Required OpenCV Libraries (image processing, face rec, raspicam....):

```
#include <raspicam/raspicam_cv.h>
#include <opencv2/core.hpp>
#include <opencv2/face/facerec.hpp>
#include "opencv2/highgui.hpp"
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/objdetect/objdetect.hpp>
#include <iostream>
#include <fstream>
#include <sstream>

using namespace cv;
using namespace cv::face;
using namespace std;
```

Face detection Function:

```
void detectFace(cv::Mat &frame,
        cv::CascadeClassifier &face_cascade,
        vector<cv::Rect> &faces,
        cv::Ptr<cv::face::BasicFaceRecognizer> &model,
        int &pos_x,
        int &pos_y,
        string &text)

{
  cv::Mat grayscale;

  // Convert frame to grayscale, normalize the brightness, and increase the contrast
  cv::cvtColor(frame, grayscale, cv::COLOR_BGR2GRAY);
  cv::equalizeHist(grayscale, grayscale);

  // Detect faces
  face_cascade.detectMultiScale(grayscale, faces, 2, 6, 0|CV_HAAR_SCALE_IMAGE, cv::Size(50, 50));

  for (size_t i = 0; i < faces.size(); i++)
{

cv::Point center( faces[i].x + faces[i].width*0.5, faces[i].y + faces[i].height*0.5 );
cv::ellipse( frame, center, cv::Size( faces[i].width*0.5, faces[i].height*0.5), 0, 0, 360, cv::Scalar( 255, 255, 255 ), 2, 8, 0 );
```

Reading the CSV file for face recognizer modeling and training:

```
static void read_csv(const string& filename, vector<Mat>& images, vector<int>& labels, char separator = ';') {
    std::ifstream file(filename.c_str(), ifstream::in);
    if (!file) {
        string error_message = "No valid input file was given, please check the given filename.";
        CV_Error(Error::StsBadArg, error_message);
    }
    string line, path, classlabel;
    while (getline(file, line)) {
        stringstream liness(line);
        getline(liness, path, separator);
        getline(liness, classlabel);
        if(!path.empty() && !classlabel.empty()) {
            images.push_back(imread(path, 0));
            labels.push_back(atoi(classlabel.c_str()));
        }
    }
}
```

Define the RPI Camera class and Face Recognizer class and face cascade classifier for face detection:

```
raspicam::RaspiCam_Cv Camera; // Camera Object
cv::Mat frame; // Image
cv::CascadeClassifier face_cascade; // Face Cascade Classifier
//cv::Ptr<cv::face::FaceRecognizer> model = cv::face::createLBPHFaceRecognizer();
  Ptr<BasicFaceRecognizer> model = createFisherFaceRecognizer();
```

Openning RPI Camera module and video capturing:

```
// Set camera params
Camera.set(CV_CAP_PROP_FORMAT, CV_8UC3); // For color

// Open camera
cout << "Opening camera..." << endl;
if (!Camera.open()) {
  cerr << "Error opening camera!" << endl;
  return -1;
}

// Start capturing
cv::namedWindow("Display Window", cv::WINDOW_AUTOSIZE);

for (;;i++) {
  Camera.grab();
  Camera.retrieve(frame);

  // Don't call detectFace on every iteration, it's too expensive
  if (i % 2 == 0) {
    detectFace(frame,
       face_cascade,
       faces,
       model,
       pos_x,
       pos_y,
       text);
  } else {

for (size_t i = 0; i < faces.size(); i++)
{

cv::Point center( faces[i].x + faces[i].width*0.5, faces[i].y + faces[i].height*0.5 );

cv::ellipse( frame, center, cv::Size( faces[i].width*0.5, faces[i].height*0.5), 0, 0, 360, cv::Scalar( 255, 255, 255 ), 2, 8, 0 );

 cv::putText( frame, text, cv::Point(pos_x, pos_y), cv::FONT_HERSHEY_PLAIN, 1.0, CV_RGB(255,255,255), 1);
}
```

Face Recognition (modeling and prediction):

modeling face recognizer and loading cascade classifier:

```
  // Load face cascade
  cout << "Loading face cascade.." << endl;
  if (!face_cascade.load(classifier_file)) {
    cerr << "Error loading face cascade!" << endl;
    return -1;
  }

  // Load FaceRecognizer model
  //cout << "Loading face recognition model.." << endl; model->load(face_model);
   model->train(images, labels);
```

prerpcessing and then Prediction and recognition:

```
// Recognize face
// Get and resize detected face
 cv::Mat face_i = cv::Mat(frame, faces[i]);

// Grab face from frame inside Rect
cv::cvtColor(face_i, face_i, cv::COLOR_BGR2GRAY);
cv::resize(face_i, face_i, cv::Size(100, 100), 0, 0, CV_INTER_NN);
double predicted_confidence = 0.0;
int prediction = -1;
model->predict(face_i,prediction,predicted_confidence);

//prediction = model->predict(face_i);

    cout << "Prediction " << prediction << " Confidence " << predicted_confidence << endl;
    if (prediction != -1) {
      text = "My gosh, you look familiar...";
      pos_x = max(faces[i].tl().x - 10, 0);
      pos_y = max(faces[i].tl().y - 10, 0);
      cv::putText( frame, text, cv::Point(pos_x, pos_y), cv::FONT_HERSHEY_PLAIN, 1.0, CV_RGB(255,255,255), 1);
    } else {
      text = "";
    }
  }
}
```

Experiment Results:

Two experiments were performed :
Variation in lighting (40 images of 3 people)
Variation in lighting and facial expression (20 images of 3 people)

Experiment 1: Lighting variations
One light source.
Using extrapolation process

1. The Yale database is available for download from http://cvc.yale.edu.

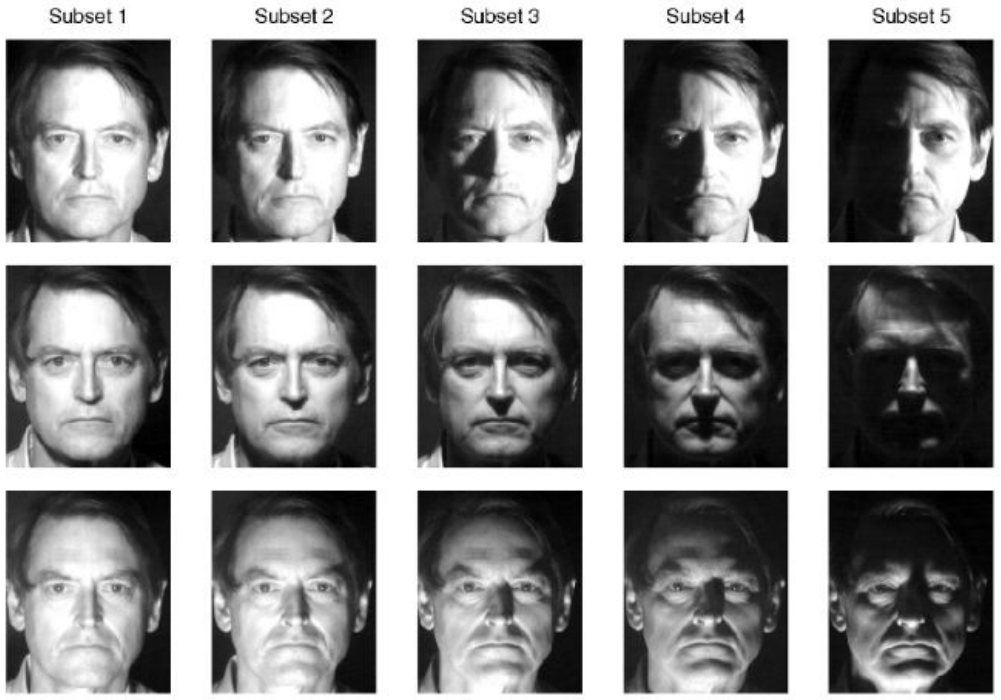


Fig. 4. Example images from each subset of the Harvard Database used to test the four algorithms.

Trained on subset 1 and tested on subsets 1, 2, 3

Error Ratio (%)		
Subset 3	**Subset 2**	**Subset 1**
4.6	0.0	0.0

Trained on subsets 1 and 5 tested in subsets 2, 3, 4

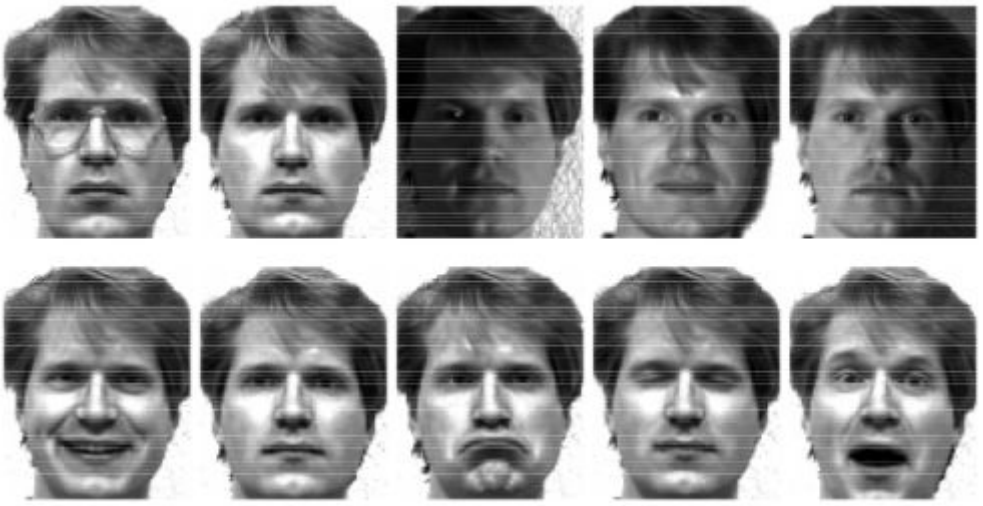
Error Ratio (%)		
Subset 4	**Subset 3**	**Subset 2**
1.2	0.0	0.0

Experiment 2: Lighting and Facial Expression

Tested on full head shots and closely cropped faces.

Faces with glasses, ambient light, many points light from different angles, and five different facial expressions.

Using leaving one out strategy.



Trained on Subset 1 and Tested on Subsets 1, 2, 3

Error Ratio	
Full crop	**Close crop**
0.6	7.3

References

https://en.wikipedia.org/wiki/Facial_recognition_system
https://users.ece.cmu.edu/~saswin/files/paper/2014/RobustFR_TIP2014.pdf
http://www.ics.uci.edu/~xzhu/paper/face-cvpr12.pdf
https://www.raspberrypi.org/products/camera-module/
http://docs.opencv.org/3.1.0/da/d60/tutorial_face_main.html#gsc.tab=0
http://opencvfacedetect.blogspot.com.eg/2010/10/face-detectionfollowed-by-eyesnose.html
https://alitarhini.wordpress.com/2011/02/27/face-recognition-image-preprocessing/
http://www.scholarpedia.org/article/Fisherfaces
http://www.cs.columbia.edu/~belhumeur/journal/fisherface-pami97.pdf
https://www.youtube.com/watch?v=5vfIM46Hvbk
https://www.youtube.com/watch?v=x8W_htbct3U