Face Recognition Code

#include<iostream>

#include<string.h>

#include <ctime>

#include <time.h>

#include "opencv2/core/core.hpp"

#include "opencv2/contrib/contrib.hpp"

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include "opencv2/objdetect/objdetect.hpp"

#define MAX\_DATE 12

#include<fstream>

#include<sstream>

using namespace std;

using namespace cv;

int label,n;

static Mat MatNorm(InputArray \_src)

{

Mat src = \_src.getMat();

// Create and return normalized image:

Mat dst;

switch (src.channels()) {

case 1:

cv::normalize(\_src, dst, 0, 255, NORM\_MINMAX, CV\_8UC1);

break;

case 3:

cv::normalize(\_src, dst, 0, 255, NORM\_MINMAX, CV\_8UC3);

break;

default:

src.copyTo(dst);

break;

}

return dst;

}

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 = "Invalid File";

CV\_Error(CV\_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()));

}

}

}

void eigenFaceTrainer(){

/\*in this two vector we put the images and labes for training\*/

vector<Mat> images;

vector<int> labels;

string fn\_name = "Database2.csv";

try {

read\_csv(fn\_name, images, labels);

} catch (cv::Exception& e) {

cerr << "Error opening file \"" << fn\_name << "\". Reason: " << e.msg << endl;

// nothing more we can do

exit(1);

}

cout<< "Starting training to Training data set........"<<endl;

//create algorithm eigenface recognizer

Ptr<FaceRecognizer> model = createEigenFaceRecognizer();

//train data

model->train(images, labels);

model->save("/home/prateek/Desktop/Database/YMLfiles/eigen.yml");

cout << "Training finished...." << endl;

waitKey(10000);

}

void fisherFaceTrainer()

{

/\*in this two vector we put the images and labes for training\*/

vector<Mat> images;

vector<int> labels;

string fn\_name = "Database2.csv";

//cout<<"done";

try {

read\_csv(fn\_name, images, labels);

} catch (cv::Exception& e) {

cerr << "Error opening file \"" << fn\_name << "\". Reason: " << e.msg << endl;

// nothing more we can do

exit(1);

}

cout<< "Starting training to Training data set........"<<endl;

Ptr<FaceRecognizer> model = createFisherFaceRecognizer();

model->train(images, labels);

//int height = images[0].rows;

model->save("/home/prateek/Desktop/Database/YMLfiles/Fisherface.yml");

cout << "Training finished...." << endl;

waitKey(10000);

}

void LBPHFaceTrainer()

{

/\*in this two vector we put the images and labes for training\*/

vector<Mat> images;

vector<int> labels;

string fn\_name = "m\_database.csv";

//cout<<"done";

try {

read\_csv(fn\_name, images, labels);

} catch (cv::Exception& e) {

cerr << "Error opening file \"" << fn\_name << "\". Reason: " << e.msg << endl;

// nothing more we can do

exit(1);

}

cout<< "Starting training to Training data set........"<<endl;

Ptr<FaceRecognizer> model = createLBPHFaceRecognizer();

model->train(images, labels);

model->save("/home/prateek/Desktop/Database/YMLfiles/LBPHface.yml");

cout << "training finished...." << endl;

waitKey(10000);

}

std::string get\_date(void)

{

time\_t now;

char the\_date[MAX\_DATE];

the\_date[0] = '\0';

now = time(NULL);

if (now != -1)

{

strftime(the\_date, MAX\_DATE, "%d\_%h", gmtime(&now));

}

return std::string(the\_date);

}

int FaceRecognition(){

cout << "start recognizing..." << endl;

//load pre-trained data sets

//Ptr<FaceRecognizer> model = createFisherFaceRecognizer();

Ptr<FaceRecognizer> model = createLBPHFaceRecognizer();

//model->load("/home/prateek/Desktop/Database/YMLfiles/Fisherface.yml");

model->load("/home/prateek/Desktop/Database/YMLfiles/LBPHface.yml");

//Mat testSample = imread("subject/s1/1.pgm",0);

Mat testSample = imread("N\_subject/s1/1c.jpg",0);

// Mat smaple = imread("prateek.jpg");

if (testSample.empty()) //check whether the image is loaded or not

{

cout << "Error : Image cannot be loaded..!!" << endl;

//system("pause"); //wait for a key press

return -1;

}

int img\_width = testSample.cols;

int img\_height = testSample.rows;

string classifier = "haarcascade\_frontalface\_default.xml";

CascadeClassifier face\_cascade;

string window = "Recognition";

if (!face\_cascade.load(classifier)){

cout << " Error loading file" << endl;

return -1;

}

VideoCapture cap(1);

if (!cap.isOpened())

{

cout << "Camera NOT Accessible" << endl;

return -1;

}

Size S = Size((int) cap.get(CV\_CAP\_PROP\_FRAME\_WIDTH), // Acquire input size

(int) cap.get(CV\_CAP\_PROP\_FRAME\_HEIGHT));

/\* int ex = static\_cast<int>(cap.get(CV\_CAP\_PROP\_FOURCC));

union { int v; char c[5];} uEx ;

uEx.v = ex; // From Int to char via union

uEx.c[4]='\0';\*/

// VideoWriter outputVideo; // Open the output

// if (askOutputType)

// outputVideo.open(NAME, ex=-1, inputVideo.get(CV\_CAP\_PROP\_FPS), S, true);

//else

// outputVideo.open("output.avi", -1, cap.get(CV\_CAP\_PROP\_FPS), S, true);

// VideoWriter Vwrite("1.avi",cap.get(CV\_CAP\_PROP\_FOURCC),cap.get(CV\_CAP\_PROP\_FPS),frameSize,true); // initilaise the object

/\* if ( !outputVideo.isOpened() ) //if not initialize the VideoWriter successfully, exit the program

{

cout << "ERROR: Failed to write the video" << endl;

return -1;

} \*/

// Output File

string s= get\_date();

std::string file= s + ".txt";

const char \*cstr = file.c\_str();

ofstream myfile;

myfile.open(cstr);

int temp[5],count[5],max;

int arr1[1000];

int cou=0;

namedWindow(window, 1);

//long count = 0;

while (true)

{

vector<Rect> faces;

Mat frame;

Mat grayo;

//Mat gray;

Mat original;

cap >> frame;

if (!frame.empty()){

//clone from original frame

original = frame.clone();

//convert image to gray scale

// Mat gray;

if (original.channels() == 3) {

cvtColor(original, grayo, CV\_BGR2GRAY);

}

else if (original.channels() == 4) {

cvtColor(original, grayo, CV\_BGRA2GRAY);

}

Mat grays;

// Histogram Normalization

equalizeHist(grayo, grays);

Mat gray= Mat(70,70,CV\_8U);

// Applying smoothing filter

try{

bilateralFilter(grays,gray,0,20,2);

} catch (cv::Exception& e) {

cerr << "Error in filtering : " << " Reason: " << e.msg << endl;

// nothing more we can do

exit(1);

}

//detect face in gray image

face\_cascade.detectMultiScale(gray, faces, 1.3, 5);

//number of faces detected

cout << faces.size() << " faces detected" << endl;

n = faces.size();

// ID(Roll Number) of person

string ID = "";

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

{

//region of interest

Rect face\_i = faces[i];

//crop the roi from grya image

Mat face = gray(face\_i);

//resizing the cropped image to suit to database image sizes

// Mat face\_resized;

//cv::resize(face, face\_resized, Size(img\_width,img\_height), 1.0, 1.0, INTER\_CUBIC);

//recognizing what faces detected

int label =-1;

double confidence=0;

//int label=model->predict(face\_resized);

label = model->predict(face);

//drawing green rectagle in recognize face

rectangle(original, face\_i, CV\_RGB(0, 255, 0), 1);

switch(label)

{

case 0: ID= "107112077"; break; // 0 - Samyuktha

case 1: ID= "107112061"; break; // 1- Manish keshri

case 2: ID= "107112062"; break; // 2- Samyuktha

case 4: ID= "107112000"; break; // 3-Pankaj

case 3: ID= "Moorthi Sir"; break;

// case 5: ID= "107112077"; break;

default: ID= "UNKNOWN";

}

//cout<< label;

int pos\_x = std::max(face\_i.tl().x - 10, 0);

int pos\_y = std::max(face\_i.tl().y - 10, 0);

arr1[cou++]=label;

//name the person who is in the image

putText(original, ID, Point(pos\_x, pos\_y),FONT\_HERSHEY\_COMPLEX\_SMALL, 1.0, CV\_RGB(0, 255, 0), 2.0);

}

// outputVideo << original;

//display to the winodw

cv::imshow(window, original);

}

if (waitKey(30) >= 0) break;

}

// To get real tie and date

for(int j=0;j<5; j++)

count[j]=0;

for(int i=0;i<cou;i++)

{

switch(arr1[i])

{

case 0: count[0]++;

break;

case 1: count[1]++;

break;

case 2: count[2]++;

break;

case 3: count[3]++;

break;

case 4: count[4]++;

break;

}

}

for(int i=0; i<5; i++)

{ max = 0;

for(int k=0; k<5; k++)

{

if(count[k]>max)

max = count[k];

}

for(int j=0; j<5; j++)

{

if(count[j]==max)

{

count[j]=0;

temp[i] = j;

break;

}

}

}

for(int i=0; i<n; i++)

{

switch(temp[i])

{

case 0: myfile << "107112077\n"; break;

case 1: myfile << "107112061\n"; break;

case 2: myfile << "107112062\n"; break;

case 3: myfile << "Moorthi sir\n"; break;

case 4: myfile << "107112000\n"; break;

}

}

myfile.close();

return 0;

//return label;

}