Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Волгоградский государственный технический университет»
Кафедра «ЭВМиС»

Семестровая работа

Тема: «Детектирование ладоней рук в реальном времени на Android устройстве»

Выполнил Титов А.К.
Группа ИВТ-360
Проверил

Волгоград 2015

Содержание

Введение

Изначально я ориентировался на написание приложения, при помощи которого можно было бы управлять Android девайсом при помощи жестов.

План работ

- Реализовать детектирование ладони и пальцев на изображении
- Реализовать данный алгоритм в реальном времени для Android устройства.
- Реализовать определение расстояния до ладони
- Реализовать управление Android устройством при помощи эмуляции событий Click

Реализовано

На данный момент частично реализованы 1 и 2 пункт.

Алгоритм поиска оказался слишком ресурсоемким для процессора моего Android устройства. Поэтому в дальнейшем запланировано реализовать его при помощи другого подхода.

Реализация

Постановка задачи

Под детектированием рук и пальцев будем понимать обнаружение таких точек, по которым можно восстановить положение ладони на плоскости и её позу. В качестве таких точек рационально использовать центр масс ладони и точки описывающие кончики пальцев.

Описание алгоритма

Поиск особой точки ладони

Сначала определим точку, являющуюся описателем ладони. Как было упомянуто выше, в качестве такой точки будем использовать центр масс контура. Для его нахождения нам потребуется вычисление пространственных моментов. Момент является характеристикой контура, вычисляемой путем интегрирования (или суммирования) всех пикселей контура.

Поиск особых точек пальцев

Теперь рассмотрим части контура соответствующие пальцам.
Для каждой точки P[n] контура, будем также рассматривать точки P[n-r], P[n+r], где r — некоторое положительное число (r<n).
Три такие точки образуют угол.
На контуре соответствующем силуэту пальцев, может быть 2 типа точек:
1) Точки лежащие на прямой(соответствуют точкам пальца). Угол P[n-r]P[n]P[n+r] тупой.
2) Точки лежащие на дугах(соответствуют кончикам пальцев и промежуткам между пальцами). Угол P[n-r]P[n]P[n+r] острый.
Нас интересуют точки второго типа, т.к. они описывают кончики пальцев.
В качестве точек описывающих кончики пальцев будем искать точки 2 типа с максимальным(в окрестности) косинусом угла P[n-r]P[n]P[n+r].
Точки 2 типа соответствуют не только кончикам пальцев, но и промежуткам между пальцами. Для определения является ли точка кончиком пальца, воспользуемся свойствами обхода контура. Пусть мы обходим пиксели контура по часовой стрелке, тогда точки соответствующие кончикам пальцев будут соответствовать правому повороту P[n]P[n+r] относительно P[n-r]P[n], а точкам лежащим в промежутке между пальцами левому повороту.
Для определения, образуют ли три точки P[n-r], P[n], P[n+r] правый поворот, можно использовать обобщение векторного произведения на двумерное пространство.
Таким образом получим координаты точек соответствующих кончикам пальцев.

Реализация алгоритма. OpenCV на Android

Я реализовал этого алгоритм использованием OpenCV на Android.

В процессе настройки я пользовался инструкциями с **Habrahabr**:

http://habrahabr.ru/post/262089/

И **StackOverflow**:

http://stackoverflow.com/questions/27406303/opencv-in-android-studio

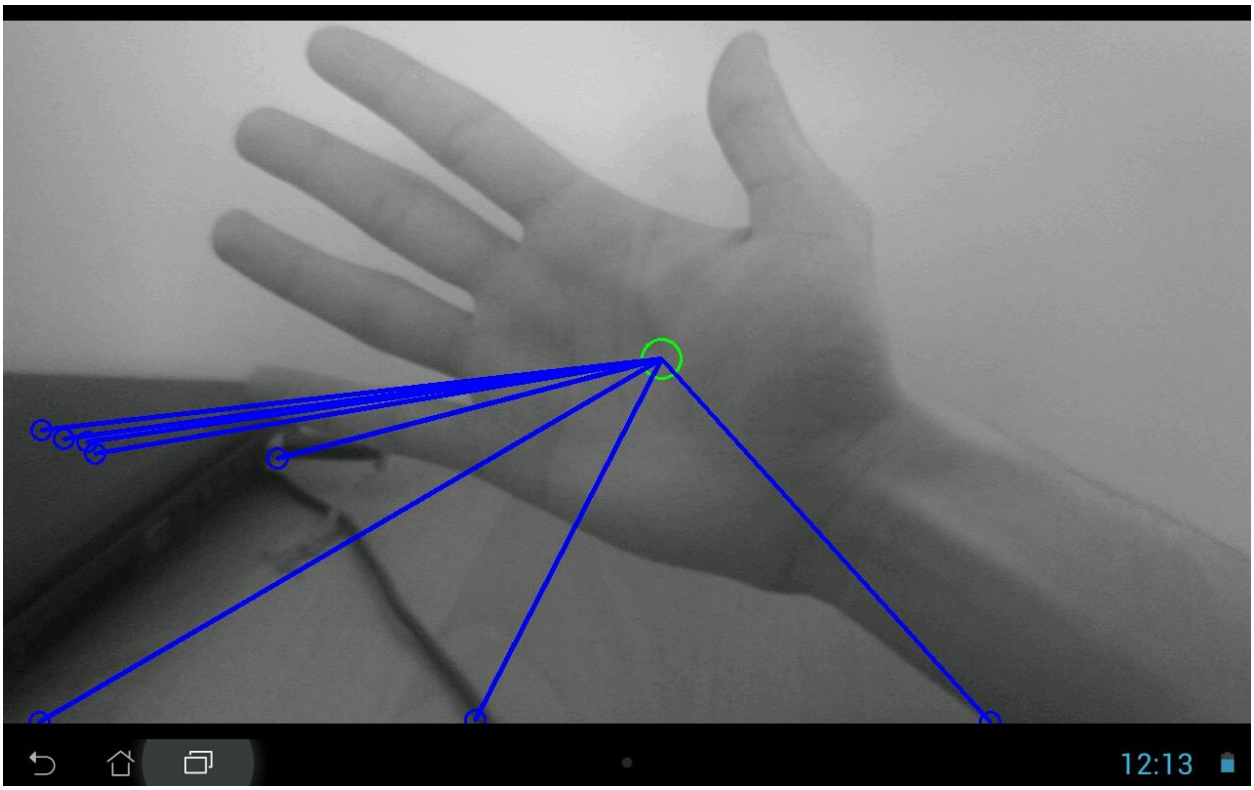
Выбранный мною способ заключался в:

1) Установке Java обвертки от создателей **OpenCV**.
2) Подключении Java обвертки к проекту в Android Studio
3) Реализация алгоритма на Java обвертке OpenCV , путем реализации интерфейса CvCameraViewListener2, у которого есть метод
   public Mat onCameraFrame(CvCameraViewFrame inputFrame), внутри которого мы можем работать с inputFrame.
   Возвращаемое из метода изображение Mat будет в последствии отображено на форме в элементе JavaCameraView.

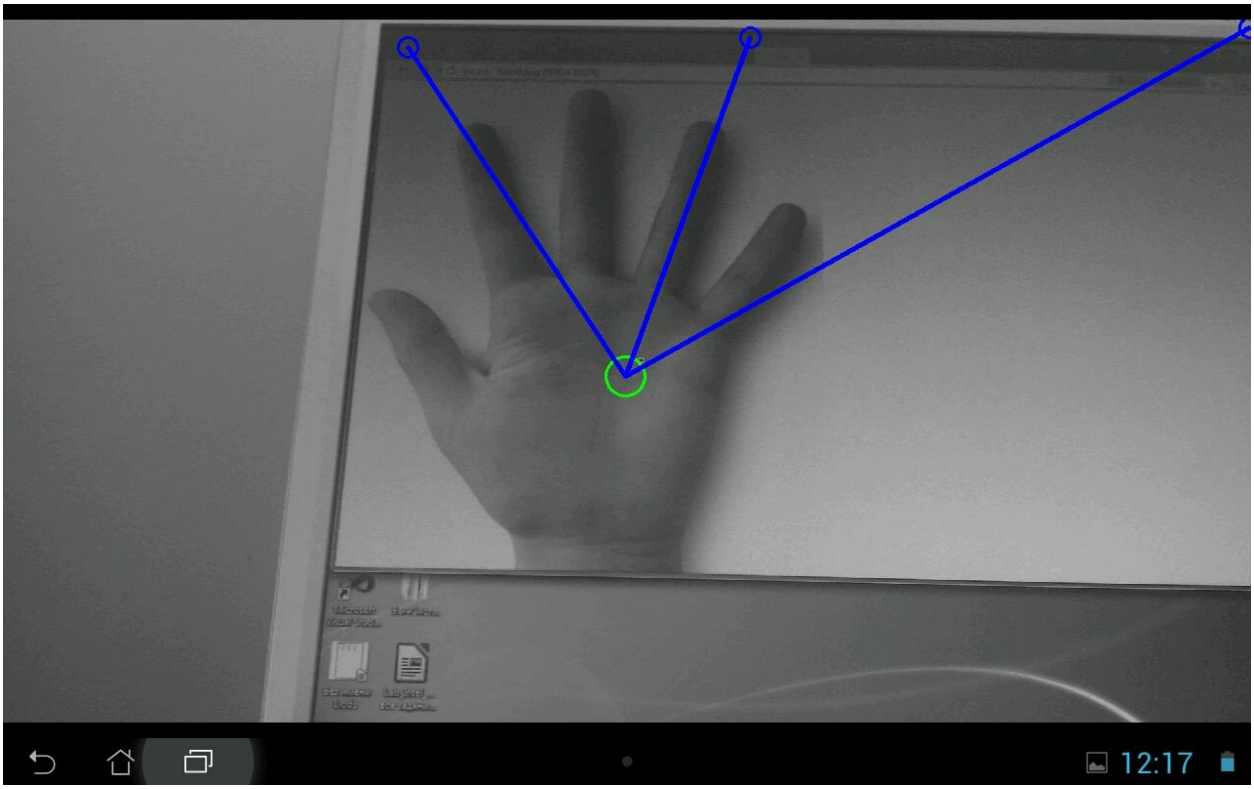
В реализации были использованы стандартные классы бибилотеки OpenCV в Java обвертке.
Я пытался работать с камерой, не реализуя CvCameraViewListener2, но, насколько мне известно, это единственный адекватный способ работы с ней в данной реализации OpenCV.

Результаты экспериментов

Поиск ладони. Как видно, центр ладони определяется отлично, однако есть проблемы с детектированием пальцев.



Поиск ладони на изображении. Та же проблема с поиском пальцев.



Код программы

MainActivity.java

```
package com.alex.handdetection;

import android.app.Activity;
import android.os.Bundle;
import android.view.SurfaceView;
import android.view.WindowManager;

import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.android.Utils;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;


public class MainActivity extends Activity implements CvCameraViewListener2 {
    private CameraBridgeViewBase mOpenCvCameraView;
    private HandDetector hd;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        setContentView(R.layout.activity_main);
        mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.view);
        mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
        mOpenCvCameraView.setCvCameraViewListener(this);
        hd = new HandDetector();
    }

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

    @Override
    public void onResume()
    {
        super.onResume();
        OpenCVLoader.initDebug();
        mOpenCvCameraView.enableView();
    }

    public void onDestroy() {
        super.onDestroy();
        if (mOpenCvCameraView != null)
            mOpenCvCameraView.disableView();
    }

    public void onCameraViewStarted(int width, int height) {
    }

    public void onCameraViewStopped() {
    }

    public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
        Mat img = hd.getImageWithHands(inputFrame.gray());
        img.convertTo(img, CvType.CV_8UC1);
        return img;//.t();
    }

    // private native Mat getFrameWithHands(Mat in);
}
```

Hand.java

```
package com.alex.handdetection;
import org.opencv.core.Point;
import java.util.ArrayList;

public class Hand
{
    Hand()
    {
        fingers = new ArrayList<Point>();
        center  = new Point();
        contour = new ArrayList<Point>();
    };
    ArrayList<Point> fingers;
    Point center;
    ArrayList<Point> contour;
};
```

HandDetector.java

```
package com.alex.handdetection;

import org.opencv.core.*;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.*;
import java.util.List;
import java.util.ArrayList;
import org.opencv.utils.Converters;


public class HandDetector
{
    class Params
    {
        int area;
        int r;
        int step;
        double cosThreshold;
        double equalThreshold;
    };

    public HandDetector(){
        param = new Params();
    };

    void detect(Mat mask, List<Hand> hands)
    {
        hands.clear();
        ArrayList<MatOfPoint> contours = new ArrayList<MatOfPoint>();
        Mat hierarchy = new Mat();
        int mysize;

        Imgproc.findContours(mask.clone(), contours, hierarchy, Imgproc.RETR_EXTERNAL,
Imgproc.CHAIN_APPROX_NONE, new Point(0, 0));
        if(!contours.isEmpty())
        {
            for(int i=0; i<contours.size(); i++)
            {
                if(Imgproc.contourArea(contours.get(i))>param.area)
                {
                    Hand tmp = new Hand();
                    Moments m=Imgproc.moments(contours.get(i));
                    tmp.center.x=m.m10/m.m00;
                    tmp.center.y=m.m01/m.m00;
```

```
                    mysize = (int) contours.get(i).toList().size();
                    for(int j = 0; j < mysize; j+= param.step ) // or height?
                    {
                        double cos0 = angle (contours.get(i).toList(), j, param.r);

                        if ((cos0 > 0.5)&&(j+param.step<mysize)) // or height?
                        {
                            double cos1 = angle (contours.get(i).toList(), j - param.step,
param.r);
                            double cos2 = angle (contours.get(i).toList(), j + param.step,
param.r);
                            double maxCos = Math.max(Math.max(cos0, cos1), cos2);
                            boolean equal = isEqual (maxCos , cos0);
                            int z = rotation (contours.get(i).toList(), j, param.r);
                            if (equal == true && z<0)
                            {
                                tmp.fingers.add(contours.get(i).toList().get(j));
                            }
                        }
                    }
                    tmp.contour = new ArrayList<Point>(contours.get(i).toList());
                    hands.add(tmp);
                }
            }
        }
    }

    void setParams(Params p)
    {
        param.area = p.area;
        param.cosThreshold = p.cosThreshold;
        param.equalThreshold = p.equalThreshold;
        param.r = p.r;
        param.step = p.step;
    }

    private Params param;
    int rotation(List<Point> contour, int pt, int r)
    {
        int size = contour.size();
        Point p0=(pt>0)?contour.get(pt%size):contour.get(size-1+pt);
        Point p1=contour.get((pt+r)%size);
        Point p2=(pt>r)?contour.get(pt-r):contour.get(size-1-r);

        double ux=p0.x-p1.x;
        double uy=p0.y-p1.y;
        double vx=p0.x-p2.x;
        double vy=p0.y-p2.y;
        return (int)(ux*vy - vx*uy);
    }
    double angle(List<Point> contour, int pt, int r)
    {
        int size = contour.size();
        Point p0=(pt>0)?contour.get(pt%size):contour.get(size-1+pt);
        Point p1=contour.get((pt+r)%size);
        Point p2=(pt>r)?contour.get(pt-r):contour.get(size-1-r);

        double ux=p0.x-p1.x;
        double uy=p0.y-p1.y;
        double vx=p0.x-p2.x;
        double vy=p0.y-p2.y;
        return (ux*vx + uy * vy) / Math.sqrt((ux * ux + uy * uy) *(vx*vx + vy*vy));
    }

    boolean isEqual(double a, double b)
    {
        return Math.abs(a - b) <= param.equalThreshold;
    }


    void drawHands(Mat image, List<Hand> hands)
    {
```

```
        int size = hands.size();
        ArrayList<ArrayList<Point>> c = new ArrayList<ArrayList<Point>>();
        for(int i = 0; i<size; i++)
        {
            c.clear();
            c.add(hands.get(i).contour);
            Imgproc.circle(image, hands.get(i).center, 20, new Scalar(0, 255, 0), 2);
            int fingersSize = hands.get(i).fingers.size();
            for(int j = 0; j < fingersSize; j++)
            {
                Imgproc.circle(image, hands.get(i).fingers.get(j), 10, new Scalar(0, 0, 255),
2);
                Imgproc.line(image, hands.get(i).center, hands.get(i).fingers.get(j), new
Scalar(0, 0, 255), 4);
            }
        }
    }

    public Mat getImageWithHands(Mat in)
    {
        Params p = new Params();
        p.area=3000;
        p.cosThreshold=0.3;
        p.equalThreshold=1e-7;
        p.r=40;
        p.step=15;

        setParams(p);

        ArrayList<Hand> hands = new ArrayList();
        Mat depthMap = new Mat();
        Mat bgrImage = new Mat();
        in.clone().convertTo(depthMap, CvType.CV_8UC1);
        in.clone().convertTo(bgrImage, CvType.CV_32FC1);

        // Mat bgrImage = new Mat(); it's in argument of this method

        //Mat tmp = new Mat();
        Mat tmp = new Mat();
        Imgproc.cvtColor(depthMap.clone(), tmp, Imgproc.COLOR_GRAY2BGR); // CV_GRAY_TO_BGR
        Mat depthMapCopy = new Mat();
        Imgproc.threshold(depthMap, depthMapCopy, 60, 255, Imgproc.THRESH_BINARY);

        detect(depthMapCopy, hands);

        if(!hands.isEmpty())
        {
            //drawHands(tmp, hands);
            drawHands(tmp, hands);
        }

        return tmp;
    }
};
```

activity_main.xml

```
<FrameLayout
        xmlns:android=http://schemas.android.com/apk/res/android
        xmlns:opencv="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <org.opencv.android.JavaCameraView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:visibility="gone"
        android:id="@+id/view"
        opencv:camera_id="any" />
</FrameLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.alex.handdetection" >
        <supports-screens android:resizeable="true"
            android:smallScreens="true"
            android:normalScreens="true"
            android:largeScreens="true"
            android:anyDensity="true" />

        <uses-permission android:name="android.permission.CAMERA"/>

        <uses-feature android:name="android.hardware.camera" android:required="true"/>
        <uses-feature android:name="android.hardware.camera.autofocus"
android:required="true"/>
        <uses-feature android:name="android.hardware.camera.front"
android:required="true"/>
        <uses-feature android:name="android.hardware.camera.front.autofocus"
android:required="true"/>

        <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
            <activity
                android:name=".MainActivity"
                android:label="@string/app_name"
                android:screenOrientation="landscape"
                android:configChanges="keyboardHidden|orientation">
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
        </application>
    </manifest>
```

Структура проекта