package org.bytedeco.javacv.android.recognize.technowings;

import com.constant.ServerConstants;

import static org.bytedeco.javacpp.opencv\_imgproc.putText;

import java.io.File;

import java.io.IOException;

import java.util.Date;

import java.util.logging.Level;

import java.util.logging.Logger;

import static org.bytedeco.javacpp.opencv\_core.\*;

import org.bytedeco.javacpp.opencv\_core;

import org.bytedeco.javacpp.opencv\_core.Mat;

import org.bytedeco.javacv.FFmpegFrameRecorder;

import org.bytedeco.javacv.FrameRecorder.Exception;

import util.opencv.OpenCVHelper;

public class VideoWriter {

FFmpegFrameRecorder recorder = null;

public static long START = System.currentTimeMillis();

public void stopVideoIfAny() {

if (recorder != null) {

try {

recorder.stop();

recorder.release();

recorder = null;

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

public void writeFrame(Mat mat) {

long end = System.currentTimeMillis();

try {

if (recorder == null) {

// showInfo("Creating New Video");

CommonFunctions.log(" Creating VideoWrite video is null ");

createNewVideoWriter();

} else {

if ((end - START) >= ServerConstants.VIDEO\_DURATION\_IN\_MIN) {

CommonFunctions.log(" Creating New VideoWrite video " + end);

// showInfo("Creating New Video");

START = System.currentTimeMillis();

recorder.stop();

recorder.release();

recorder = null;

// videoWriter.close();

// videoWriter.release();

createNewVideoWriter();

}

}

// Mat m = (Mat) mats[0];

// Mat dst = new Mat(IMAGE\_WIDTH, IMAGE\_HEIGHT, m.type());

// resize(m, dst, new opencv\_core.Size(imageWidth, imageHeight));

// Log.i(LOG\_TAG, "dst Type "+m.type()+" "+m.channels()); //24 4

// cvtColor(dst,dst,COLOR\_BGRA2RGBA);

// videoWriter.write(m);

long videoTS = 1000 \* (System.currentTimeMillis() - START);

// Check for AV drift

if (videoTS > recorder.getTimestamp()) {

System.out.println(

"Lip-flap correction: "

+ videoTS + " : "

+ recorder.getTimestamp() + " -> "

+ (videoTS - recorder.getTimestamp()));

// We tell the recorder to write this frame at this timestamp

recorder.setTimestamp(videoTS);

}

String todaysDate = ServerConstants.videoDisplayDateFormat.format(new Date());

// showInfo(todaysDate);

// Send the frame to the

// org.bytedeco.javacv.FFmpegFrameRecorder putText

// (rgbaMat

// , todaysDate

// , new opencv\_core.Point(50, 20)

// ,

// FONT\_HERSHEY\_PLAIN

// , 1, new opencv\_core.Scalar(0, 255, 0, 0)

// );

putText(mat, todaysDate, new opencv\_core.Point(20, 50),

FONT\_HERSHEY\_DUPLEX, 1.3, new opencv\_core.Scalar(0, 0, 255, 0));

// putText(mat, "In Count- X Out Count - Y", new opencv\_core.Point(20, 100), FONT\_HERSHEY\_DUPLEX, 1.3, new opencv\_core.Scalar(0,

// 0, 255, 0));

recorder.record(OpenCVHelper.mat2frame(mat));

// dst.release();

// m.release();

} catch (org.bytedeco.javacv.FrameRecorder.Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public File generateFile() {

String todaysDate = ServerConstants.dt.format(new Date());

String fileName = "Video\_" + todaysDate + "\_.mp4";

// File dir = new

// File(Environment.getExternalStorageDirectory().toString() +

// "/Database/");

File dir = new File("./stored/changed/");

if (!dir.exists()) {

dir.mkdirs();

}

File file = new File("./stored/changed/" + fileName);

return file;

}

public void createNewVideoWriter() {

File file = generateFile();

recorder = new FFmpegFrameRecorder(file, ServerConstants.IMAGE\_WIDTH, ServerConstants.IMAGE\_HEIGHT, 0);

recorder.setFormat("mp4");

recorder.setFrameRate(ServerConstants.FRAME\_RATE);

try {

CommonFunctions.log("Creating New Video " + file.getName());

recorder.start();

CommonFunctions.log(" Creating Video " + file.getCanonicalPath() + "-----------------------------");

} catch (IOException ex) {

Logger.getLogger(VideoWriter.class.getName()).log(Level.SEVERE, null, ex);

}

}

}