ДОДАТОК А

Лістинг програми

```
#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"
#include <iostream>
#include <conio.h>
#include <fstream>
#include <sstream>

using namespace cv;
using namespace std;

static void read_file(const string& filename, vector<Mat>&
images, vector<int>& labels, char separator = ';') {
     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(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()));
          }
     }
}

int  FaceRecognition() {

     cout << "start recognizing..." << endl;
     Ptr<FaceRecognizer>  model = createFisherFaceRecognizer();
     model->load("C:/opencv/astfisherface.yml");

     string classifier = "haarcascade_frontalface_default.xml";

     CascadeClassifier face_cascade;
     string window = "Capture - face detection";

     if (!face_cascade.load(classifier))
     {
          cout << " Error loading file" << endl;
          return -1;
```

```
	}

	VideoCapture cap(0);

	if (!cap.isOpened())
	{
		cout << "exit" << endl;
		return -1;
	}

	namedWindow(window, 1);
	long count = 0;

	while (true)
	{
		vector<Rect> faces;
		Mat frame;
		Mat graySacleFrame;
		Mat original;

		cap >> frame;
		count = count + 1;

		if (!frame.empty()) {

			original = frame.clone();

			cvtColor(original, graySacleFrame, CV_BGR2GRAY);
			equalizeHist(graySacleFrame, graySacleFrame);

			face_cascade.detectMultiScale(graySacleFrame,
faces, 1.1, 3, 0 | CV_HAAR_SCALE_IMAGE, Size(100,100),
Size(300,300));

			cout << faces.size() << " faces detected" <<
endl;
			std::string frameset = std::to_string(count);
			std::string faceset =
std::to_string(faces.size());

			int width = 0, height = 0;

			string Pname = "";

			for (int i = 0; i < faces.size(); i++)
			{
				Rect face_i = faces[i];

				Mat face = graySacleFrame(face_i);

				Mat face_resized;
				cv::resize(face, face_resized, Size(218,
218), 1.0, 1.0, INTER_CUBIC);
```

```
                    int label = -1; double confidence = 0;
                    model->predict(face_resized, label,
confidence);

                    cout << " confidencde " << confidence <<
endl;

                    rectangle(original, face_i, CV_RGB(0, 255,
0), 1);
                    string text = "Detected";
                    if (label == 1) {
                         text = "Evgenia";
                    }
                    else {
                         text = "Unknown";
                    }


                    int pos_x = std::max(face_i.tl().x - 10, 0);
                    int pos_y = std::max(face_i.tl().y - 10, 0);

                    putText(original, text, Point(pos_x, pos_y),
FONT_HERSHEY_COMPLEX_SMALL, 1.0, CV_RGB(0, 255, 0), 1.0);

               }

               cv::imshow(window, original);
          }
          int c = waitKey(10);

          if (27 == char(c))
          {
               break;
          }

     }
}

void eigenFaceTrainer() {
     vector<Mat> images;
     vector<int> labels;

     try {
          string filename = "C:/opencv/project.txt";
          read_file(filename, images, labels);

          cout << "size of the images is " << images.size() <<
endl;
          cout << "size of the labes is " << labels.size() <<
endl;
          cout << "Training begins...." << endl;
     }
```

```
	catch (cv::Exception& e) {
		cerr << " Error opening the file " << e.msg << endl;
		exit(1);
	}

	Ptr<FaceRecognizer>  model = createEigenFaceRecognizer();
	model->train(images, labels);

	model->save("C:/opencv/projecteigehface.yml");

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

	waitKey(10000);
}

void fisherFaceTrainer() {
	vector<Mat> images;
	vector<int> labels;

	try {
		string filename = "C:/opencv/project.txt";
		read_file(filename, images, labels);

		cout << "size of the images is " << images.size() <<
endl;
		cout << "size of the labes is " << labels.size() <<
endl;
		cout << "Training begins...." << endl;
	}
	catch (cv::Exception& e) {
		cerr << " Error opening the file " << e.msg << endl;
		exit(1);
	}


	Ptr<FaceRecognizer> model = createFisherFaceRecognizer();

	model->train(images, labels);

	model->save("C:/opencv/projectfisherface.yml");

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

	Mat eigenvalues = model->getMat("eigenvalues");
	Mat W = model->getMat("eigenvectors");
	Mat mean = model->getMat("mean");
	waitKey(10000);
}

void LBPHFaceTrainer() {

	vector<Mat> images;
	vector<int> labels;
```

```
     try {
          string filename = "C:/opencv/project.txt";
          read_file(filename, images, labels);

          cout << "size of the images is " << images.size() <<
endl;
          cout << "size of the labes is " << labels.size() <<
endl;
          cout << "Training begins...." << endl;
     }
     catch (cv::Exception& e) {
          cerr << " Error opening the file " << e.msg << endl;
          exit(1);
     }

     Ptr<FaceRecognizer> model = createLBPHFaceRecognizer();

     model->train(images, labels);

     model->save("C:/opencv/projectLBPHface.yml");

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

     waitKey(10000);
}

int main() {
     int h, value;
     cout << "The face recognition program welcomes you.\nFor an
action, enter the appropriate number to him. Possible actions:
\n1. Cascade training by FisherFace method. \n2. Cascade
training by EigenFace method. \n3. Cascade training by the LBPH
method. \n4. Demonstration of the application. \nChoose an
option for further work:" << endl;
     cin >> h;
     switch (h)
     {
     case 1: fisherFaceTrainer(); break;
     case 2 : eigenFaceTrainer(); break;
     case 3 : LBPHFaceTrainer(); break;
     case 4 : value = FaceRecognition(); break;
     default : cout << "\nIncorrect number entered."; break;
     }
     return 0;
}
```

ДОДАТОК Б

Демонстраційний матеріал

Формат	Позиція.	Позначення	Найменування	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮИК.502840.039ПЗ	Пояснювальна записка	72	с.
			Інші документи		
А4	2		Лістинг програми	5	с.
А4	3		Демонстраційний матеріал	16	с.

					ГЮИК.502840.039ВД			
Змін	А	№ докум.	Підпис	Дата				
Розроб.		Нестеренко В.В.			Модернізація автоматизованої системи ідентифікації особи на виробничому підприємстві шляхом розроблення комбінованого каскаду класифікаторів Відомість кваліфікаційної роботи	Літера	Аркуш	Аркушів
Переві		Аллахверанов Р.Ю.				н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.конт.		Демська Н.П.						
Затв.		Невлюдов І.Ш.						