for Land, Sea and Air, GT2009, June 8-12, 2009, Orlando, FL, USA.

Paşayev, A., Abdullayev, P. and Samedov, A. (2018) Sıvı yakıtlı roket motorunun itme odasının geliştirilmiş tasarım yöntemi, SAVTEK 2018, 9.Savunma Teknolojileri Kongresi, ODTÜ, Ankara, 27-29 Haziran, 2018.

Rizkalla, O., Chinitz, W. and Erdos, J.I. (1990) Calculated Chemical and Vibrational Non equilibrium Effects in Hypersonic Nozzles, Journal of propulsion and power, pp.50-57.

Sutton, G.P. (2010) Rocket Propulsion Elements, New York: John Wiley & Sons, Inc.,

Vasiliev, A.P., Kudryavtsev, V.M., Kuznetsov, V.A., Kurpatenkov, V.D., Obelnitsky, A.M., Polyaev, V.M., Poluyan, B.Y. (1983) Fundamentals of the theory and calculation of liquid rocket engines, Textbook. Edited by V.M. Kudryavtsev. Moscow, High School, 3rd edition, revised and enlarged, 703 p., in russian.

Zebbiche, T. (2011) Stagnation temperature effect on the supersonic axisymmetric minimum length nozzle design with application for air, Adv. Space Res. 48 (10), 1656–1675.

DEVELOPING AN APPLICATION FOR FACIAL IDENTIFICATION IN THE JAVA PROGRAMMING LANGUAGE

Boranbayev S.N.,
Kabdulkarimov Y.Z.
Eurasian National University named after L.N. Gumilyov, Nur-Sultan

Abstract. This article describes a developed application for identifying individuals in the Java programming language. For recognition of image templates, the OpenCV library was selected. Based on the methods that the OpenCV library classes offer, a program with a graphical user interface for detecting faces has been developed.

Keywords: identification, recognition, image, pattern, processing, confidentiality, security.

1.Introduction

Alan Kay said: “People who are really serious about their software must create their own hardware” [1]. This expression is also suitable for ensuring your own safety. A country that is truly serious about its own security must create its own security software and hardware. This means that each country must create its own devices for the recognition, processing, identification and analysis of data obtained from video and photo cameras of outdoor surveillance and other monitoring devices for private and public sectors.Since, if these devices were purchased abroad, this can lead to information leakage, because the device can be controlled remotely by the manufacturers of this device. To ensure confidentiality and complete control of security systems by your own government agencies, you must create your own software for the recognition, processing, identification and analysis of information. Therefore, creating an application for recognizing and identifying certain image patterns, such as people's faces, partially solves this problem.

2.Development of an application for facial identification

In the process of developing a program for face recognition, the following sources were analyzed:

- Existing face recognition approaches used by Google, Apple and Samsung to authenticate users and face recognition in photos and videos; [2-4]
- modern principles of security systems for face recognition and identification; [5,6]
- neural networks that are used to process and analyze video and photos; [7.8]

Algorithms based on existing approaches have been developed. A high-level Java programming language is used to create this application. The choice of this particular programming language is that the operating system of many devices, such as cell phones, televisions, drones, camcorders, cameras is Android, which is written in Java. Thus, it is possible to integrate the created program into a device that uses Android. Also programs written in Java are portable. After compiling the program on the computer, it is possible to run the bytecode of the program on all devices that have a Java Virtual Machine. [9] Next, it was necessary to choose a library that will recognize image templates. Currently, Java does not have its own libraries for recognizing image templates. A third-party OpenCV library (Open Source Computer Vision Library, an open-source computer vision library) was chosen - a library of computer-vision algorithms, image processing and general-purpose open-source numerical algorithms. [10] Implemented in C / C ++, also developed for Python, Java, Ruby, Matlab, Lua, and other languages. It can be freely used for academic and commercial purposes. The authors of this library are Intel Corporation. [11] OpenCV includes the following tools:

- image processing (filtering, geometric transformations, color space conversion);
- input / output of images and videos, machine learning models (SVM, decision trees, learning with stimulation);
- recognition and description of flat primitives;
- motion analysis and object tracking (optical flow, motion patterns, background removal);
- detection of objects in the image (finding faces using the Viola-Jones algorithm, recognizing HOG people), calibrating the camera, searching for stereo matching and 3D processing elements. [12]

Next, the task was to install and connect the OpenCV library during application development. The tricky part was connecting the library to the application. Since there was no detailed user manual. However,

with the release of version 11 of Java, its own Java library, JavaFx, a 3rd generation library for creating programs with a graphical user interface, was not included in the standard list of Java libraries and also became third-party. Therefore, Java has written very detailed instructions on how to connect and use third-party libraries when developing programs in various integrated development environments, including Eclipse, in which the application was developed.

These instructions were also suitable for connecting OpenCV. After connecting and configuring all the necessary components, the next stage began - creating the application.

Based on the methods that the OpenCV library classes offer, a program was developed with a graphical user interface for detecting faces from a web camera and photos.

The graphical interface of the program consists of two buttons of the checkbox category for selecting face detection algorithms, Haar and Local Binary Templates (LBP). Also, the Start camera button, which starts the web camera only after selecting one of the recognition algorithms. In the center is a frame in which video from a web camera will be broadcast. The graphical interface was created using SceneBuilder 2.0 (Figure 1).

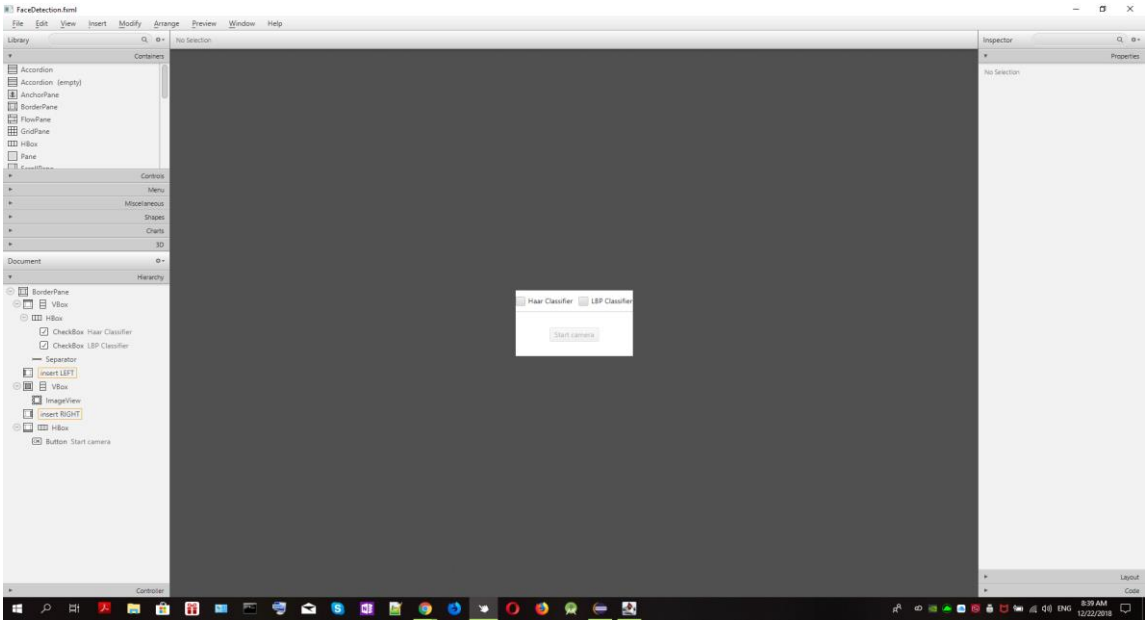
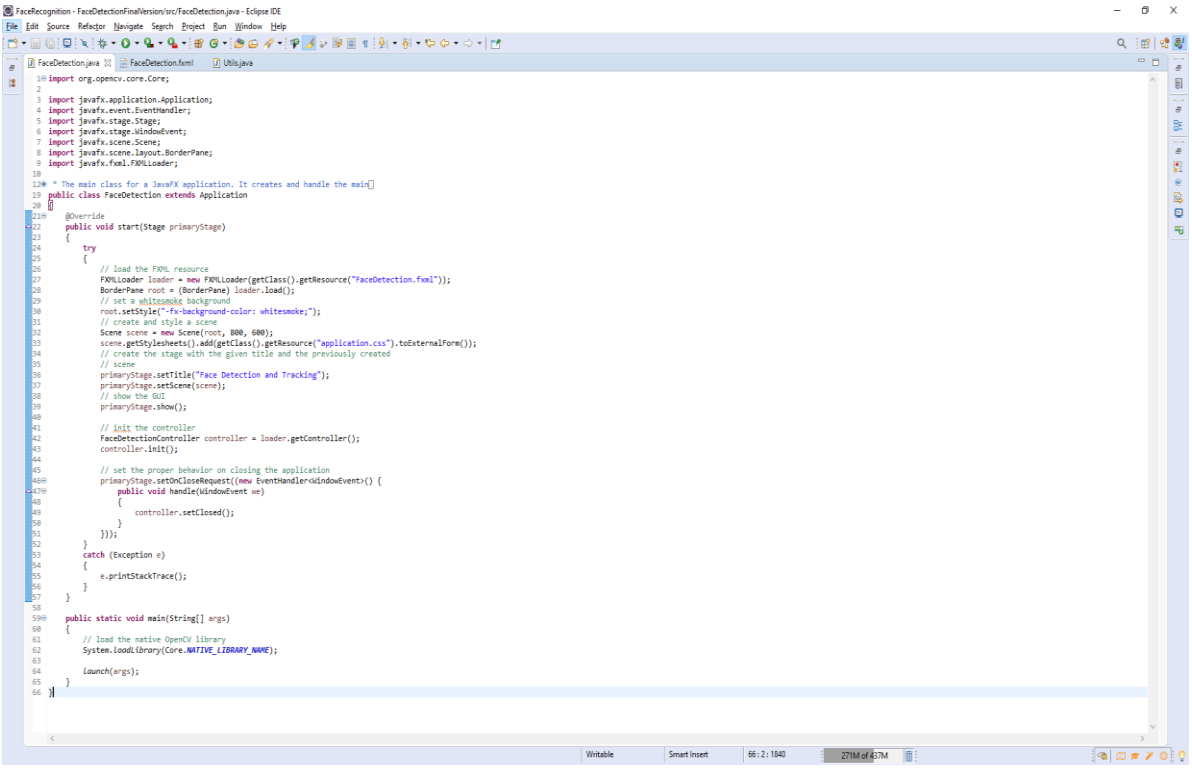


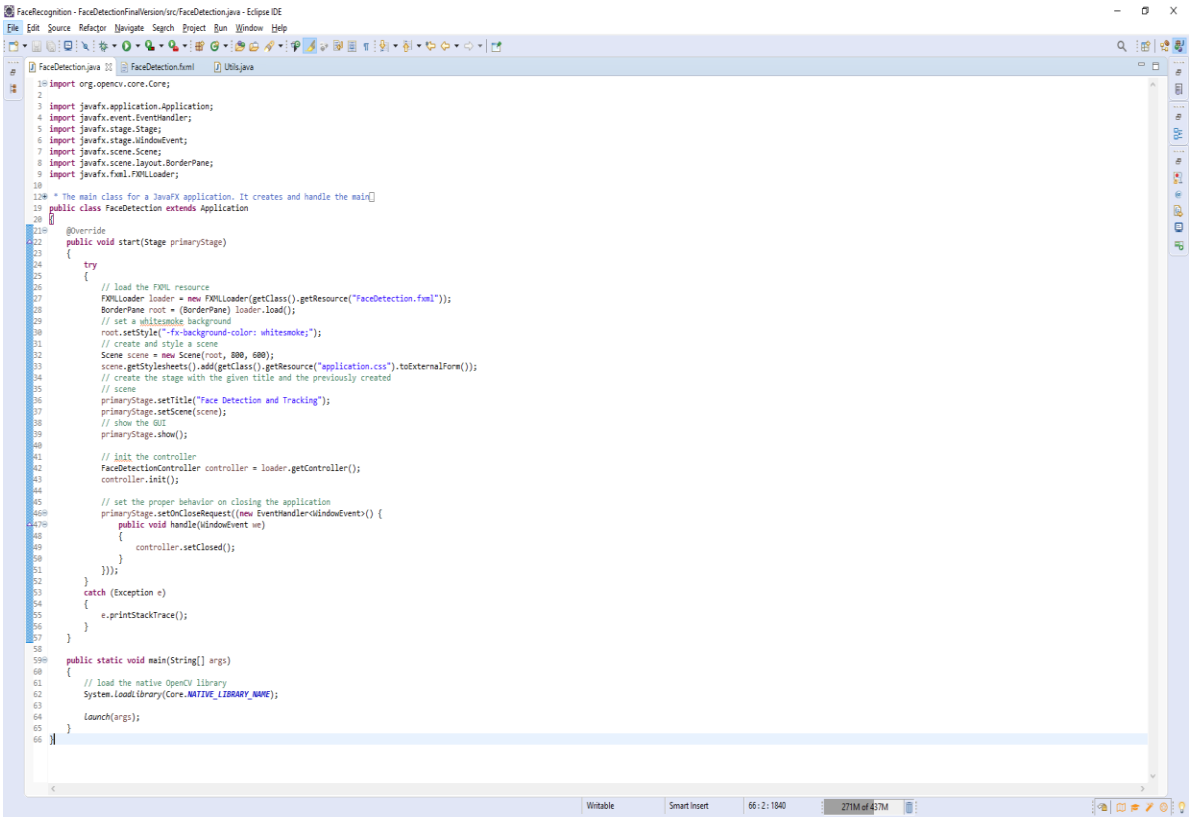
Figure 1. Creating a graphical user interface for the program using SceneBuilder 2.0.

The program itself consists of 3 classes FaceDetection.java, FaceDetectionController.java, Utils.java and a FaceDetection.fxml file created using SceneBuilder 2.0. These classes are illustrated in Figures 2, 3, 4. The Utils.java class stores methods for using OpenCV objects in JavaFX. The FaceDetectionController.java class is responsible for the type of application, the application logic is also implemented here. This class contains 7 class variables: a button for turning the camera on and off, a frame where the image from the camera is illustrated, 2 buttons for choosing the LBP or Haar algorithms. Methods for starting or stopping the camera (startCamera), receiving a video stream (grabFrame), an algorithm for controlling, detecting, tracking faces (detectAndDisplay) are also stored. This method uses an object of class Mat to obtain an image from the camera. It then converts the image into an object for LBP or Haar algorithms. After this object has been processed, coordinates are recorded. Next, rectangular frames are created that will indicate faces in the camera image. These rectangles are stored in the array. The updateImageView method retrieves new frames received from the camera. The haarSelected method loads a trained set of algorithms for face recognition based on the Haar algorithm. The lbpSelected method loads a trained set of face recognition algorithms based on the LBP algorithm. These methods are the main methods of this program. The main class for the application FaceDetection.java is a descendant of the Application class creates and processes the main window with its resources (style, graphics). This window processes the video stream and looks for a person's face using Haar or LBP algorithms. When a person is found, it is framed. It consists of 2 methods: start and main. The start method creates the main panel of the program, on which there are buttons for starting the camera, choosing Haar or LBP algorithms. There is also a panel on which the image received from the camera is located. This camera transmits and updates the image in real time. Also, this method sets the dimensions of the panels, the colors of the program. The main method starts the program using the start method. After the methods and classes are defined, they can be used for training and forecasting [13-16]. Figures [2-4] show code fragments that implement classes in Java.



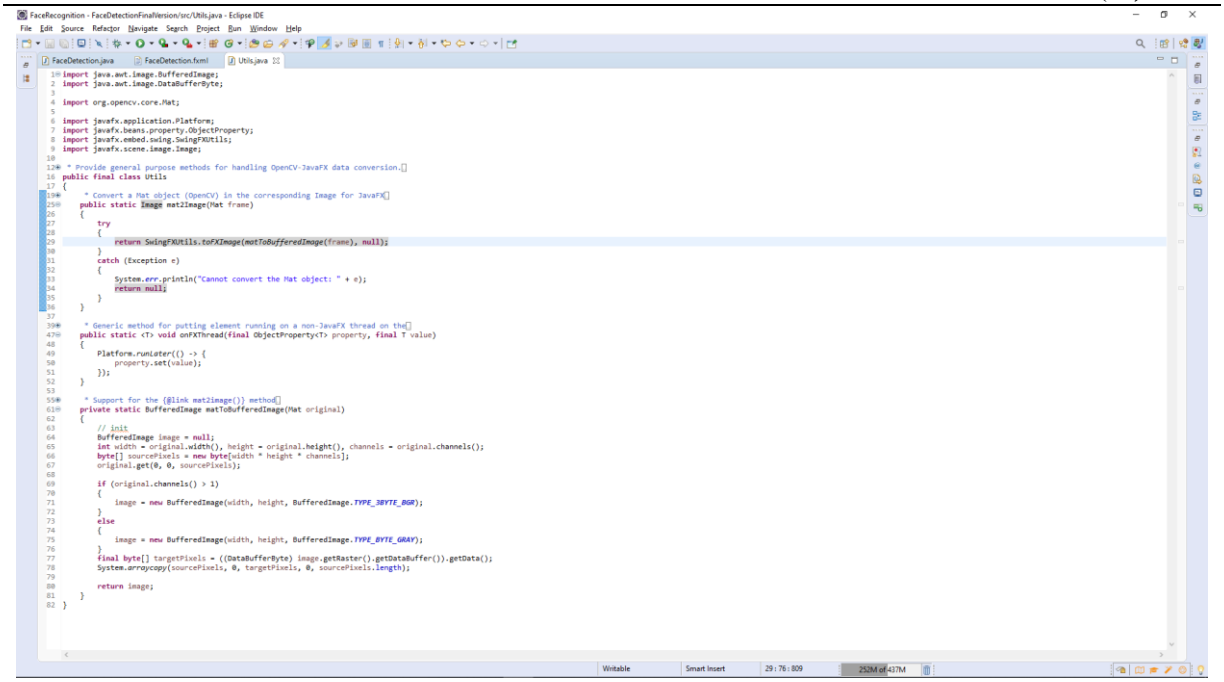
```
import org.opencv.core.Core;

import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.fxml.FXMLLoader;

/**
 * The main class for a JavaFX application. It creates and handle the main
 */
public class FaceDetection extends Application
{
    @Override
    public void start(Stage primaryStage)
    {
        try
        {
            // load the FXML resource
            FXMLLoader loader = new FXMLLoader(getClass().getResource("FaceDetection.fxml"));
            BorderPane root = (BorderPane) loader.load();
            // set a whitesmoke background
            root.setStyle("-fx-background-color: whitesmoke;");
            // create and style a scene
            Scene scene = new Scene(root, 800, 600);
            scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
            // create the stage with the given title and the previously created
            // scene
            primaryStage.setTitle("Face Detection and Tracking");
            primaryStage.setScene(scene);
            // show the GUI
            primaryStage.show();

            // init the controller
            FaceDetectionController controller = loader.getController();
            controller.init();

            // set the proper behavior on closing the application
            primaryStage.setOnCloseRequest((new EventHandler<WindowEvent>() {
                public void handle(WindowEvent we)
                {
                    controller.setClosed();
                }
            }));
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    public static void main(String[] args)
    {
        // load the native OpenCV library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        launch(args);
    }
}
```

Figure 2. The FaceDetection.java class.



```
import org.opencv.core.Core;

import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.fxml.FXMLLoader;

/**
 * The main class for a JavaFX application. It creates and handle the main
 */
public class FaceDetection extends Application
{
    @Override
    public void start(Stage primaryStage)
    {
        try
        {
            // load the FXML resource
            FXMLLoader loader = new FXMLLoader(getClass().getResource("FaceDetection.fxml"));
            BorderPane root = (BorderPane) loader.load();
            // set a whitesmoke background
            root.setStyle("-fx-background-color: whitesmoke;");
            // create and style a scene
            Scene scene = new Scene(root, 800, 600);
            scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
            // create the stage with the given title and the previously created
            // scene
            primaryStage.setTitle("Face Detection and Tracking");
            primaryStage.setScene(scene);
            // show the GUI
            primaryStage.show();

            // init the controller
            FaceDetectionController controller = loader.getController();
            controller.init();

            // set the proper behavior on closing the application
            primaryStage.setOnCloseRequest((new EventHandler<WindowEvent>() {
                public void handle(WindowEvent we)
                {
                    controller.setClosed();
                }
            }));
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    public static void main(String[] args)
    {
        // load the native OpenCV library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        launch(args);
    }
}
```

Figure 3. The FaceDetectionController.java class.



```
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;

import org.opencv.core.Mat;

import javafx.application.Platform;
import javafx.beans.property.ObjectProperty;
import javafx.embed.swing.SwingFXUtils;
import javafx.scene.image.Image;

/**
 * Provide general purpose methods for handling OpenCV-JavaFX data conversion.
 */
public final class Utils
{
    /**
     * Convert a Mat object (OpenCV) in the corresponding Image for JavaFX
     */
    public static Image mat2Image(Mat frame)
    {
        try
        {
            return SwingFXUtils.toFXImage(matToBufferedImage(frame), null);
        }
        catch (Exception e)
        {
            System.err.println("Cannot convert the Mat object: " + e);
            return null;
        }
    }

    /**
     * Generic method for putting element running on a non-JavaFX thread on the
     */
    public static <T> void onFXThread(final ObjectProperty<T> property, final T value)
    {
        Platform.runLater(() -> {
            property.set(value);
        });
    }

    /**
     * Support for the {@link mat2image()} method
     */
    private static BufferedImage matToBufferedImage(Mat original)
    {
        // init
        BufferedImage image = null;
        int width = original.width(), height = original.height(), channels = original.channels();
        byte[] sourcePixels = new byte[width * height * channels];
        original.get(0, 0, sourcePixels);

        if (original.channels() > 1)
        {
            image = new BufferedImage(width, height, BufferedImage.TYPE_3BYTE_BGR);
        }
        else
        {
            image = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);
        }
        final byte[] targetPixels = ((DataBufferByte) image.getRaster().getDataBuffer()).getData();
        System.arraycopy(sourcePixels, 0, targetPixels, 0, sourcePixels.length);

        return image;
    }
}
```

Figure 4. The Utils.java class.

Figure 5 shows the operation of the application and finding faces using the LBP algorithm.

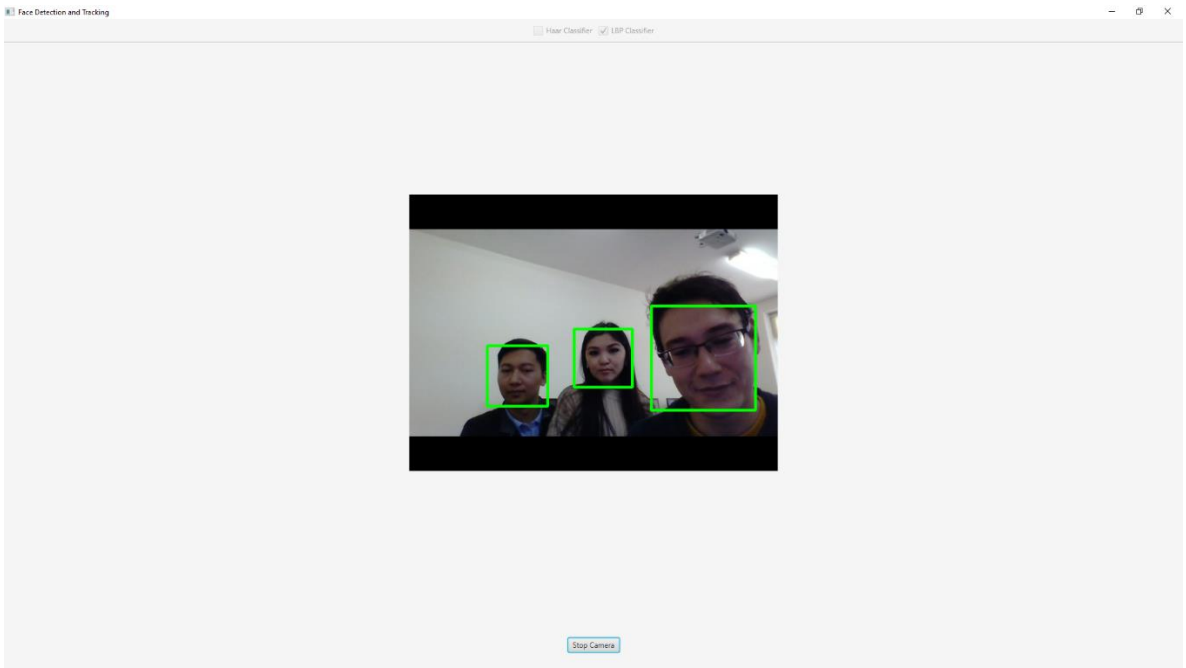


Figure 5. Face recognition using the LBP algorithm.

Figure 6 shows the operation of the application and finding faces using the Haar algorithm.

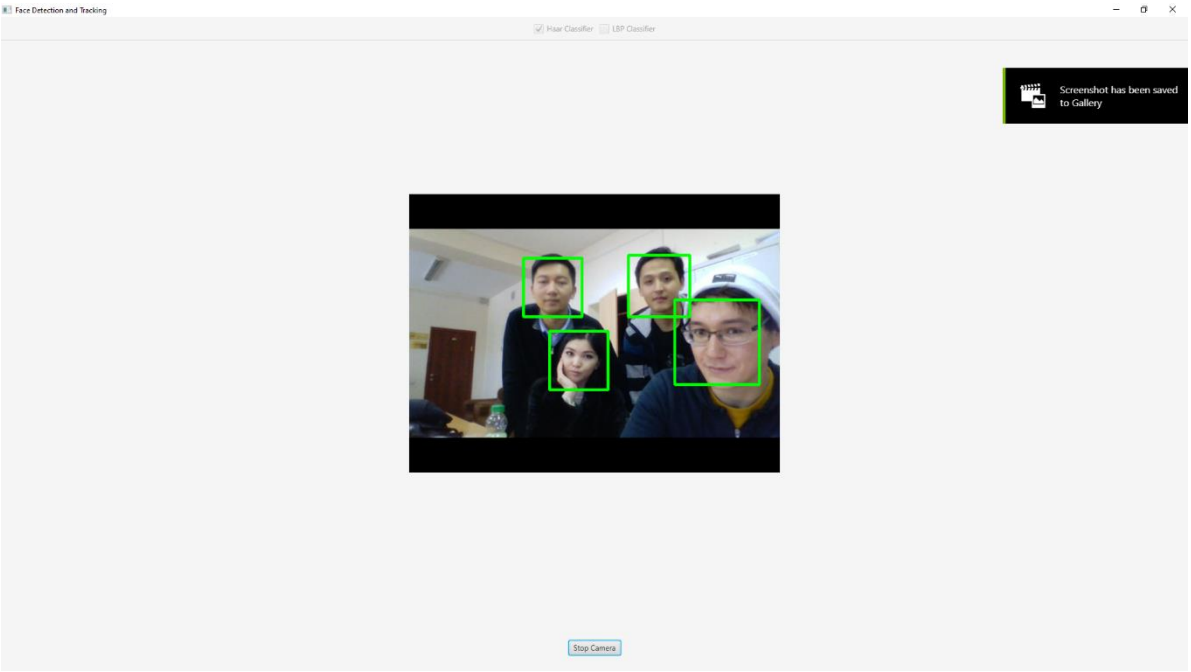


Figure 6. Face recognition using the Haar algorithm.

3. Conclusion

The developed application can be used to identify and locate people by scanning files from video cameras that are located in public places. Since this procedure will be completely automatic, it can help to find certain people, and will also save time for government agencies.

Also, the application can be integrated into the operating system of drones and used in search and rescue operations, in agriculture, and other fields.

References

1. talk at Creative Think seminar, 20 July 1982, URL:
https://www.folklore.org/StoryView.py?project=Macintosh&story=Creative_Think.txt
2. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weekly supervised place recognition. In: CVPR (2016) 12
3. Avrithis, Y., Kalantidis, Y., Tolias, G., Spyrou, E.: Retrieving Landmark and Non-Landmark Images from Community Photo Collections. In: ACM Multimedia. pp. 153–162 (2010)
4. Azizpour, H., Razavian, A.S., Sullivan, J., Maki, A., Carlsson, S.: From Generic to Specific Deep Representations for Visual Recognition. In: CVPR DeepVision Workshop (2015) 12
5. Baatz, G., Koeser, K., Chen, D., Grzeszczuk, R., Pollefeys, M.: Handling urban location recognition as a 2D homothetic problem. In: ECCV (2010)
6. Babenko, A., Lempitsky, V.: Aggregating Local Deep Features for Image Retrieval. In: ICCV (2015)
7. Savitch, W.: Absolute Java. Pearson (2016)
8. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: ECCV (2014)
9. Bergamo, A., Sinha, S.N., Torresani, L.: Leveraging Structure from Motion to Learn Discriminative Codebooks for Scalable Landmark Classification. In: CVPR. pp. 763–770 (2013)
10. 1. Fundamentals of Computer Vision, Mubarak Shah, Computer Science Department, University of Central Florida, Orlando, FL 32816, December 7, 2010.
11. The OpenCV 2.4.3 documentation, URL: http://docs.opencv.org
12. Mastering OpenCV with Practical Computer Vision Projects, Published by Packt Publishing Ltd. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK. ISBN 978-1-84951-782-9.
13. Boranbayev, A., Boranbayev, S., Nurusheva, A. Analyzing methods of recognition, classification and development of a software system. Advances in Intelligent Systems and Computing. –2018, Vol. 869, pp. 690-702.
14. Boranbayev, A., Shuitenov, G., Boranbayev, S. The method of data analysis from social networks using apache Hadoop. Advances in Intelligent Systems and Computing. –2018, Vol. 558, pp. 281-288.
15. Boranbayev, S., Nurkas, A., Tulebayev, Y., Tashtai, B. Method of Processing Big Data. Advances in Intelligent Systems and Computing. –2018, Vol. 738, pp. 757-758.
16. Boranbayev A.S., Boranbayev S.N., Khassanova A.A. Comparative analysis of methods of face detection and classification of images // Bulletin of L.N. Gumilyov Eurasian National University №2, 2017. - P.71-89.