```
/*
Chris Gonzales
very rushed version of my project so far

Credit to openCV sample for the barebones detectAndDisplay
*/

#include <opencv/cv.h>
#include <opencv/highgui.h>
#include <opencv2/core/core.hpp> //basic OpenCV structures
#include <opencv2/highgui/highgui.hpp> //no clue what this shit is
#include <opencv2/imgproc/imgproc.hpp> //image processing stuff
#include <opencv2/objdetect/objdetect.hpp> //object detewction

#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <math.h>


using namespace std;
using namespace cv;

//face data structure. All I have now is face center and eyes so i'll add more to this at one point
struct faceData{
    int faceCenterx, faceCentery; //cause your face only have once center.
    int eyex[2], eyey[2];       //cause you have two eyes.
};

//function prototypes
faceData detectAndDisplay(Mat frame);
//faceData detectAndDisplay(Mat frame, faceData x);
void detectAndDisplay2(Mat frame);
void stillImage(String filename);
float distForm(int x1, int x2, int y1, int y2);
float percDiff(float a, float b);
float faceDiff(faceData a, faceData b);


String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
CascadeClassifier face_cascade;
CascadeClassifier eyes_cascade;
//main
int main(){
    //loading the eye and face cascades
    if (!face_cascade.load(face_cascade_name)){ printf("--(!)Error loading\n"); return 1000000; };
    if (!eyes_cascade.load(eyes_cascade_name)){ printf("--(!)Error loading\n"); return 9999999; };

    //GUI temporary replacement.
    cout << "1) Video Capture" << endl << "2) Still Image" << endl << "3) Exit" << endl;
    int in;
    cin >> in;
    //video capture
    //this actually partly works right now
    if (in == 1){
        //get the image from camera 0. Which is the onyl camera I have..
        CvCapture* capture = cvCaptureFromCAM(0);

        //Mat is basically a matrix of colors RGB. Holds images.
        Mat frame;
        if (capture)
        {
            while (true)
            {
```

```
                //capture images from the camera and store it in the frame
                frame = cvQueryFrame(capture);

                if (!frame.empty())
                {
                    //go to detectAndDisplay2 for proper explanation
                    detectAndDisplay2(frame);
                }
                else
                {
                    printf(" --(!) No captured frame -- Break!"); break;
                }

                //press C if you're tired of looking at your image on the camera
                int c = waitKey(10);
                if ((char)c == 'c') { break; }
            }
        }
        return 99;
    }
    //Still image
    else if (in == 2){
        cout << "1) Create Profile" << endl << "2) Recognize" << endl;
        int input;
        cin >> input;
        //create face data and saves it in the txt "database" quotation.
        if (input == 1){
            Mat frame;
            String ifname, ipname;
            faceData face;
            //input the filename of the image
            cout << "Enter the filename: ";
            cin >> ifname; cout << endl;
            //get the name for the profile
            cout << "Enter the person's name: ";
            cin >> ipname; cout << endl;

            //convert String to const char *
            const char * fname = ifname.c_str();

            //load the image from file
            IplImage *img = cvLoadImage(fname);

            //conver the image into the matrix format
            frame = Mat(img, false);

            //check the function itself for explanation
            face = detectAndDisplay(frame);
            while (true){
                int c = waitKey(10);
                if ((char)c == 'c') { break; }
                //press C if you're done
            }

            //very tedious saving data into a text file
            //i was tempted to save just the face but I'd run into trouble of having to
            //read two files at once and extract data from each one. I figured it's easier
            //to just extract the data, save the data and just pull it out later.
            //i don't think it worked as I wanted it to.. but it works enough and that's all that matters.
            ofstream database;
            database.open("database.txt", ios::app);
            database << ipname << endl;
            database << face.faceCenterx << endl; database << face.faceCentery << endl;
            database << face.eyex[0] << endl;  database << face.eyey[0] << endl;
            database << face.eyex[1] << endl;  database << face.eyey[1] << endl;
```

```
            database.close();

            return 5000;
        }

        //Compare the still image input to your database for recognition.. Kind of.
        else if (input == 2){
            Mat frame;
            String ifname, ipname, name, finName;
            faceData tempFace, loadFace;
            float difference, fdiff;
            fdiff = 99999;

            cout << "Enter filename: ";
            cin >> ifname; cout << endl;

            const char * fname = ifname.c_str();
            IplImage *img = cvLoadImage(fname);

            frame = Mat(img, false);

            loadFace = detectAndDisplay(frame);
            while (true){
                int c = waitKey(10);
                if ((char)c == 'c') { break; }
            }

            ifstream file;
            file.open("database.txt");
            //now begins the tedious job of pulling out each line from the "database"
            while (!file.eof()){
                String temp;
                file >> temp;
                name = temp;
                file >> temp;
                tempFace.faceCenterx = atoi(temp.c_str()); //converts string to integer
                file >> temp;
                tempFace.faceCentery = atoi(temp.c_str());

                for (int i = 0; i < 2; i++){
                    file >> temp;
                    tempFace.eyex[i] = atoi(temp.c_str());
                    file >> temp;
                    tempFace.eyey[i] = atoi(temp.c_str());
                }
                //for some reason, eof doesn't end when the file ends and try to iterate one more time
                //so to stop that I added this if statement to break when there's no name
                if (name.empty()){ break; }

                //returns the %difference go to the function for explanation
                difference = faceDiff(loadFace, tempFace);

                //find the smallest difference.
                if (difference < fdiff){
                    fdiff = difference;
                    finName = name;
                }
            }
            //prints out on console. I'll have this output directly on the image at one point. Just like the capture
            cout << "Closest Match:\t" << finName << endl;
            cout << "Approximately:\t" << fdiff << "%" << endl;

        }
        else{
```

```
            cout << "Incorrect input" << endl;
            return 10000;
        }
        return 100;
    }
    //exit
    else{
        return 100;
    }
    return 5000;
}

//ignore the amount of returns. I was typing this in a rush and at one point I forgot
//which ones I don't really need. so I'm just gonna leave that there for now.

//detects faces for still images
faceData detectAndDisplay(Mat frame)
{
    std::vector<Rect> faces; //my program only works for one face right now but detectMultiScale r
                             //for some reason only work with vector of faces.
    Mat frame_gray;          //placeholder for the grayscale of image
    faceData a;              //faceData structure.

    cvtColor(frame, frame_gray, CV_BGR2GRAY);   //converts the frame to grayscale.
    equalizeHist(frame_gray, frame_gray);       //equalizeHistogram. Improves contrast in image. Used for multiscale.

    //detects faces
    face_cascade.detectMultiScale(frame_gray, faces, 1.1, 2, 0 | CV_HAAR_SCALE_IMAGE, Size(30, 30));

    for (size_t i = 0; i < faces.size(); i++)
    {
        //finds the center of the rectangle for the current face
        Point center(faces[i].x + faces[i].width*0.5, faces[i].y + faces[i].height*0.5);
        //draws the circle on the face
        ellipse(frame, center, Size(faces[i].width*0.5, faces[i].height*0.5), 0, 0, 360, Scalar(255, 0, 255), 4, 8, 0);

        //record the values for the face center
        a.faceCenterx = center.x;
        a.faceCentery = center.y;

        //saves the rectangle of Interest. The part with  just the face
        Mat faceROI = frame_gray(faces[i]);
        std::vector<Rect> eyes; //vector of eyes

        //detects eyes
        eyes_cascade.detectMultiScale(faceROI, eyes, 1.1, 2, 0 | CV_HAAR_SCALE_IMAGE, Size(30, 30));

        //looks for eyes inside ONLY the area of interest.
        for (size_t j = 0; j < eyes.size(); j++)
        {
            Point center(faces[i].x + eyes[j].x + eyes[j].width*0.5, faces[i].y + eyes[j].y + eyes[j].height*0.5);
            int radius = cvRound((eyes[j].width + eyes[j].height)*0.25);
            //draws the circle for each Eye
            circle(frame, center, radius, Scalar(255, 0, 0), 4, 8, 0);

            //record the eye values
            a.eyex[j] = center.x;
            a.eyey[j] = center.y;

        }
    }

    //displays the image with the doodles
```

```
    imshow("hello", frame);
    //return the face data for evaluation and stuff
    return a;
}

//detects faces and displays them with doodles for video capture
void detectAndDisplay2(Mat frame)
{
    std::vector<Rect> faces;
    Mat frame_gray;
    faceData a, b;

    for (int i = 0; i < 2; i++){
        a.faceCenterx = a.faceCentery = a.eyex[i] = a.eyey[i] = 0;
    }

    cvtColor(frame, frame_gray, CV_BGR2GRAY);
    equalizeHist(frame_gray, frame_gray);

    face_cascade.detectMultiScale(frame_gray, faces, 1.1, 2, 0 | CV_HAAR_SCALE_IMAGE, Size(30, 30));

    for (size_t i = 0; i < faces.size(); i++)
    {
        Point center(faces[i].x + faces[i].width*0.5, faces[i].y + faces[i].height*0.5);
        ellipse(frame, center, Size(faces[i].width*0.5, faces[i].height*0.5), 0, 0, 360, Scalar(255, 0, 255), 4, 8, 0);

        //record the values for the face center
        a.faceCenterx = center.x;
        a.faceCentery = center.y;

        Mat faceROI = frame_gray(faces[i]);
        std::vector<Rect> eyes;

        eyes_cascade.detectMultiScale(faceROI, eyes, 1.1, 2, 0 | CV_HAAR_SCALE_IMAGE, Size(30, 30));

        for (size_t j = 0; j < eyes.size(); j++)
        {
            Point center(faces[i].x + eyes[j].x + eyes[j].width*0.5, faces[i].y + eyes[j].y + eyes[j].height*0.5);
            int radius = cvRound((eyes[j].width + eyes[j].height)*0.25);
            circle(frame, center, radius, Scalar(255, 0, 0), 4, 8, 0);

            //record the eye values
            a.eyex[j] = center.x;
            a.eyey[j] = center.y;

        }
    }

    //almost a carbon copy of some code from above. more or less a copy and paste.
    ifstream file;
    String c, d;
    float e,f;
    f = 999;
    file.open("database.txt");

    //forgive the very descriptive names
    while (!file.eof()){

        String temp;
        file >> temp;
        c = temp;
        file >> temp;
        b.faceCenterx = atoi(temp.c_str());
        file >> temp;
```

```
        b.faceCentery = atoi(temp.c_str());
        for (int i = 0; i < 2; i++){
            file >> temp;
            b.eyex[i] = atoi(temp.c_str());
            file >> temp;
            b.eyey[i] = atoi(temp.c_str());
        }
        if (c.empty()){ break; }

        e = faceDiff(a, b);
        //cout << c << "---" << e << endl;

        if (e < f){
            f = e;
            d = c;
        }

    }
    //writes on the image.
    Point p(30, 50); Point p2(30, 100);
    putText(frame, d, p, CV_FONT_HERSHEY_SIMPLEX, 1.5, Scalar(255, 0, 0), 1, 8, false);
    ostringstream stringer;
    stringer << f << "%";
    string str = stringer.str();
    cout << str << endl;
    putText(frame, str, p2, CV_FONT_HERSHEY_SIMPLEX, 1.5, Scalar(255, 0, 0), 1, 8, false);

    //displays the image
    imshow("hello", frame);

}
//distance formula
float distForm(int x1, int x2, int y1, int y2){
    float dist, a, b;

    a = x2 - x1;
    a = a*a;
    b = y2 - y1;
    b = b*b;

    dist = sqrt(a + b);

    return dist;
}
//percent difference
float percDiff(float a, float b){
    float c, ave;
    ave = (a + b) / 2;
    c = abs((a - b) / ave);

    return c;
}

float faceDiff(faceData a, faceData b){
    //aRef and bRef are the distance between each eye. Will be used as reference for the ratios
    float aRef = distForm(a.eyex[1], a.eyex[0], a.eyey[1], a.eyey[0]);
    float bRef = distForm(b.eyex[1], b.eyex[0], b.eyey[1], b.eyey[0]);

    //finds distance of each eye to center
    float eyeDistA1 = distForm(a.faceCenterx, a.eyex[0], a.faceCentery, a.eyey[0]);
    float eyeDistA2 = distForm(a.faceCenterx, a.eyex[1], a.faceCentery, a.eyey[1]);


    //again
    float eyeDistB1 = distForm(b.faceCenterx, b.eyex[0], b.faceCentery, b.eyey[0]);
    float eyeDistB2 = distForm(b.faceCenterx, b.eyex[1], b.faceCentery, b.eyey[1]);
```

```


    float var, varr;
    //calculates difference between each respective eye
    var = percDiff(eyeDistA1/aRef, eyeDistB1/bRef);
    varr = percDiff(eyeDistA2/aRef, eyeDistB2/bRef);

    /*
    add some mroe stuff here at one point.
    */


    cout << endl;
    //that's pretty much it.
    float ret = (var + varr) / 2;

    return ret * 100;

}
```