File - AlarmCameraView.java

```java
package com.example.alarm.alarmapp.views;

import android.content.Context;
import android.hardware.Camera;
import android.os.Handler;
import android.util.AttributeSet;
import android.util.Log;

import com.example.alarm.alarmapp.R;

import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.JavaCameraView;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.imgproc.Imgproc;

import java.util.ArrayList;
import java.util.Date;

/**
 * A camera view that has a movement detection.
 */
public class AlarmCameraView extends JavaCameraView implements
CameraBridgeViewBase.CvCameraViewListener2 {
    private static final String TAG = AlarmCameraView.class.getName();

    private int mTimeToCalibrate = 10000;
    private double mAlarmThreshold = 6d;
    private int mProcessFps = 4;
    private double mAverageOver = 100d / mProcessFps;

    private long mCalibratingStartedAt = 0;

    private boolean mAreCameraParamsSet = false;
    private State mState = State.IDLE;
    private IAlarmCameraListener mAlarmListener = null;
    private Mat mLastMat = null;
    private Mat mEmptyGrayMap = null;

    private double mMovingAbsDiffAvg = -1d;
    private double mMovingDiffAvg = -1d;
    private double mMaxDiff = 0;

    public AlarmCameraView(Context context, int cameraId) {
        super(context, cameraId);
        this.setCvCameraViewListener(this);
    }

    public AlarmCameraView(Context context, AttributeSet attrs) {
        super(context, attrs);
        this.setCvCameraViewListener(this);
    }

```

File - AlarmCameraView.java

```
 54     public interface IAlarmCameraListener {
 55         void onAlarm();
 56         void onCalibrating();
 57         void onRun();
 58         Handler getHandler();
 59     }
 60
 61     private void onAlarmInternal() {
 62         Log.d(TAG, "onAlarmInternal");
 63         if (mAlarmListener != null) mAlarmListener.getHandler().post(
    () -> mAlarmListener.onAlarm());
 64     }
 65
 66     private void onCalibratingInternal() {
 67         Log.d(TAG, "onCalibratingInternal");
 68         if (mAlarmListener != null) mAlarmListener.getHandler().post(
    () -> mAlarmListener.onCalibrating());
 69     }
 70
 71     private void onRunInternal() {
 72         Log.d(TAG, "onRunInternal");
 73         if (mAlarmListener != null) mAlarmListener.getHandler().post(
    () -> mAlarmListener.onRun());
 74     }
 75
 76     public enum State {
 77         IDLE, CALIBRATING, RUNNING
 78     }
 79
 80     public State getCurrState() {
 81         return mState;
 82     }
 83
 84     public void startAlarm() {
 85         mCalibratingStartedAt = System.currentTimeMillis();
 86         mState = State.CALIBRATING;
 87         Log.d(TAG, "State: " + mState);
 88         onCalibratingInternal();
 89     }
 90
 91     public void stopAlarm() {
 92         mCalibratingStartedAt = 0;
 93         mMovingAbsDiffAvg = -1;
 94         mMaxDiff = 0;
 95
 96         mState = State.IDLE;
 97     }
 98
 99     public void setAlarmListener(IAlarmCameraListener alarmListener)
    {
100         this.mAlarmListener = alarmListener;
101     }
102
103     public void removeAlarmListener() {
```

File - AlarmCameraView.java

```
104         this.mAlarmListener = null;
105     }
106 
107     @Override
108     public void onPreviewFrame(byte[] frame, Camera arg1) {
109         if (!mAreCameraParamsSet) {
110             mAreCameraParamsSet = true;
111             Camera.Parameters params = mCamera.getParameters();
112             params.setFocusMode(Camera.Parameters.FOCUS_MODE_FIXED);
113             mCamera.setParameters(params);
114         }
115         super.onPreviewFrame(frame, arg1);
116     }
117 
118     @Override
119     public void onCameraViewStarted(int width, int height) {
120         mEmptyGrayMap = new Mat(height, width, CvType.CV_8UC1);
121     }
122 
123     @Override
124     public void onCameraViewStopped() {
125         if (mEmptyGrayMap != null) mEmptyGrayMap.release();
126         mEmptyGrayMap = null;
127         if (mLastMat != null) mLastMat.release();
128         mLastMat = null;
129     }
130 
131     @Override
132     public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
133         Mat screenMat = inputFrame.rgba();
134 
135         if (mState == State.RUNNING || mState == State.CALIBRATING) {
136             if (mState == State.CALIBRATING) {
137 
138                 if (System.currentTimeMillis() -
    mCalibratingStartedAt >= mTimeToCalibrate) {
139                     mState = State.RUNNING;
140                     onRunInternal();
141                     Log.d(TAG, "State: " + mState);
142                 }
143             }
144 
145             if (mLastMat != null) {
146                 //calculate trigger values
147                 Mat grayMat = inputFrame.gray();
148                 Mat diff = new Mat(grayMat.size(), grayMat.type());
149                 Core.absdiff(grayMat, mLastMat, diff);
150                 final double diffD = Core.mean(diff).val[0];
151                 if (mMovingAbsDiffAvg == -1d) mMovingAbsDiffAvg =
    diffD;
152                 else {
153                     mMovingAbsDiffAvg = (mMovingAbsDiffAvg * (
    mAverageOver - 1) + diffD) / mAverageOver;
154                 }
```

```
155                 if (mMovingDiffAvg == -1d) mMovingDiffAvg = 0;
156                 else {
157                     mMovingDiffAvg = (mMovingDiffAvg * (mAverageOver
    - 1) + Math.abs(diffD - mMovingAbsDiffAvg)) / mAverageOver;
158                 }
159 
160                 if (mMaxDiff < diffD) {
161                     mMaxDiff = diffD;
162                 }
163                 double absCurrAlarmThreshold = (mMovingDiffAvg *
    mAlarmThreshold) + mMovingAbsDiffAvg;
164                 boolean alarmTriggered = diffD >
    absCurrAlarmThreshold;
165                 if (alarmTriggered) Log.d(TAG, "Alarm Triggered: " +
    new Date().toGMTString());
166 
167                 Log.v(TAG,
168                     String.format("onProcessedFrame:\t%s\t%s\t%s\t%s\
    t%s",
169                         String.format(getContext().getString(R.string
    .curr_alarm_threshold_val), absCurrAlarmThreshold),
170                         String.format(getContext().getString(R.string
    .moving_diff_abs_avg_val), mMovingAbsDiffAvg),
171                         String.format(getContext().getString(R.string
    .moving_diff_avg_val), mMovingDiffAvg),
172                         String.format(getContext().getString(R.string
    .max_diff_val), mMaxDiff),
173                         String.format(getContext().getString(R.string
    .AbsDiff_val), diffD)
174                     )
175                 );
176                 if (alarmTriggered && mState == State.RUNNING)
    onAlarmInternal();
177 
178                 //draw changes on screenMat
179                 ArrayList<Mat> mergeMats = new ArrayList<>(3);
180                 mergeMats.add(diff);
181                 mergeMats.add(mEmptyGrayMap);
182                 mergeMats.add(mEmptyGrayMap);
183 
184                 //create buffer for red diff (rgb mat)
185                 Mat matRedDiff = new Mat(diff.size(), screenMat.type(
    ));
186                 //merge two empty single channel mats and the diff
    mat as different channels into rgb mat
187                 Core.merge(mergeMats, matRedDiff);
188                 //create buffer for gray image in rgb mat
189                 Mat grayRgbMat = new Mat(screenMat.size(), screenMat.
    type());
190                 //convert gray single channel mat to rgb mat
191                 Imgproc.cvtColor(grayMat, grayRgbMat, Imgproc.
    COLOR_GRAY2RGB);
192                 //add the gray and red overlay together
193                 Core.add(grayRgbMat, matRedDiff, screenMat);
```

```
194
195                 //release resources
196                 grayRgbMat.release();
197                 grayMat.release();
198                 matRedDiff.release();
199                 diff.release();
200                 mLastMat.release();
201             }
202             mLastMat = inputFrame.gray();
203         }
204         return screenMat;
205     }
206 }
207
```