基类：DetectionBasedTracker.java

```
package org.opencv.samples.facedetect;
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;

public class DetectionBasedTracker
{
    //DetectionBasedTracker 构造函数
    public DetectionBasedTracker(String cascadeName, int minFaceSize) {
        mNativeObj = nativeCreateObject(cascadeName, minFaceSize);
    }

    public void start() {
        nativeStart(mNativeObj);
    }

    public void stop() {
        nativeStop(mNativeObj);
    }

    public void setMinFaceSize(int size) {
        nativeSetFaceSize(mNativeObj, size);
    }

    public void detect(Mat imageGray, MatOfRect faces) {
        nativeDetect(mNativeObj, imageGray.getNativeObjAddr(),
faces.getNativeObjAddr());
    }

    public void release() {
        nativeDestroyObject(mNativeObj);
        mNativeObj = 0;
    }
    public int judge(Mat imageGray,int type){
        return nativeJudge(mNativeObj,imageGray.getNativeObjAddr(),type);
        }

    public int judge_m(Mat imageGray){
        return nativeJudgem(mNativeObj,imageGray.getNativeObjAddr());
    }
    public int judge_p(Mat imageGray){
        return nativeJudgep(mNativeObj,imageGray.getNativeObjAddr());
    }
```

```
        private long mNativeObj = 0;

        private static native long nativeCreateObject(String cascadeName, int
minFaceSize);
        //native 函数
        private static native void nativeDestroyObject(long thiz);
        private static native void nativeStart(long thiz);
        private static native void nativeStop(long thiz);
        private static native void nativeSetFaceSize(long thiz, int size);
        private static native void nativeDetect(long thiz, long inputImage, long faces);
        private static native int nativeJudge(long thiz,long inputImage,int type);
        private static native int nativeJudgem(long thiz,long inputImage);
        private static native int nativeJudgep(long thiz,long inputImage);
}

//图像处理类 FdActivity
package org.opencv.samples.facedetect;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.lang.Math.*;
import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;
import org.opencv.objdetect.CascadeClassifier;
import org.opencv.objdetect.Objdetect;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.WindowManager;

public class FdActivity extends Activity implements CvCameraViewListener2 {

    private static final String    TAG                 =
"OCVSample::Activity";
    private static final Scalar    FACE_RECT_COLOR     = new Scalar(0, 255,
0, 255);
    public static final int        JAVA_DETECTOR       = 0;
    public static final int        NATIVE_DETECTOR     = 1;
    private MenuItem               mItemFace50;
    private MenuItem               mItemFace40;
    private MenuItem               mItemFace30;
    private MenuItem               mItemFace20;
    private MenuItem               mItemFace10;
    private MenuItem               mItemType;

    private Mat                    mRgba;
    private Mat                    mGray;
    private Mat                    teplateR;
    private Mat                    teplateL;
    private Mat                    teplateM;
    private Mat                    mZoomWindow;
    private Mat                    mZoomWindow2;
    private Mat                    mZoomWindow1;
    private File                   mCascadeFile;

    private CascadeClassifier      mJavaDetector;
    private DetectionBasedTracker  mNativeDetector;
    private CascadeClassifier      mJavaDetectorEyeL;
    private CascadeClassifier      mJavaDetectorEyeR;
    private CascadeClassifier      mJavaDetectorMouth;
    private  int                   mDetectorType       =
JAVA_DETECTOR;
    private String[]               mDetectorName;
    private int status_R;
    private int status_L;
    private int status_M;
    private float                  mRelativeFaceSize   = 0.2f;
    private int                    mAbsoluteFaceSize   = 0;
    double xCenter = -1;
    double yCenter = -1;
```

```
    private CameraBridgeViewBase      mOpenCvCameraView;
    private double eyel_x;
    private double eyel_y;
    private double eyer_x;
    private double eyer_y;
    private double distance1=0.0;
    private double distance2=0.0;
    private double width_m=0;
    private double height_m=0;
    private BaseLoaderCallback    mLoaderCallback = new BaseLoaderCallback(this)
{
        @Override
        public void onManagerConnected(int status) {
            switch (status) {
                case LoaderCallbackInterface.SUCCESS:
                {
                    Log.i(TAG, "OpenCV loaded successfully");

                    // Load native library after(!) OpenCV initialization
                    System.loadLibrary("detection_based_tracker");

                    try {
                        // load cascade file from application resources
                        InputStream is =
getResources().openRawResource(R.raw.lbpcascade_frontalface);
                        File cascadeDir = getDir("cascade",
Context.MODE_PRIVATE);
//加载脸部检测模板
                        mCascadeFile = new File(cascadeDir,
"lbpcascade_frontalface.xml");
                        FileOutputStream os = new
FileOutputStream(mCascadeFile);

                        byte[] buffer = new byte[4096];
                        int bytesRead;
                        while ((bytesRead = is.read(buffer)) != -1) {
                            os.write(buffer, 0, bytesRead);
                        }

                        is.close();
                        os.close();
                        /*
                         * 加载右眼分类器
                         */
```

```
                    InputStream iser = getResources().openRawResource(
                    R.raw.haarcascade_righteye_2splits);
                    File cascadeDirER = getDir("cascadeER",
                    Context.MODE_PRIVATE);
                    File cascadeFileER = new File(cascadeDirER,
                    "haarcascade_righteye_2splits.xml");
                    FileOutputStream oser = new
FileOutputStream(cascadeFileER);
                    /*
                    * 定义一个足够大的 byte 类型的数组，存放加载右
眼部分类器文件的数据
                    */
                    byte[] bufferER = new byte[4096];
                    int bytesReadER;
                    while ((bytesReadER = iser.read(bufferER)) != -1) {
                    oser.write(bufferER, 0, bytesReadER);
                    }
                    iser.close();
                    oser.close();
                    /*
                     * 加载左眼分类器
                     */
                     InputStream isel =
getResources().openRawResource(
                     R.raw.haarcascade_lefteye_2splits);
                     File cascadeDirEL = getDir("cascadeEL",
                     Context.MODE_PRIVATE);
                     File cascadeFileEL = new File(cascadeDirEL,
                     "haarcascade_lefteye_2splits.xml");
                     FileOutputStream osel = new
FileOutputStream(cascadeFileEL);
                     /*
                     * 定义一个足够大的 byte 类型的数组，存放加载
左眼部分类器文件的数据
                     */
                     byte[] bufferEL = new byte[4096];
                     int bytesReadEL;
                     while ((bytesReadEL = isel.read(bufferEL)) != -1) {
                     osel.write(bufferEL, 0, bytesReadEL);
                     }
                     isel.close();
                     osel.close();
                     /**加载脸部检测*/
                     InputStream                ism                =
```

```
getResources().openRawResource(R.raw.haarcascade_mcs_mouth);
                                File cascadeDirM = getDir("cascadeM",
                                Context.MODE_PRIVATE);
                                File cascadeFileM = new File(cascadeDirM,
                                "haarcascade_mcs_mouth.xml");
                                FileOutputStream osm = new
FileOutputStream(cascadeFileM);
/*
                                * 定义一个足够大的 byte 类型的数组，存放加载
脸部分类器文件的数据
                                */

                                byte[] bufferM = new byte[4096];
                                int bytesReadM;
                                while ((bytesReadM = ism.read(bufferM)) != -1) {
                                osm.write(bufferM, 0, bytesReadM);
                                }
                                ism.close();
                                osm.close();
                               mJavaDetector = new
CascadeClassifier(mCascadeFile.getAbsolutePath());
                               if (mJavaDetector.empty()) {
                                    Log.e(TAG, "Failed to load cascade classifier");
                                    mJavaDetector = null;
                               } else
                                    Log.i(TAG, "Loaded cascade classifier from " +
mCascadeFile.getAbsolutePath());
                               /**
                                * opencv 的迭代分类器可以持续迭代，将分类器迭
代为强分类器，这样就可以从集中一张脸到
                                集中到眼睛上面，可以实现对人眼的识别与检测；
这里是对右眼进行检测
                                */
                                mJavaDetectorEyeR = new CascadeClassifier(
                                cascadeFileER.getAbsolutePath());
                                if (mJavaDetectorEyeR.empty()) {
                                Log.e(TAG, "Failed to load cascade classifier");
                                mJavaDetectorEyeR = null;
                                } else
                                Log.i(TAG, "Loaded cascade classifier from "
                                + mCascadeFile.getAbsolutePath());
                                cascadeDirER.delete();
                                /**
                                * 这里是对左眼进行迭代与检测
```

```
                        */
                        mJavaDetectorEyeL = new CascadeClassifier(
                        cascadeFileEL.getAbsolutePath());
                        if (mJavaDetectorEyeL.empty()) {
                        Log.e(TAG, "Failed to load cascade classifier");
                        mJavaDetectorEyeL = null;
                        } else
                        Log.i(TAG, "Loaded cascade classifier from "
                        + mCascadeFile.getAbsolutePath());
                        cascadeDirEL.delete();
                        //mouth detector  嘴巴检测部分
                        mJavaDetectorMouth=new CascadeClassifier(
                                cascadeFileM.getAbsolutePath());
                        if (mJavaDetectorMouth.empty()) {
                            Log.e(TAG, "Failed to load cascade classifier");
                            mJavaDetectorMouth = null;
                            }    else
                                 Log.i(TAG, "Loaded cascade classifier from
"
                                                + mCascadeFile.getAbsolutePath());
                                                cascadeDirM.delete();
                                mNativeDetector                =                new
DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0);
                                cascadeDir.delete();
                            } catch (IOException e) {
                                e.printStackTrace();
                                Log.e(TAG, "Failed to load cascade. Exception thrown:
" + e);
                            }
//确定使用前后摄像头
                            mOpenCvCameraView.setCameraIndex(1);//TODO
                            /**
                            * enableFpsMeter()方法确定在屏幕上 fps 值的标签，其中
FPS：Frame Per Second (每秒
                            帧数)
                            */
                            mOpenCvCameraView.enableFpsMeter();
                            /**
                            * enableView()方法建立 Camera 连接
                            */
                            mOpenCvCameraView.enableView();
                        } break;
                        default:
                        {
```

```
                    super.onManagerConnected(status);
                } break;
            }
        }
    };

    public FdActivity() {
        mDetectorName = new String[2];
        mDetectorName[JAVA_DETECTOR] = "Java";
        mDetectorName[NATIVE_DETECTOR] = "Native (tracking)";
        Log.i(TAG, "Instantiated new " + this.getClass());
    }

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "called onCreate");
        super.onCreate(savedInstanceState);

getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)
;
        setContentView(R.layout.face_detect_surface_view);
        mOpenCvCameraView = (CameraBridgeViewBase)
findViewById(R.id.fd_activity_surface_view);
        mOpenCvCameraView.setCvCameraViewListener(this);
    }

    @Override
    public void onPause()
    {
        super.onPause();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
    }

    @Override
    public void onResume()
    {
        super.onResume();
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_3,
this, mLoaderCallback);
    }

    public void onDestroy() {
```

```
        super.onDestroy();
        mOpenCvCameraView.disableView();
    }

    public void onCameraViewStarted(int width, int height) {
        mGray = new Mat();
        mRgba = new Mat();
        CreateAuxiliaryMats();
    }

    public void onCameraViewStopped() {
        mGray.release();
        mRgba.release();
        mZoomWindow.release();
        mZoomWindow2.release();
        mZoomWindow1.release();
    }

    public Mat onCameraFrame(CvCameraViewFrame inputFrame) {

        mRgba = inputFrame.rgba();
        mGray = inputFrame.gray();

        if (mAbsoluteFaceSize == 0) {
            int height = mGray.rows();
            if (Math.round(height * mRelativeFaceSize) > 0) {
                mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
            }
            mNativeDetector.setMinFaceSize(mAbsoluteFaceSize);
        }
        if (mZoomWindow == null || mZoomWindow2 == null||mZoomWindow1==null)
            CreateAuxiliaryMats();
        MatOfRect faces = new MatOfRect();
        if (mDetectorType == JAVA_DETECTOR) {
            if (mJavaDetector != null){
                mJavaDetector.detectMultiScale(mGray, faces, 1.1, 2, 2, // TODO: objdetect.CV_HAAR_SCALE_IMAGE
                        new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());
                }
        }
        else if (mDetectorType == NATIVE_DETECTOR) {
            if (mNativeDetector != null)
```

```
                    mNativeDetector.detect(mGray, faces);
            }
            else {
                Log.e(TAG, "Detection method is not selected!");
            }
//这里只探测第一个 face，其余的 faces 舍掉
            Rect[] facesArray = faces.toArray();
            if( facesArray.length>0)
            {
            Core.rectangle(mRgba, facesArray[0].tl(), facesArray[0].br(),
            FACE_RECT_COLOR, 3);
            xCenter = (facesArray[0].x + facesArray[0].width + facesArray[0].x) / 2;
            yCenter = (facesArray[0].y + facesArray[0].y + facesArray[0].height) / 2;
            Point center = new Point(xCenter, yCenter);
            Core.circle(mRgba, center, 10, new Scalar(255, 0, 0, 255), 3);
            //Core.putText(mRgba, "[" + center.x + "," + center.y + "]",
           // new Point(center.x + 20, center.y + 20),
           // Core.FONT_HERSHEY_SIMPLEX, 0.7, new Scalar(255, 255, 255, 255));
            Rect r = facesArray[0];
            // compute the eye area
            // split it   分离眼部区域 从选定的脸部区域获取嘴部和眼部的矩形
            Rect eyearea_right = new Rect(r.x + r.width / 16,
            (int) (r.y + (r.height / 4.5)), (r.width - 2 * r.width / 16) / 2, (int) (r.height /
3.0));
            Rect eyearea_left = new Rect(r.x + r.width / 16+ (r.width - 2 * r.width / 16) /
2,
            (int) (r.y + (r.height / 4.5)),(r.width - 2 * r.width / 16) / 2, (int) (r.height /
3.0));
//分离嘴部区域 用于后面的嘴部识别函数
           // Rect mouth_area = new Rect(r.x + r.width / 2- r.width /5,(int) (r.y +
(r.height / 18)*13),(r.width/3+r.width/10) , (int) (r.height / 4.0));
            Rect mouth_area = new Rect(r.x + r.width / 2- r.width /5,(int) (r.y +
r.height-r.height*9/30),
                    (r.width/3+r.width/10) , (int) (r.height / 4.0));
            // draw the area - mGray is working grayscale mat, if you want to
            // see area in rgb preview, change mGray to mRgba
//打印显示信息
            Core.rectangle(mRgba, eyearea_left.tl(), eyearea_left.br(),
            new Scalar(255, 0, 0, 255), 2);
            Core.rectangle(mRgba, eyearea_right.tl(), eyearea_right.br(),
            new Scalar(255, 0, 0, 255), 2);
            Core.rectangle(mRgba, mouth_area.tl(), mouth_area.br(),
              new Scalar(255, 0, 0, 255), 2);
//两个眼睛的识别
```

```
            teplateR = get_template(mJavaDetectorEyeR, eyearea_right, 24,0);
            teplateL = get_template(mJavaDetectorEyeL, eyearea_left, 24,1);

distance1=Math.sqrt((eyel_x-eyer_x)*(eyel_x-eyer_x)+(eyel_y-eyer_y)*(eyel_y-eyer
_y));
          if(distance2!=0.0){
              if(distance2<=0.5*distance1)
                    Core.putText(mRgba, "HEI      stay focus !!!",new Point(center.x
+ 20, center.y + 20),
                    Core.FONT_HERSHEY_SIMPLEX, 0.7, new Scalar(255, 255,
255, 255));
            }
            distance2=distance1;

          /* Core.putText(mRgba,distance1+"!!!" ,new Point(center.x + 20, center.y +
20),
                    Core.FONT_HERSHEY_SIMPLEX, 0.7, new Scalar(255, 255,
255, 255));
            */
//识别嘴部，从上面分离的区域中识别嘴部
            teplateM= get_template(mJavaDetectorMouth, mouth_area, 60,2);

            if(!teplateR.empty())
                status_R=mNativeDetector.judge(teplateR,2);//TODO
            Core.putText(mRgba, "teplateR:"+status_R+"!!!!!!!!!!", new Point(center.x
-100, center.y - 20),
                    Core.FONT_HERSHEY_SIMPLEX, 0.7, new Scalar(100, 0, 0,
100));
            if(!teplateL.empty())
                status_L=mNativeDetector.judge(teplateL,2);
            Core.putText(mRgba, "teplateL:"+status_L+"!!!!!!!!!!", new Point(center.x
+100, center.y -20),
                    Core.FONT_HERSHEY_SIMPLEX, 0.7, new Scalar(100, 0, 0,
100));
            if(!teplateM.empty())
                /**status_M=mNativeDetector.judge(teplateM,1);
                if(status_M<=0.23*teplateM.cols())
                    status_M=0;
                else
                    status_M=1;*/
                status_M=mNativeDetector.judge_m(teplateM);
          // Imgproc.resize(teplateM, mZoomWindow1,mZoomWindow1.size());
            Core.putText(mRgba,
"teplateM:"+status_M+"--width:"+teplateM.cols()+"!!!!!!!!!!", new Point(center.x ,
```

```
center.y +100),
                    Core.FONT_HERSHEY_SIMPLEX, 0.7, new Scalar(100, 100, 100,
100));
            }

            return mRgba;
        }

        @Override
        //设置 menulist android API
        public boolean onCreateOptionsMenu(Menu menu) {
            Log.i(TAG, "called onCreateOptionsMenu");
            mItemFace50 = menu.add("Face size 50%");
            mItemFace40 = menu.add("Face size 40%");
            mItemFace30 = menu.add("Face size 30%");
            mItemFace20 = menu.add("Face size 20%");
            mItemFace10 = menu.add("Face size 10%");
            mItemType     = menu.add(mDetectorName[mDetectorType]);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            Log.i(TAG, "called onOptionsItemSelected; selected item: " + item);
            if (item == mItemFace50)
                setMinFaceSize(0.5f);
            else if (item == mItemFace40)
                setMinFaceSize(0.4f);
            else if (item == mItemFace30)
                setMinFaceSize(0.3f);
            else if (item == mItemFace20)
                setMinFaceSize(0.2f);
            else if (item == mItemFace10)
               setMinFaceSize(0.1f);
            else if (item == mItemType) {
                int tmpDetectorType = (mDetectorType + 1) %
mDetectorName.length;
                item.setTitle(mDetectorName[tmpDetectorType]);
                setDetectorType(tmpDetectorType);
            }
            return true;
        }

        private void setMinFaceSize(float faceSize) {
```

```
            mRelativeFaceSize = faceSize;
            mAbsoluteFaceSize = 0;
        }
        private void CreateAuxiliaryMats() {
        if (mGray.empty())
        return;
        int rows = mGray.rows();
        int cols = mGray.cols();
        if (mZoomWindow == null) {
        //mZoomWindow = mRgba.submat(rows / 2 + rows / 10, rows, cols / 2+ cols / 10,
cols);
        mZoomWindow = mRgba.submat(0, rows / 2 - rows / 8, 0, cols / 2-cols / 8);
        mZoomWindow2 = mRgba.submat(0, rows / 2 - rows / 8, cols / 2+ cols / 8, cols);
        }
        if( mZoomWindow1==null){
            mZoomWindow1 = mRgba.submat((rows/3)*2, rows , cols /3, 4*cols / 6);
        }
        }
        //获取兴趣区矩形 函数（嘴部和眼部）
        private Mat get_template(CascadeClassifier clasificator, Rect area, int size,int
type) {
            Mat template = new Mat();
            Mat mROI = mGray.submat(area);
            MatOfRect eyes = new MatOfRect();
            Point iris = new Point();
            Rect eye_template = new Rect();
            if(type==1||type==0)//探测嘴巴还是眼睛
            clasificator.detectMultiScale(mROI, eyes, 1.1,
2,Objdetect.CASCADE_FIND_BIGGEST_OBJECT |
Objdetect.CASCADE_SCALE_IMAGE, new Size(50, 50),
            new Size());
            else
            clasificator.detectMultiScale(mROI, eyes, 1.1,
2,Objdetect.CASCADE_FIND_BIGGEST_OBJECT, new Size(30, 30),new Size());
            Rect[] eyesArray = eyes.toArray();
            if(eyesArray.length>0){
            Rect e = eyesArray[0];//在该区域内只提取一个眼部矩形
            e.x = area.x + e.x;
            e.y = area.y + e.y;
                        if(type==1||type==0){
                        Rect eye_only_rectangle = new Rect((int) e.tl().x,
                        (int) (e.tl().y + e.height * 0.4), (int) e.width,
                        (int) (e.height * 0.6));
                        mROI = mGray.submat(eye_only_rectangle);
```

```
                        Mat vyrez = mRgba.submat(eye_only_rectangle);
                        Core.MinMaxLocResult mmG = Core.minMaxLoc(mROI);
                         Core.circle(vyrez, mmG.minLoc, 4, new Scalar(255, 255, 255,
255), 2);
                        if(type==1)//左眼中心
                        {eyel_x=mmG.minLoc.x;
                           eyel_y=mmG.minLoc.y;
                        }if(type==0){//右眼中心位置
                           eyer_x=mmG.minLoc.x;
                           eyer_y=mmG.minLoc.y;
                        }
                        iris.x = mmG.minLoc.x + eye_only_rectangle.x;
                        iris.y = mmG.minLoc.y + eye_only_rectangle.y;
                        eye_template = new Rect((int) iris.x - size / 2, (int) iris.y- size /
2, size, size);
                                                Core.rectangle(mRgba,       eye_template.tl(),
eye_template.br(),new Scalar(0, 0, 0, 0), 1);
                        template = (mGray.submat(eye_template)).clone();//TODO
                        return template;//返回眼睛区域矩形
                }
                if(type==2){//嘴部检测部分

                    //Rect          mouth_only_rectangle          =          new          Rect((int)
(e.tl().x+(e.width-(e.height+e.width)/2)/2),(int)          (e.tl().y          ),          (int)
((e.height+e.width)/2),(int) ((e.height+e.width)/2));
                    Rect mouth_only_rectangle = new Rect((int) (e.tl().x),(int) (e.tl().y ), (int)
((e.height*0.3+e.width*0.7)),(int) ((e.height*0.3+e.width*0.7)));
                                        Core.rectangle(mRgba,       mouth_only_rectangle.tl(),
mouth_only_rectangle.br(),new Scalar(0, 0, 0, 0), 1);//框住  嘴部
                     width_m=e.width;
                     height_m=e.height;
                     template = (mRgba.submat(mouth_only_rectangle )).clone();//复制嘴部
区域返回该矩形；
                     return template;//返回嘴部矩形区

                }
                }
                return template;            }
        private void setDetectorType(int type) {//选择本地方法和 android 方法
                if (mDetectorType != type) {
                     mDetectorType = type;

                     if (type == NATIVE_DETECTOR) {
                          Log.i(TAG, "Detection Based Tracker enabled");
```

```
                    mNativeDetector.start();
                } else {
                    Log.i(TAG, "Cascade detector enabled");
                    mNativeDetector.stop();
                }
            }
        }
    }
```

```
//Native jni    本地方法类 DetectionBasedTracker_jni.cpp
#include <DetectionBasedTracker_jni.h>
#include <opencv2/core/core.hpp>
#include <opencv2/contrib/detection_based_tracker.hpp>
#include <opencv2/opencv.hpp>
#include <cv.h>
#include <highgui.h>
#include <string>
#include <vector>
#include "img.cpp";
#include <android/log.h>

#define LOG_TAG "FaceDetection/DetectionBasedTracker"
#define LOGD(...) ((void)__android_log_print(ANDROID_LOG_DEBUG,
LOG_TAG, __VA_ARGS__))

using namespace std;
using namespace cv;
/**扩充**/
#define CLOSE 1
#define OPEN 0
#define OFFSET -20
#define STEP 8      //平滑步长
#define STEP_F 5   //寻找局部峰值步长
IplImage* painty;
IplImage* painty2;
int WITH=90,HEIGHT=60;
int x,y;
CvScalar s,t;
int summit_num = 0;//波峰数
int k;
int summit_1=0;//峰值坐标 1
int summit_2=0;//峰值坐标 2
int check;//检测点
```

```
int temp = 0;

img arr_img[100];
vector<vector<Point> > contours;

int trs = 0;//像素值发生改变标志
IplImage* bin_nom_eye;
/*扩充结束**/
void sort_img();
void exchange(img * p1,img * p2);
inline int otsu(IplImage* src)//大津算法（最大类间差算法）
{
	int height;
	int width;
	int i,j;
	unsigned char*p;
	float histogram[256] = {0};
	int size;
	float avgValue=0;
	int threshold;
	float maxVariance=0;
	float w = 0, u = 0;
	float t;
	float variance;

	height = src->height;
	width = src->width;
	//histogram
	for(i=0; i < height; i++)
	{
		p = (unsigned char*)src->imageData + src->widthStep * i;
		for(j = 0; j < width; j++)
		{
			histogram[*p++]++;
		}
	}
	//normalize histogram
	size = height * width;
	for(i = 0; i < 256; i++)
	{
		histogram[i] = histogram[i] / size;
	}

	//average pixel value
```

```
    for(i=0; i < 256; i++)
    {
        avgValue += i * histogram[i];   //整幅图像的平均灰度
    }

    for(i = 0; i < 256; i++)
    {
        w += histogram[i];
        u += i * histogram[i];
        t = avgValue * w - u;
        variance = t * t / (w * (1 - w) );
        if(variance > maxVariance)
        {
            maxVariance = variance;
            threshold = i;
        }
    }
    return threshold+OFFSET;
}

inline void smooth(int a[], int n)//平滑、寻找峰值
{
    int start = STEP - STEP/2;
    int end = n - STEP/2 ;
    double temp = 0;
    int i,j;

        int res[256] = {0};

    for (i=start; i<end; i++)
    {
        temp = 0;
        for (j=0-STEP/2; j<STEP/2; j++)
            temp += a[i + j];
                temp /= STEP;
                res[i] = (int)temp;
    }
    for (i=0; i<n; i++)
        a[i] = res[i];
//find summit
    for (i=0; i<n; i++)
        res[i]=0;
    for (i=start; i<end; i++)
        for(j=i-STEP_F/2; j<=i+STEP_F/2; j++)
```

```
            {
                if(a[i]<a[j])
                {
                    res[i]=0;
                    break;
                }
                else
                    res[i]=a[i];
            }
        for (i=0; i<n; i++)
        {
            if(res[i] != res[i-1])
                a[i]=res[i];
        }

        return;
}


inline void vector_Rect_to_Mat(vector<Rect>& v_rect, Mat& mat)
{
     mat = Mat(v_rect, true);
}

JNIEXPORT jlong JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeCreate
Object
(JNIEnv * jenv, jclass, jstring jFileName, jint faceSize)
{

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
CreateObject enter");
        const char* jnamestr = jenv->GetStringUTFChars(jFileName, NULL);
        string stdFileName(jnamestr);
        jlong result = 0;

        try
        {
            DetectionBasedTracker::Parameters DetectorParams;
            if (faceSize > 0)
                DetectorParams.minObjectSize = faceSize;
            result = (jlong)new DetectionBasedTracker(stdFileName,
DetectorParams);
        }
```

```
    catch(cv::Exception& e)
    {
        LOGD("nativeCreateObject caught cv::Exception: %s", e.what());
        jclass je = jenv->FindClass("org/opencv/core/CvException");
        if(!je)
            je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, e.what());
    }
    catch (...)
    {
        LOGD("nativeCreateObject caught unknown exception");
        jclass je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, "Unknown exception in JNI code of
DetectionBasedTracker.nativeCreateObject()");
        return 0;
    }


LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
CreateObject exit");
    return result;
}

JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeDestro
yObject
(JNIEnv * jenv, jclass, jlong thiz)
{

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
DestroyObject enter");
    try
    {
        if(thiz != 0)
        {
            ((DetectionBasedTracker*)thiz)->stop();
            delete (DetectionBasedTracker*)thiz;
        }
    }
    catch(cv::Exception& e)
    {
        LOGD("nativeestroyObject caught cv::Exception: %s", e.what());
        jclass je = jenv->FindClass("org/opencv/core/CvException");
        if(!je)
```

```
            je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, e.what());
    }
    catch (...)
    {
        LOGD("nativeDestroyObject caught unknown exception");
        jclass je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, "Unknown exception in JNI code of
DetectionBasedTracker.nativeDestroyObject()");
    }

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
DestroyObject exit");
}

JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeStart
(JNIEnv * jenv, jclass, jlong thiz)
{

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Start enter");
    try
    {
        ((DetectionBasedTracker*)thiz)->run();
    }
    catch(cv::Exception& e)
    {
        LOGD("nativeStart caught cv::Exception: %s", e.what());
        jclass je = jenv->FindClass("org/opencv/core/CvException");
        if(!je)
            je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, e.what());
    }
    catch (...)
    {
        LOGD("nativeStart caught unknown exception");
        jclass je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, "Unknown exception in JNI code of
DetectionBasedTracker.nativeStart()");
    }

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Start exit");
```

```
}

JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeStop
(JNIEnv * jenv, jclass, jlong thiz)
{

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Stop enter");
    try
    {
        ((DetectionBasedTracker*)thiz)->stop();
    }
    catch(cv::Exception& e)
    {
        LOGD("nativeStop caught cv::Exception: %s", e.what());
        jclass je = jenv->FindClass("org/opencv/core/CvException");
        if(!je)
            je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, e.what());
    }
    catch (...)
    {
        LOGD("nativeStop caught unknown exception");
        jclass je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, "Unknown exception in JNI code of
DetectionBasedTracker.nativeStop()");
    }

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Stop exit");
}

JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeSetFac
eSize
(JNIEnv * jenv, jclass, jlong thiz, jint faceSize)
{

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
SetFaceSize enter");
    try
    {
        if (faceSize > 0)
```

```
            {
                DetectionBasedTracker::Parameters DetectorParams = \
                ((DetectionBasedTracker*)thiz)->getParameters();
                DetectorParams.minObjectSize = faceSize;

((DetectionBasedTracker*)thiz)->setParameters(DetectorParams);
            }
        }
        catch(cv::Exception& e)
        {
            LOGD("nativeStop caught cv::Exception: %s", e.what());
            jclass je = jenv->FindClass("org/opencv/core/CvException");
            if(!je)
                je = jenv->FindClass("java/lang/Exception");
            jenv->ThrowNew(je, e.what());
        }
        catch (...)
        {
            LOGD("nativeSetFaceSize caught unknown exception");
            jclass je = jenv->FindClass("java/lang/Exception");
            jenv->ThrowNew(je, "Unknown exception in JNI code of
DetectionBasedTracker.nativeSetFaceSize()");
        }

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
SetFaceSize exit");
}

//兴趣区探测函数
JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeDetect
(JNIEnv * jenv, jclass, jlong thiz, jlong imageGray, jlong faces)
{

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Detect enter");
        try
        {
            vector<Rect> RectFaces;
            ((DetectionBasedTracker*)thiz)->process(*((Mat*)imageGray));
            ((DetectionBasedTracker*)thiz)->getObjects(RectFaces);
            vector_Rect_to_Mat(RectFaces, *((Mat*)faces));
        }
        catch(cv::Exception& e)
```

```
    {
        LOGD("nativeCreateObject caught cv::Exception: %s", e.what());
        jclass je = jenv->FindClass("org/opencv/core/CvException");
        if(!je)
            je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, e.what());
    }
    catch (...)
    {
        LOGD("nativeDetect caught unknown exception");
        jclass je = jenv->FindClass("java/lang/Exception");
        jenv->ThrowNew(je, "Unknown exception in JNI code
DetectionBasedTracker.nativeDetect()");
    }

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Detect exit");
}

JNIEXPORT jint JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeJudge
(JNIEnv * jenv, jclass, jlong thiz, jlong imageGray,jint type)
{
  //判断函数眼睛开闭函数
LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Detect enter");
    try
    {   int threshod;
        jint status;
        jint max_m;
        int max=0,cur=0;
       IplImage ipl_img(*(Mat*)imageGray);
        threshod=otsu(&ipl_img);
        WITH=ipl_img.width;
        HEIGHT=ipl_img.height;
        int DEPTH=ipl_img.depth;
        int CHANNEL=ipl_img.nChannels;
        int h_acc[HEIGHT];
        bin_nom_eye = cvCreateImage(cvSize(WITH,HEIGHT),DEPTH,
CHANNEL);
       cvThreshold(&ipl_img, bin_nom_eye,threshod, 255,
CV_THRESH_BINARY);
       painty = cvCreateImage(cvSize(WITH,HEIGHT),DEPTH,
CHANNEL);
```

```
cvZero(painty);
memset(h_acc, 0, HEIGHT*4);
for(y=0; y<HEIGHT; y++)
    for(x=0; x<WITH; x++)
    {
        s = cvGet2D(bin_nom_eye,y, x);
        if(s.val[0] == 0)
            h_acc[y]++;
    }
for(y=0; y<HEIGHT; y++)
{
    if(h_acc[y]>8){
            cur++;
            }else{
            if(max<cur)
            max=cur;
            cur=0;
            }
}
if(max<cur)
max=cur;
smooth(h_acc, HEIGHT);
if(type==2){

//检测波峰或波峰间二值图是否有空洞
for(k=0; k<HEIGHT; k++)
{
    if(h_acc[k]!=0)
    {
        summit_num++;
        if(h_acc[k]>summit_1 && h_acc[k]>summit_2)
        {
            summit_2 = summit_1;
            summit_1 = k;
        }
        else if(h_acc[k]>summit_2)
            summit_2 = k;
    }
}
if(summit_num == 1)
    check = summit_1;
else if(summit_num >= 2)
    check = (summit_1 + summit_2)/2;
trs = 0;
```

```
            s = cvGet2D(bin_nom_eye, check, 1);
            temp = s.val[0];
            for(x=0; x<WITH; x++)
            {
                s = cvGet2D(bin_nom_eye, check, x);
                if(s.val[0]==0 && temp==255)
                    trs++;
                else if(trs==1 && s.val[0]==255 && temp==0 )
                    trs++;
                temp = s.val[0];
            }
            if(trs >= 2)
                status = 1;
            else
                status = 0;
            return status;
            }else if(type==1){
                max_m=max;
                return max_m;
            }
        }
        catch(cv::Exception& e)
        {
            LOGD("nativeCreateObject caught cv::Exception: %s", e.what());
            jclass je = jenv->FindClass("org/opencv/core/CvException");
            if(!je)
                je = jenv->FindClass("java/lang/Exception");
            jenv->ThrowNew(je, e.what());
        }
        catch (...)
        {
            LOGD("nativeDetect caught unknown exception");
            jclass je = jenv->FindClass("java/lang/Exception");
            jenv->ThrowNew(je, "Unknown exception in JNI code
DetectionBasedTracker.nativeDetect()");
        }

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Detect exit");
}

// 进行肤色检测
IplImage* MouthSegmentHSV(IplImage* srcs,IplImage* imgout)
{
```

```
  //定义一些中间指针，指向处理过程中的中间变量
  IplImage* HSV = NULL;
  IplImage* HImg= NULL;
  IplImage* SImg= NULL;
  IplImage* VImg= NULL;
  //指向处理后的结果
IplImage* result = NULL;
  if (!srcs||!imgout)   {     return NULL;   }
  //获取输入图像的大小
  CvSize SrcSize = cvGetSize(srcs);
  //为中间结果指针分配存储空间
  HSV = cvCreateImage(SrcSize,8,3);
  HImg= cvCreateImage(SrcSize,8,1);
  IplImage* src = cvCreateImage(SrcSize,8,3);
SImg= cvCreateImage(SrcSize,8,1);
VImg= cvCreateImage(SrcSize,8,1);
  result= cvCreateImage(cvGetSize(imgout),8,1);
  //将图像从 RGB 颜色空间转换到 HSV 空间
  cvCvtColor(srcs,src,CV_RGBA2BGR);
  //将图像从 RGB 颜色空间转换到 HSV 空间
  cvCvtColor(src,HSV,CV_BGR2HSV);
  //将 HSV 分解为三张单通道图像，便于后面就行处理
cvSplit(HSV,HImg,SImg,VImg,NULL);
     int i,j;   int value = 0;
//对通道 H 进行处理
for (i = 0; i < HImg->height; i++)
  {
              for (j = 0; j < HImg->width; j++)
                {
                        value = cvGetReal2D(HImg,i,j);
                        if ((value >=0 && value <= 25 )|| value>=350)     //25
                        {       *(HImg->imageData+i*HImg->widthStep+j) =
255;      }
                    else      {       *(HImg->imageData+i*HImg->widthStep+j)
= 0;      }
                  }
  }
  //对通道 S 进行处理
  for (i = 0; i < SImg->height; i++)   {
       for (j = 0; j < SImg->width; j++)
         {
             value = cvGetReal2D(SImg,i,j);
  if ((value >=0 && value <= 255) )     //26
  {       *(SImg->imageData+i*SImg->widthStep+j) = 255;       }
```

```
else      {       *(SImg->imageData+i*SImg->widthStep+j) = 0;       }
 }
 }
 //对通道 V 进行处理
 for (i = 0; i < VImg->height; i++)   {
 for (j = 0; j < VImg->width; j++)    {
 value = cvGetReal2D(VImg,i,j);
 if (value >=0 && value <= 255)        {
      *(VImg->imageData+i*VImg->widthStep+j) = 255;        }
else       {         *(VImg->imageData+i*VImg->widthStep+j) =
0;       }      }    }
   cvAnd(HImg,SImg,result,0);
   cvAnd(VImg,result,result,0);
 //对所得到的结果进行形态学腐蚀处理，去除小区域部分
 //cvErode(result,result);   cvErode(result,result);   cvErode(result,result);
cvDilate(result,result);
 //将处理后的结果赋值给输出图像
 cvCopy(result,imgout);
 //释放相关资源
cvReleaseImage(&src);
cvReleaseImage(&HSV);
cvReleaseImage(&HImg);
cvReleaseImage(&SImg);
cvReleaseImage(&VImg);
cvReleaseImage(&result);
 //返回处理后的结果
 return imgout;
}

//判断嘴部开闭函数
JNIEXPORT jint JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeJudge
m
(JNIEnv * jenv, jclass, jlong thiz, jlong imageGray)
{

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Detect enter");
      try
      {
            jint status;
            int summit_num = 0;//波峰数
            int k;
            int summit_1=0;//峰值坐标 1
```

```
            int summit_2=0;//峰值坐标 2
            int check;//检测点
            int temp = 0;
            IplImage src=IplImage(*(Mat*)imageGray);
            IplImage* ipl_img=&src;
            WITH=ipl_img->width;
            HEIGHT=ipl_img->height;
            int DEPTH=ipl_img->depth;
            int CHANNEL=ipl_img->nChannels;
            IplImage *dsc = cvCreateImage(cvSize(WITH,HEIGHT),DEPTH, 1);
            IplImage*
painty=cvCreateImage( cvSize(WITH,HEIGHT),IPL_DEPTH_8U, 1 );
            cvZero(painty);
            int* v=new int[dsc->width];
                  int* h=new int[dsc->height];
                  memset(v,0,dsc->width*4);
                  memset(h,0,dsc->height*4);
                 int x,y;
                  CvScalar s,t;
                int max=0,cur=0;
                MouthSegmentHSV(ipl_img, dsc);
                  for(y=0;y<HEIGHT;y++)
                     {
                           for(x=0;x<WITH;x++)
                           {
                                 s=cvGet2D(dsc,y,x);
                                 if(s.val[0]==0)
                                       h[y]++;
                           //cout<<s.val[0]<<" ";
                           }
                     //cout<<endl;
                     }
                     for(y=0;y<dsc->height;y++)
                     {
                        LOGD(" h[y] is ******************************%d",h[y]);
                     if(h[y]>10){
                     cur++;
                     }else{
                     if(max<cur)
                     max=cur;
                     cur=0;
                     }
                     //cout<<"第"<<y<<"个"<<h[y]<<endl;
                     }
```

```
                if(max<cur)
                max=cur;
                status=max;
                smooth(h, HEIGHT);
                //检测空洞
                for(k=0; k<HEIGHT; k++)
                {
                    if(h[k]!=0)
                    {
                        summit_num++;
                        if(h[k]>summit_1 && h[k]>summit_2)
                        {
                            summit_2 = summit_1;
                            summit_1 = k;
                        }
                        else if(h[k]>summit_2)
                            summit_2 = k;
                    }
                }
                if(summit_num == 1)
                    check = summit_1;
                else if(summit_num >= 2)
                    check = (summit_1 + summit_2)/2;
                trs = 0;
                s = cvGet2D(dsc, check, 1);
                temp = s.val[0];
                for(x=0; x<WITH; x++)
                {
                    s = cvGet2D(dsc, check, x);
                    if(s.val[0]==0 && temp==255)
                        trs++;
                    else if(trs==1 && s.val[0]==255 && temp==0 )
                        trs++;
                    temp = s.val[0];
                }
                if(trs >= 2)
                    status = OPEN;
                else
                    status = CLOSE;
                return status;
}
catch(cv::Exception& e)
{
    LOGD("nativeCreateObject caught cv::Exception: %s", e.what());
```

```
            jclass je = jenv->FindClass("org/opencv/core/CvException");
            if(!je)
                je = jenv->FindClass("java/lang/Exception");
            jenv->ThrowNew(je, e.what());
        }
        catch (...)
        {
            LOGD("nativeDetect caught unknown exception");
            jclass je = jenv->FindClass("java/lang/Exception");
            jenv->ThrowNew(je, "Unknown exception in JNI code
DetectionBasedTracker.nativeDetect()");
        }

LOGD("Java_org_opencv_samples_facedetect_DetectionBasedTracker_native
Detect exit");
}

void sort_img(){
    for(int i=0;i<contours.size();i++){
        for(int j=1;j<contours.size()-i;j++){
            if(arr_img[j-1].nums<arr_img[j].nums)
                exchange(&arr_img[j-1],&arr_img[j]);
        }
    }
}

void exchange(img * p1,img * p2){
    int id_;
    int nums_;
    id_ = p1->id;
    nums_=p1->nums;
    p1->id=p2->id;
    p1->nums=p2->nums;
    p2->id=id_;
    p2->nums=nums_;
}

//DetectionBasedTracker_jni.h
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class org_opencv_samples_fd_DetectionBasedTracker */

#ifndef _Included_org_opencv_samples_fd_DetectionBasedTracker
#define _Included_org_opencv_samples_fd_DetectionBasedTracker
```

```
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     org_opencv_samples_fd_DetectionBasedTracker
 * Method:    nativeCreateObject
 * Signature: (Ljava/lang/String;F)J
 */
JNIEXPORT jlong JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeCreate
Object
  (JNIEnv *, jclass, jstring, jint);

/*
 * Class:     org_opencv_samples_fd_DetectionBasedTracker
 * Method:    nativeDestroyObject
 * Signature: (J)V
 */
JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeDestro
yObject
  (JNIEnv *, jclass, jlong);

/*
 * Class:     org_opencv_samples_fd_DetectionBasedTracker
 * Method:    nativeStart
 * Signature: (J)V
 */
JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeStart
  (JNIEnv *, jclass, jlong);

/*
 * Class:     org_opencv_samples_fd_DetectionBasedTracker
 * Method:    nativeStop
 * Signature: (J)V
 */
JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeStop
  (JNIEnv *, jclass, jlong);

  /*
   * Class:     org_opencv_samples_fd_DetectionBasedTracker
   * Method:    nativeSetFaceSize
```

```
     * Signature: (JI)V
     */
    JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeSetFac
eSize
    (JNIEnv *, jclass, jlong, jint);

/*
 * Class:     org_opencv_samples_fd_DetectionBasedTracker
 * Method:    nativeDetect
 * Signature: (JJJ)V
 */
JNIEXPORT void JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeDetect
    (JNIEnv *, jclass, jlong, jlong, jlong);

//修改 judge
JNIEXPORT jint JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeJudge
    (JNIEnv *, jclass, jlong, jlong,jint);
//检测 judge_m
JNIEXPORT jint JNICALL
Java_org_opencv_samples_facedetect_DetectionBasedTracker_nativeJudge
m
    (JNIEnv *, jclass, jlong, jlong);

#ifdef __cplusplus
}
#endif
#endif
```

//Adroid.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

#OPENCV_CAMERA_MODULES:=off
#OPENCV_INSTALL_MODULES:=off
#OPENCV_LIB_TYPE:=SHARED
include
E:\thesunE\libs\opencv\android-opencv\OpenCV-2.4.9-android-sdk\sdk\native\j
ni\OpenCv.mk
```

```
LOCAL_SRC_FILES  := DetectionBasedTracker_jni.cpp
LOCAL_C_INCLUDES += $(LOCAL_PATH)
LOCAL_LDLIBS     += -llog -ldl
LOCAL_MODULE     := detection_based_tracker

include $(BUILD_SHARED_LIBRARY)
```

//Application.mk

```
APP_STL := gnustl_static
APP_CPPFLAGS := -frtti -fexceptions
APP_ABI := armeabi-v7a
APP_PLATFORM := android-8
```