(MASS) Motion Activated Surveillance System

Here is a list of used classed and links to the documentation if you are looking forward to implementing your own javafx+opencv project.

- BufferedImage See here
- DataBufferByte See here
- ClassArrayList See here
- List See here
- Executors See here
- ScheduledExecutorService See here
- Time Unit See her
- SwingFXUtils See here
- JavaFXevent See here
- FXML See here
- Image See here
- ImageView See here

org.opencv.core Class Core

https://docs.opencv.org/java/2.4.9/org/opencv/core/Core.html

org.opencv.core Class Mat

https://docs.opencv.org/java/2.4.2/org/opencv/core/Mat.html

org.opencv.core Class MatOfPoint

https://docs.opencv.org/java/2.4.7/org/opencv/core/MatOfPoint.html

org.opencv.core Class Point

https://docs.opencv.org/java/2.4.2/org/opencv/core/Point.html

org.opencv.core Class Scalar

https://docs.opencv.org/java/2.4.2/org/opencv/core/Scalar.html

org.opencv.core Class Size

https://docs.opencv.org/java/2.4.2/org/opencv/core/Size.html

Class Imgproc

https://docs.opencv.org/java/2.4.9/org/opencv/imgproc/Imgproc.html

Class VideoCapture

https://docs.opencv.org/java/3.0.0/org/opencv/videoio/VideoCapture.html

Class Videoio

https://docs.opencv.org/java/3.0.0/org/opencv/videoio/Videoio.html

```
package javafxapplicationopencv;

import com.jfoenix.controls.JFXButton;
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import javafx.embed.swing.SwingFXUtils;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;

import javafx.scene.image.ImageView;
```

```
import javafx.stage.Stage;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Point;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.videoio.VideoCapture;
import org.opencv.videoio.VideoWriter;
import static org.opencv.videoio.VideoWriter.fourcc;
import org.opencv.videoio.Videoio;
import sun.audio.*;

public class FXMLDocumentController {

    @FXML
    private JFXButton startbutton;

    @FXML
    private ImageView imagevw;

    @FXML
    private JFXButton stopbutton;

    @FXML
    private JFXButton settings;

    @FXML
    private JFXButton videos;

    @FXML
    private JFXButton clock;

    private ScheduledExecutorService timer;

    VideoWriter videoWriter;
    //opencv declarations
    VideoCapture capture;
    Mat webcamMatImage = new Mat();
    Image i1;


    //program variables
    boolean cameraActive;

    public void alarm() throws Exception {
        InputStream in = new FileInputStream("./resources/A.wav");
        // create an audiostream from the inputstream
        AudioStream audioStream;

        audioStream = new AudioStream(in);
```

```
        // play the audio clip with the audioplayer class
        AudioPlayer.player.start(audioStream);

    }




    public void initialize() {
        this.capture = new VideoCapture();
        this.cameraActive = false;


    }

    private void preprocess() {

        Mat frame = new Mat(480,640,CvType.CV_8UC3,Scalar.all(127));
        Mat firstFrame = new Mat(480,640,CvType.CV_8UC3,Scalar.all(127));
        Mat gray = new Mat(480,640,CvType.CV_8UC3,Scalar.all(127));
        Mat frameDelta = new Mat(480,640,CvType.CV_8UC3,Scalar.all(127));
        Mat thresh = new Mat(480,640,CvType.CV_8UC3,Scalar.all(127));
        Mat toVideo=new Mat(480,640,CvType.CV_8UC3,Scalar.all(127));
        List<MatOfPoint> cnts = new ArrayList<>();

        if (!this.cameraActive) {
            // start the video capture
            this.capture.open(0);

            // is the video stream available?
            if (this.capture.isOpened()) {
                this.cameraActive = true;



                //Size frameSize = new Size((int) capture.get(Videoio.CAP_PROP_FRAME_WIDTH),
(int) capture.get(Videoio.CAP_PROP_FRAME_HEIGHT));


                videoWriter = new VideoWriter("test.avi", VideoWriter.fourcc('M',
'J','P','G'),20, frame.size(), true);
                if(!videoWriter.isOpened()){
                    System.out.println("Cannot open videwriter");
                }


                // grab a frame every 33 ms (30 frames/sec)
                Runnable frameGrabber = () -> {
                    int j = 0;

                    //convert to grayscale and set the first frame

                    while (true) {
```

```
                        capture.read(frame);
                        Imgproc.cvtColor(frame, firstFrame, Imgproc.COLOR_BGR2GRAY);
                        Imgproc.GaussianBlur(firstFrame, firstFrame, new Size(21, 21), 0);

                        capture.read(frame);

                        //convert to grayscale
                        Imgproc.cvtColor(frame, gray, Imgproc.COLOR_BGR2GRAY);
                        Imgproc.GaussianBlur(gray, gray, new Size(21, 21), 0);

                        //compute difference between first frame and current frame
                        Core.absdiff(firstFrame, gray, frameDelta);
                        Imgproc.threshold(frameDelta, thresh, 25, 255, Imgproc.THRESH_BINARY);

                        Imgproc.dilate(thresh, thresh, new Mat(), new Point(-1, -1), 2);

                        if (false) {

                            if (!thresh.empty()) {
                                i1 = mat2Image(thresh);
                                imagevw.setImage(i1);

                            }

                        }

                        Imgproc.findContours(thresh, cnts, new Mat(), Imgproc.RETR_EXTERNAL,
Imgproc.CHAIN_APPROX_SIMPLE);

                        for (int i = 0; i < cnts.size(); i++) {
                            if (Imgproc.contourArea(cnts.get(i)) > 500) {

                                Imgproc.drawContours(frame, cnts, i, new Scalar(0, 255, 0), 2);

                                videoWriter.write(frame);
                                System.out.println("Motion detected:" + j);

                                try {
                                    this.alarm();
                                } catch (Exception e) {
                                    System.out.println(e);
                                }
                                j++;

                                if (!frame.empty()) {
                                    i1 = mat2Image(frame);
                                    imagevw.setImage(i1);
                                    //Imgcodecs.imwrite("pic.jph",frame);
                                }

                            }
```

```
                        }

                        cnts.clear();

                    }

                };

                this.timer = Executors.newSingleThreadScheduledExecutor();
                this.timer.scheduleAtFixedRate(frameGrabber, 0, 20, TimeUnit.MILLISECONDS);

            } else {

                System.err.println("Impossible to open the camera connection...");
            }
        } else {
            // the camera is not active at this point
            this.cameraActive = false;
            this.stopAcquisition();
        }

    }

    // converting an opencv mat object to an image
    private static Image mat2Image(Mat frame) {
        try {
            return SwingFXUtils.toFXImage(matToBufferedImage(frame), null);
        } catch (Exception e) {
            System.err.println("Cannot convert the Mat obejct: " + e);
            return null;
        }
    }

    //called by opencv mat2Image for mat to bufferedImage
    private static BufferedImage matToBufferedImage(Mat original) {

        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;
```

```
    }

    private void stopAcquisition() {

        if (this.timer != null && !this.timer.isShutdown()) {
            try {
                // stop the timer
                this.timer.shutdown();
                this.timer.awaitTermination(33, TimeUnit.MILLISECONDS);
            } catch (Exception e) {
                // log any exception
                System.err.println("Exception in stopping the frame capture, trying to release
the camera now... " + e);
            }
        }
        if (this.capture.isOpened()) {
            // release the camera
            this.capture.release();

        }
    }

    private void start() {

        Image i1 = new Image("content/aa.jpg");

        imagevw.setImage(i1);

    }

    @FXML

    void startButton(ActionEvent event) {

        //  this.start();
        this.initialize();
        // this.motion();
        this.preprocess();

    }

    @FXML
    void stopButton(ActionEvent event) {

        this.stopAcquisition();

    }

    @FXML
    void fa1111(ActionEvent event) {

    }
```

```
    @FXML
    void db2f21(ActionEvent event) {

    }


    @FXML
    void onSettings(ActionEvent event) {

     try {
        FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("videosettings.fxml"));
                Parent root1 = (Parent) fxmlLoader.load();
                Stage stage = new Stage();
                stage.setScene(new Scene(root1));
                stage.setTitle("settings");
                stage.show();
        } catch(Exception e) {
           e.printStackTrace();
      }


    }



    @FXML
    void onVideos(ActionEvent event) {



    }



}




// new file

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package javafxapplicationopencv;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;

import javafx.stage.Stage;
```

```
import org.opencv.core.*;
/**
 *
 * @author Vivek
 */
public class JavaFXApplicationOpencv extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));

        Scene scene = new Scene(root);

        stage.setScene(scene);
        stage.setTitle("MASS");

        stage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
       System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        launch(args);
    }

}


// new setting fxml

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import com.jfoenix.controls.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="javafxapplicationopencv.VideosettingsController">
   <children>
      <Pane layoutX="14.0" layoutY="35.0" prefHeight="89.0" prefWidth="577.0">
         <children>
            <JFXSlider id="sliderbar" blockIncrement="1000.0" layoutX="7.0" layoutY="42.0"
max="5000.0" min="50.0" prefHeight="14.0" prefWidth="554.0" />
            <Text layoutX="7.0" layoutY="19.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="Motion Sensitivity (Threshold)" wrappingWidth="262.13671875">

               <font>
```

```
                  <Font name="Calibri" size="14.0" />
               </font>
            </Text>
         </children>
      </Pane>
      <Pane layoutX="14.0" layoutY="124.0" prefHeight="253.0" prefWidth="561.0">
         <children>
            <Text layoutX="21.0" layoutY="36.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="FPS:" />
            <Text layoutX="296.0" layoutY="36.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="Video Capture: with camera no:" />
            <JFXRadioButton layoutX="21.0" layoutY="88.0" prefHeight="17.0" prefWidth="154.0"
text="Original" />
            <JFXRadioButton layoutX="21.0" layoutY="118.0" prefHeight="17.0" prefWidth="154.0"
text="Difference" />
            <JFXRadioButton layoutX="21.0" layoutY="148.0" prefHeight="17.0" prefWidth="154.0"
text="Threshold" />
            <JFXRadioButton layoutX="22.0" layoutY="180.0" prefHeight="17.0" prefWidth="154.0"
text="GrayScale" />
            <JFXTextField layoutX="56.0" layoutY="19.0" prefHeight="25.0" prefWidth="61.0" />
            <JFXTextField layoutX="487.0" layoutY="12.0" prefHeight="25.0" prefWidth="61.0" />
            <JFXRadioButton layoutX="21.0" layoutY="209.0" prefHeight="17.0" prefWidth="154.0"
text="Contours" />
         </children>
      </Pane>
   </children>
</AnchorPane>


//new file
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package javafxapplicationopencv;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.Initializable;

/**
 * FXML Controller class
 *
 * @author Vivek
 */
public class VideosettingsController implements Initializable {

    /**
     * Initializes the controller class.
     */
    @Override

    public void initialize(URL url, ResourceBundle rb) {
```

```
        // TODO
    }

}
```