

博客园 首页 新随笔 联系 订阅 管理

ZBar开发详解

博客转载自: <https://blog.csdn.net/skillcollege/article/details/38855023>

什么是ZBar ?

ZBar是一个开源库,用于扫描、读取二维码和条形码。支持的二维码包括: EAN/UPC, QR等。

如果你是一个iPhone应用开发人员,做到二维码模块的时候,是不是会考虑ZBar开源项目来助你一臂之力呢?可是我这里说的是Android平台的开发,我为什么提到ZBar项目呢,难道我要用ZBar在Android平台扫描二维码吗?对的,没有错!这将会是一个极其不错的选择。为什么这么说呢,不是很多Android开发都是用Z*来翻译二维码的么?好吧,Z*是我下一篇文章要写的,这里先抛砖引玉说一点点。我将Z*和ZBar做一个比较,说说它们的优缺点,便于大家的取舍。

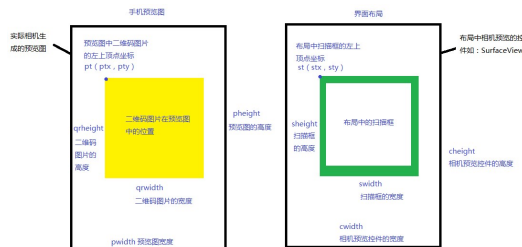
- Z*项目的示例程序对于摄像头控制写的非常全面,ZBar的没有
- ZBar基于C语言编写,解码效率高于Z*项目
- ZBar是日本人写的,对于中文解析乱码这个肯定有人遇到过的,Z*不会乱码
- 扫描框的绘制,Z*的扫描框绘制是自定义View的,截取区域不好控制,ZBar的可以自定义,只要你会计算截取区

下载ZBar项目

- ZBar官网: [传送口](#)
- ZBar GitHub地址: [传送口](#)

编写ZBar示例程序

1. 着重介绍一下扫描截取界面的计算



- 1 pt: 预览图中二维码图像的中心点坐标,也就是手机中相机预览中看到的待扫描二维码的位置
- 2 qheight: 预览图中二维码图像的高度
- 3 qwidth: 预览图中二维码图像的宽度
- 4 pheight: 预览图的高度,也就是camera的分辨率高度
- 5 pwidth: 预览图的宽度,也就是camera的分辨率宽度
- 6
- 7 st: 布局文件中扫描框的左上顶点坐标
- 8 sheight: 布局文件中扫描框的高度
- 9 swidth: 布局文件中扫描框的宽度
- 10 cheight: 布局文件中相机可见控件的高度
- 11 cwidth: 布局文件中相机可见控件的宽度

其中存在这样一个等比例公式

- ```
1 ptx / pwidth = stx / cwidth
2 pty / pheight = sty / cheight
3 qwidth / pwidth = swidth / cwidth
4 qrheight / pheight = sheight / cheight
```

并外一种表达形式

- ```
1 ptx = stx * pwidth / cwidth ;
2 pty = sty * pheight / cheight ;
3 qwidth = swidth * pwidth / cwidth ;
4 qrheight = sheight * pheight / cheight ;
```

以上ptx, pty, qwidth, qrheight四个参数也就是ZBar中解码是需要crop时传入的四个参数,如此便知道了截取区域应该如何计算了。这样扫描的灵活性都大大增强了

2. ZBar中文乱码的解决

ZBar扫描含有中文的二维码图片时,结果是乱码的,所以需要修改c文件重新编译打包so文件才行

a. 需要修改的文件是zbar/qrcode/qrcdetxt.c文件

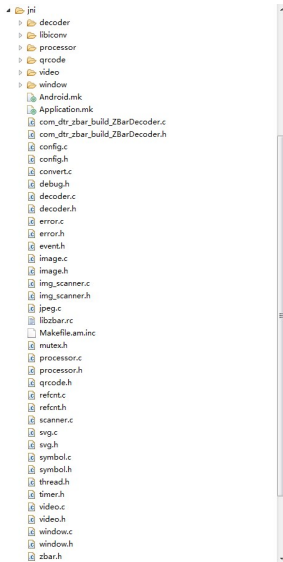
- ```
1 /*This is the encoding the standard says is the default.*/
2 latin1_cd_iconv_open("UTF-8","ISO8859-1");
```

修改为

- ```
1 /*This is the encoding the standard says is the default.*/
2 latin1_cd_iconv_open("UTF-8","GBK");
```

b. 重新编译zbar生成so文件

这个真的需要一定的NDK开发经验了,我个人只是了解一点NDK的知识,所以在网上找到了一个大神的博客一步一步做下来才算编译完成了。



最后的编译项目结构

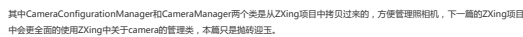


```

1 package com.dtr.zbar.build;
2
3 public class ZbarDecoder {
4
5     static {
6         System.loadLibrary("ZbarDecoder");
7     }
8
9     /**
10      * 解码方法
11      *
12      * @param data
13      *      图片数据
14      * @param width
15      *      原始宽度
16      * @param height
17      *      原始高度
18      * @return
19      */
20     public native String decodeRaw(byte[] data, int width, int height);
21
22     /**
23      * 解码方法(需要裁剪图片)
24      *
25      * @param data
26      *      图片数据
27      * @param width
28      *      原始宽度
29      * @param height
30      *      原始高度
31      * @param x
32      *      裁剪的x坐标
33      * @param y
34      *      裁剪的y坐标
35      * @param width2
36      *      裁剪的区域宽度
37      * @param height2
38      *      裁剪的区域高度
39      * @return
40      */
41     public native String decodeCrop(byte[] data, int width, int height, int x, int y, int width2, int height2);
42 }

```

3. 编写Android示例程序



```

1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical" >
6
7     <RelativeLayout
8         android:id="@+id/capture_container"
9         android:layout_width="match_parent"
10        android:layout_height="match_parent" >
11
12        <FrameLayout
13            android:id="@+id/capture_preview"
14            android:layout_width="match_parent"
15            android:layout_height="match_parent" />
16
17        <ImageView
18            android:id="@+id/capture_mask_top"
19            android:layout_width="match_parent"
20            android:layout_height="128dp"
21            android:layout_alignParentTop="true"
22            android:background="@drawable/shadow" />
23
24        <RelativeLayout
25            android:id="@+id/capture_crop_view"
26            android:layout_width="128dp"
27            android:layout_height="128dp"
28            android:layout_centerHorizontal="true"
29            android:layout_below="@+id/capture_mask_top"
30            android:background="@drawable/cr_code_bg" >
31
32            <ImageView
33                android:id="@+id/capture_scan_line"
34                android:layout_width="match_parent"
35                android:layout_height="wrap_content"
36                android:layout_alignParentTop="true"
37                android:layout_marginBottom="1dp"
38                android:layout_marginTop="1dp"
39                android:src="@drawable/scan_line" />
40        </RelativeLayout>
41
42        <ImageView
43            android:id="@+id/capture_mask_bottom"
44            android:layout_width="match_parent"
45            android:layout_height="wrap_content"
46            android:layout_alignParentBottom="true"
47            android:layout_below="@+id/capture_crop_view"
48            android:background="@drawable/shadow" />
49
50        <ImageView
51            android:id="@+id/capture_mask_left"
52            android:layout_width="wrap_content"
53            android:layout_height="match_parent"
54            android:layout_above="@+id/capture_mask_bottom"
55            android:layout_alignParentLeft="true"
56            android:layout_below="@+id/capture_mask_top"
57            android:layout_toLeftOf="@+id/capture_crop_view"
58            android:background="@drawable/shadow" />
59
60        <ImageView
61            android:id="@+id/capture_mask_right"
62            android:layout_width="wrap_content"
63            android:layout_height="match_parent"
64            android:layout_above="@+id/capture_mask_bottom"
65            android:layout_alignParentRight="true"
66            android:layout_below="@+id/capture_mask_top"
67            android:layout_toRightOf="@+id/capture_crop_view"
68            android:background="@drawable/shadow" />
69    </RelativeLayout>
70
71    <Button
72        android:id="@+id/capture_restart_scan"
73        android:layout_width="match_parent"
74        android:layout_height="48dp"
75        android:layout_alignParentBottom="true"
76        android:background="@null"
77        android:gravity="center"
78        android:text="restart scan"
79        android:textColor="@android:color/white"
80        android:textSize="14sp" />
81
82    <TextView
83        android:id="@+id/capture_scan_result"
84        android:layout_width="match_parent"
85        android:layout_height="wrap_content"
86        android:layout_above="@+id/capture_restart_scan"
87        android:layout_marginBottom="128dp"
88        android:gravity="center"

```

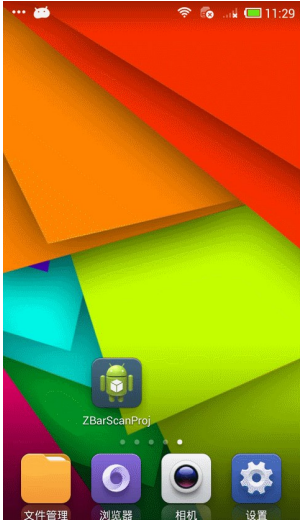
```

280         android:text="Scanning..."
281         android:textColor="@android:color/white"
282         android:textSize="14sp" />
283
284     </RelativeLayout>
285
286 c扫描Activity关键代码
287
288 1 package com.dtr.zbar.scanner;
289
290 2 import java.io.IOException;
291 3 import java.lang.reflect.Field;
292
293 4
294 5 import android.app.Activity;
295 6 import android.content.pm.ActivityInfo;
296 7 import android.graphics.Rect;
297 8 import android.hardware.Camera;
298 9 import android.hardware.Camera.AutoFocusCallback;
299 10 import android.hardware.Camera.PreviewCallback;
300 11 import android.hardware.Camera.Size;
301 12 import android.os.Bundle;
302 13 import android.os.Handler;
303 14 import android.os.Handler;
304 15 import android.text.TextUtils;
305 16 import android.view.View;
306 17 import android.view.View.OnClickListener;
307 18 import android.view.animation.Animation;
308 19 import android.view.animation.TranslateAnimation;
309 20 import android.widget.Button;
309 21 import android.widget.FrameLayout;
310 22 import android.widget.ImageView;
311 23 import android.widget.RelativeLayout;
312 24 import android.widget.TextView;
313
314 25 import com.dtr.zbar.build.ZBarDecoder;
315
316 26
317 27 public class CaptureActivity extends Activity {
318 28
319 29     private Camera mCamera;
320 30     private CameraPreview mPreview;
321 31     private Handler autoFocusHandler;
322 32     private CameraManager mCameraManager;
323
324 33
325 34     private TextView scanResult;
326 35     private FrameLayout scanPreview;
327 36     private Button scanRestart;
328 37     private RelativeLayout scanContainer;
329 38     private RelativeLayout scanCropView;
330 39     private ImageView scanLine;
331
332 40
333 41
334 42     private Rect mCropRect = null;
335 43     private boolean barcodeScanned = false;
336 44     private boolean previewing = true;
337
338 45     public void onCreate(Bundle savedInstanceState) {
339 46         super.onCreate(savedInstanceState);
340 47         setContentView(R.layout.activity_capture);
341 48         setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
342 49         findViewById();
343 50         addEvents();
344 51         initView();
345 52     }
346 53
347 54
348 55     private void findViewById() {
349 56         scanPreview = (FrameLayout) findViewById(R.id.capture_preview);
350 57         scanResult = (TextView) findViewById(R.id.capture_scan_result);
351 58         scanRestart = (Button) findViewById(R.id.capture_restart_scan);
352 59         scanContainer = (RelativeLayout) findViewById(R.id.capture_container);
353 60         scanCropView = (RelativeLayout) findViewById(R.id.capture_crop_view);
354 61         scanLine = (ImageView) findViewById(R.id.capture_scan_line);
355 62     }
356 63
357 64     private void addEvents() {
358 65         scanRestart.setOnClickListener(new OnClickListener() {
359 66             public void onClick(View v) {
360 67                 if (barcodeScanned) {
361 68                     barcodeScanned = false;
362 69                     scanResult.setText("Scanning...");
363 70                     mCamera.setPreviewCallback(previewCb);
364 71                     mCamera.startPreview();
365 72                     previewing = true;
366 73                     mCamera.autoFocus(autoFocusCB);
367 74                 }
368 75             }
369 76         });
370 77     }
371 78
372 79     private void initView() {
373 80         autoFocusHandler = new Handler();
374 81         mCameraManager = new CameraManager(this);
375 82         try {
376 83             mCameraManager.openDriver();
377 84         } catch (IOException e) {
378 85             e.printStackTrace();
379 86         }
380 87
381 88         mCamera = mCameraManager.getCamera();
382 89         mPreview = new CameraPreview(this, mCamera, previewCb, autoFocusCB);
383 90         scanPreview.addView(mPreview);
384 91
385 92         TranslateAnimation animation = new TranslateAnimation(Animation.RELATIVE_TO_PARENT, 0.0f, Animation.REL
386 93             0.0f);
387 94         animation.setDuration(3000);
388 95         animation.setRepeatCount(-1);
389 96         animation.setRepeatMode(Animation.REVERSE);
390 97         scanLine.startAnimation(animation);
391 98     }
392 99
393 100     public void onPause() {
394 101         super.onPause();
395 102         releaseCamera();
396 103     }
397 104
398 105     private void releaseCamera() {
399 106         if (mCamera != null) {
400 107             previewing = false;
401 108             mCamera.setPreviewCallback(null);
402 109             mCamera.release();
403 110             mCamera = null;
404 111         }
405 112     }
406 113
407 114     private Runnable doAutoFocus = new Runnable() {
408 115         public void run() {
409 116             if (previewing)
410 117                 mCamera.autoFocus(autoFocusCB);
411 118             }
412 119         };
413 120
414 121     PreviewCallback previewCb = new PreviewCallback() {
415 122         public void onPreviewFrame(byte[] data, Camera camera) {
416 123             Size size = camera.getParameters().getPreviewSize();
417 124
418 125             // 这里需要计算数据的大小并计算一下,因为相机从摄像头采集的数据
419 126             byte[] rotatedData = new byte[data.length];
417 127             for (int y = 0; y < size.height; y++) {
418 128                 for (int x = 0; x < size.width; x++)
419 129                     rotatedData[x * size.height + size.height - y - 1] = data[x * size.width];
420 130             }
421 131
422 132             // 宽高要调换
423 133             int tmp = size.width;
424 134             size.width = size.height;
425 135             size.height = tmp;
426 136
427 137             initCrop();
428 138             ZBarDecoder zBarDecoder = new ZBarDecoder();
429 139             String result = zBarDecoder.decodeCrop(rotatedData, size.width, size.height, mCropRect.left, mCrop
430 140
431 142             if (!TextUtils.isEmpty(result)) {
432 143                 previewing = false;
433 144                 mCamera.setPreviewCallback(null);
434 145                 mCamera.stopPreview();
435 146                 scanResult.setText("barcode result " + result);
436 147                 barcodeScanned = true;
437 148             }
438 149         }
439 150     };
440 151
441 152     // Mine continuous auto-focusing
442 153     AutoFocusCallback autoFocusCB = new AutoFocusCallback() {
443 154         public void onAutoFocus(boolean success, Camera camera) {
444 155             autoFocusHandler.postDelayed(doAutoFocus, 1000);
445 156         }
446 157     };
447 158
448 159     /**
449 160      * 初始化截取的矩形区域
447 161      */
448 162     private void initCrop() {
449 163         int cameraWidth = mCameraManager.getCameraResolution().y;
450 164         int cameraHeight = mCameraManager.getCameraResolution().x;
451 165

```

```
166  /** 获取布局中扫描框的位置信息 */
167  int[] location = new int[2];
168  scanCropView.getLocationInWindow(location);
169
170
171  int cropLeft = location[0];
172  int cropTop = location[1] - getStatusBarHeight();
173
174  int cropWidth = scanCropView.getWidth();
175  int cropHeight = scanCropView.getHeight();
176
177  /** 获取布局容器的宽高 */
178  int containerWidth = scanContainer.getWidth();
179  int containerHeight = scanContainer.getHeight();
180
181  /** 计算裁剪框的矩形的左上角点x坐标 */
182  int x = cropLeft * cameraWidth / containerWidth;
183  /** 计算裁剪框的矩形的左上角点y坐标 */
184  int y = cropTop * cameraHeight / containerHeight;
185
186  /** 计算裁剪框的矩形的宽度 */
187  int width = cropWidth * cameraWidth / containerWidth;
188  /** 计算裁剪框的矩形的高度 */
189  int height = cropHeight * cameraHeight / containerHeight;
190
191  /** 生成裁剪框的矩形的矩形 */
192  mCropRect = new Rect(x, y, width + x, height + y);
193  }
194
195  private int getStatusBarHeight() {
196  try {
197  Class<?> c = Class.forName("com.android.internal.R$dimen");
198  Object obj = c.newInstance();
199  Field field = c.getField("status_bar_height");
200  int x = Integer.parseInt(field.get(obj).toString());
201  return getResources().getDimensionPixelSize(x);
202  } catch (Exception e) {
203  e.printStackTrace();
204  }
205  return 0;
206  }
```

d 运行效果图



猛犸下载源码示例程序

[好文推荐](#)[关注我](#)[收藏原文](#)



采男的小酒馆

关注 - 3

粉丝 - 120

[+ 加关注](#)

0

0

[0 推荐](#)[0 反对](#)

« 上一篇：人工智能：自动寻路算法实现(四、D、D*算法)
» 下一篇：ZXing开发详解

posted @ 2018-04-03 13:04 采男的小酒馆 阅读(21458) 评论(0) 编辑 收藏 举报

[最新评论](#) [刷新页面](#) [返回顶部](#)

 登录后才能查看或发表评论，立即 [登录](#) 或者 [注册](#) 博客园首页

编辑推荐：
记一次 dump 文件分析历程
图解 | 从网上彻底理解 MySQL 的索引
技术管理进阶 —— 第三个五年，独立参考与落地实操
平时的工作如何体现一个人的技术深度？
革命性创新，动画杀手锏 @scroll-timeline

她的梦想在发光#

HWD科技女性故事有奖征集

活动时间: 2022年3月8日-3月18日



最新资讯
美国To B，大可不必？
湖南工程用中文编写操作系统，还发明了甲、乙、丙语言？？
寒武纪技术骨干梁军离职，1月曾调任CTO
魅族：还在玩小灰字？自动续费自我续命
拼多多消失在聊天框
[» 更多新闻...](#)

公告

昵称：采男的小酒馆
年龄：6年7个月
粉丝：120
关注：3
[+ 加关注](#)

星球精英

她的梦想在发光#

HWD科技女性故事有奖征集

活动时间: 2022年3月8日-3月18日

[马上参与](#)



<	2022年3月							>
日	一	二	三	四	五	六		
27	28	1	2	3	4	5		
6	7	8	9	10	11	12		
13	14	15	16	17	18	19		
20	21	22	23	24	25	26		
27	28	29	30	31	1	2		
3	4	5	6	7	8	9		

搜索

我的标签
.launch(1)
ROS(1)
VS Code(1)

随笔分类
C++(27)
Clion && ROS(2)
CMake(8)
Devices(1)
GCC(5)
github(5)
Matlab(2)
Matplotlib(1)
Matrix(6)
MRPT(2)
Nonlinear Programming(5)
OMPL(5)
OpenCV(3)
PCL(46)
python(5)
更多

随笔档案
2022年3月(2)
2022年2月(2)
2022年1月(4)
2021年12月(1)
2021年11月(3)
2021年10月(1)
2021年8月(3)
2021年7月(1)
2021年6月(4)
2021年5月(4)
2021年4月(5)
2021年3月(3)
2021年2月(1)
2021年1月(7)
2020年12月(9)
更多

阅读排行榜
1. #pragma pack()用法详解(60086)
2. 旋转矩阵、欧拉角、四元数理论及其转换关系(51237)
3. 人工智能: 自动寻路算法实现(四、D、D*算法)(37290)
4. PCD (点云数据) 文件格式(31484)
5. Visual Studio工具 vcpkg简介(28043)

评论排行榜
1. ROS 三维激光数据转化为RangeImage(3)
2. Matlab 摄像头标定+畸变校正(3)
3. Win10 VS2013 suitesparse-metis-for-windows 1.3.1(3)
4. Qt QGraphicsScene中显示网络(2)
5. Visual Studio工具 vcpkg简介(2)

推荐排行榜
1. #pragma pack()用法详解(4)
2. ROS 三维激光数据转化为RangeImage(3)
3. Visual Studio工具 vcpkg简介(3)
4. 旋转矩阵、欧拉角、四元数理论及其转换关系(3)
5. 10. ROS costmap(代价地图)(2)

最新评论
1. Re: 【Boost】boost库中timer定时器 1

thank you	--itfanr
2. Re:Qt QGraphicsScene中显示网格 @Nora_Wu std::vector<QGraphicsLineItem*> gridItemVec...	--采男孩的小蘑菇
3. Re:Qt QGraphicsScene中显示网格 你好，请问gridItemVec的类型是什么？	--Nora_Wu
4. Re:10. ROS costmap代价地图 很详尽，感谢分享！	--修齐
5. Re:人工智能: 自动寻路算法实现(四. D、D*算法) 那么如何将一个CAD建筑平面图转为数据数组来进行路径规划？	--碎碎高同学