will create for our project, we must have saved on it the Facial Classifier. We can copy [haarcascade\_frontalface\_default.xml](https://github.com/kunalyelne/Face-Recognition-using-Raspberry-Pi/blob/master/haarcascade_frontalface_default.xml) it to FaceRecognition directory.
* Download the dataset folder from git.
* If want to create new dataset create a subdirectory named dataset under FaceRecognition directory.
* Download [01\_face\_dataset.py](https://github.com/kunalyelne/Face-Recognition-using-Raspberry-Pi/blob/master/01_face_dataset.py) for creating new dataset.
* The code is very similar to the code that we saw for face detection. What we added, was an "input command" to capture a user id, that should be an integer number (1, 2, 3, etc)  
  face\_id = input('\n enter user id end press ==> ').

**Step 4:** Trainer

* Install PIL library on our Raspberry Pi : pip install pillow.
* Create trainer directory in face Recognition directory : mkdir trainer.
* Download [02\_face\_training.py](https://github.com/kunalyelne/Face-Recognition-using-Raspberry-Pi/blob/master/02_face_training.py)
* In [02\_face\_training.py](https://github.com/kunalyelne/Face-Recognition-using-Raspberry-Pi/blob/master/02_face_training.py), we have to mention complete path of dataset directory.
* Run the above python script and As a result, a file named "trainer.yml" will be saved in the trainer directory that was previously created by us.

**Step 5:** **Recognizer**

Here, we will capture a fresh face on our camera and if this person had his face captured and trained before, our recognizer will make a "prediction" returning its id and an index.

Download [03\_face\_recognition.py](https://github.com/kunalyelne/Face-Recognition-using-Raspberry-Pi/blob/master/03_face_recognition.py)

We are including here a new array, so we will display "names", instead of numbered ids for example

Names = [‘None’, ‘Pratik’,’Piyush’,’Rishab’] where index 1 denotes user id 1.