ISSN 2616-7433 Vol. 5, Issue 5: 64-68, DOI: 10.25236/FSST.2023.050510

Design and Implementation of Android Barcode Recognition Scheme

Xiaoding Deng

Electronics and Information Engineering, Heyuan Polytechnic, Heyuan, Guangdong, 517000, China

Abstract: *With the popularization of intelligent devices, many application scenarios now obtain corresponding services by scanning barcode. At present, the recognition success rate of some scanning code schemes is low in fuzzy, dark light, long-distance and other environments. By using the barcode and QR code parsing capabilities provided by Huawei's Scan Kit, the APP can realize long-distance scanning and small code amplification scanning, improve the success rate of code scanning in complex scenes such as reflection, dark light, dirt, blur, and cylinder, and thus improve user satisfaction.*

Keywords: *One dimension code; QR code; Scan Kit; Android; Android Studio*

1. Introduction

Now, with the popularization of intelligent devices, many applications use scanning barcode to replace the original manual input, which greatly facilitates users and improves input efficiency. With the increasing application of barcode, there are more and more barcode types, such as one-dimensional EAN-13, Code128, two-dimensional QR Code, PDF417, and Data Matrix. The use scenarios are also becoming more and more complex, such as blurred barcode, multiple barcodes next to each other, long-distance scanning, dim light, reflection, and dirt [1]. In these complex scenarios, the recognition success rate of barcode in many applications is low and has great limitations. At present, the commonly used barcode scanning libraries include Google Play Service, barcodescanner library, ZXing library and ZBar library. Google Play Service depends on Google services, while domestic mobile phones are basically not installed with Google services, so it cannot be used in China. Although the ZXing library does not use Google services, it is also a library launched by Google. Now the West is constantly "decoupled from the economy" and "decoupled from science and technology" from China. It is not good to use this library. It is troublesome to call barcodescanner library and ZBar library. Therefore, this paper chooses another new Huawei Scan Kit [2], which supports scanning of more than 13 mainstream codes and fast and correct scanning in complex scenarios. This improves user satisfaction with applications.

2. Huawei unified code scanning service

Huawei launched the HMS Core to replace Google's GMS. It includes Huawei Machine Learning Service (MLKit), Unified Scan Code Service (ScanKit), Huawei Map Service (MapKit), Huawei Advertising Service (HUAWEI Ads), etc. Thanks to the accumulation of Huawei's capabilities in the field of computer vision, the unified scanning service (ScanKit) can realize the detection and automatic amplification of remote codes or small codes. At the same time, it has made targeted identification optimization for common complex scanning scenes (such as reflection, dark light, dirt, blur, and cylinder) to improve the scanning success rate and user experience [3]. The latest version of the current Android version of Huawei unified scanning service is 2.7.0.302. Scan Kit provides two SDK: com.huawei.hms: Scan and com.huawei hms:scanplus. For Huawei mobile phones and tablets, the functions of the two versions are the same. However, for non-Huawei mobile phones, scanplus is more powerful and can provide an enhanced identification model. Generally, developers choose this SDK, so that applications have better compatibility with devices. The scan version only provides a common recognition model, which is not so powerful, but it is small and suitable for application development that is sensitive to SDK size, such as some wearable smart devices with small memory, children's watches, etc.

3. Application development

3.1 Development Tools

Common Android development tools are eclipse and Android Studio. Eclipse is a commonly used Java development tool, but it is troublesome to develop Android by downloading SDK and configuring environment variables. Android Studio is relatively simple. You do not need to configure environment variables yourself. Therefore, the development tool adopts the Android Studio Chipmunk version. The development language is Kotlin, a static programming language developed by JetBrains for modern multi-platform applications. Its language is relatively simple, and some java codes can be replaced by a line of cotellin. High security, which solves the problem of Java hollow type exceptions. Strong interoperability and compatibility with existing JVM libraries. Android Studio also provides convenience for Kotlin. You can mix Kotlin with java, or directly convert java code into Kotlin code.

3.2 Configure Code Warehouse

Configure the document settings in the Android Studio project Maven warehouse address [4] added to HMS Core SDK in gradle maven {url ' https://developer.huawei.com/repo/ '}.

3.3 Adding Compilation Dependencies

In build Add implementation 'com. huawei. hms: scanplus: 2.6.0.302' to dependencies in the gradle file. If a common recognition model is used, add implementation 'com. huawei. hms: scan: 2.6.0.302'. On Android Studio, enter ctrl shift alt i at the same time, and then enter new to check whether there is a new version of all compilation dependencies. If there is a new version, it can be automatically updated.

3.4 Specific design of barcode recognition software

The new version of Android is increasingly strict in security management. The application needs to set the corresponding permissions at the beginning. Users can also cancel some permissions on application management during use. Barcode recognition requires taking pictures with a camera, and then reading the photographed image for image processing. Therefore, the barcode recognition software requires at least camera and file reading permission. These permissions can be added in the AndroidManifest.xml file. The command is as follows:

```
<!— Add camera permissions -->
<uses-permission android:name="android.permission.CAMERA" />
<!-- File read permission -->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

In addition, although the camera and file reading permission are set in the above file, when the Android version on the intelligent device is equal to or greater than Android 6.0 (23), the user can cancel the corresponding permission at any time. Therefore, at the beginning of each barcode recognition, you need to dynamically verify whether you have permission. If you do not have permission, you need to ask the user to grant the corresponding permission. permission_ Code is a request code, which is a numeric constant used to distinguish different requests when receiving request permission results. The code for dynamic checksum permission request is as follows:

```
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
{requestPermissions(arrayOf(Manifest.permission.READ_EXTERNAL_STORAGE,Manifest.permissi
on.CAMERA), permission_ code);}
```

After executing dynamic verification and requesting permission. The application waits and receives the results of dynamic request and permission verification through the onRequestPermissionsResult function. If you have the permission to view the camera and read the external storage, you can continue to execute. Otherwise, the application should report an error and exit the current barcode recognition. If the application continues to execute without permission, the program may crash. Through grantResults Size to determine whether the number of received permissions is 2, because the permissions we applied for earlier are only camera and file read permissions. It is worth noting that the number of permissions here is the number of permissions dynamically applied by the requestPermissions function, not the number of permissions in the AndroidManifest.xml file. The number of permissions dynamically requested can be less than or equal to the number of permissions in the AndroidManifest.xml file. For example, your application is in Android Manifest. The xml file may have three permissions: network, camera and file reading, but now the application only needs two permissions: camera and file reading for barcode recognition. In addition to determining the number of permissions, it is also necessary to determine whether the two permissions in grantResults are PackageManager PERMISSION_ GRANTED. At the same time, check whether the request code is a dynamic check and the request code when requesting permission: permission_ code. Call ScanUtil StartScan (activity, scan_code, null) Start a ScankitActivity to scan the barcode for recognition, scan_ Code is the request code for starting recognition, and is a numeric constant. At this time, the phone will pop up the camera photo interface. The code is as follows:

```
    override fun onRequestPermissionsResult(requestCode:Int,permissions:Array<out String>,
grantResults:IntArray) {
super.onRequestPermissionsResult(requestCode, permissions, grantResults)
if (requestCode == permission_code && grantResults.size == 2 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED && grantResults[1] ==
PackageManager.PERMISSION_GRANTED) {
ScanUtil.startScan(activity, scan_code, null)}
}
```

The recognition result of code scanning is received by rewriting the onActivityResult function. First, judge that the request code should call ScanUtil Request code when startScan: scan_ code. At the same time, judge whether the return code is successful and whether the returned value is null. The barcode recognition result is saved in ScanUtil.RESULT, Scan Util. The RESULT type is a HmsScan object, which is a unified class returned by Scan Kit. It contains the coordinates of the barcode in the input image, the original data of the barcode, the format of the barcode, structured data, zoomValue and other information. Get the original barcode value through getOriginalValue(); Get the barcode format through getScanType(); Get the structured data of barcode through getScanTypeForm(); The position of the barcode in the picture can be obtained through getBorderRect(). The codes receiving the scanning identification results are as follows:

```
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
if (resultCode != RESULT_OK || data == null) {
return}
else if (requestCode == scan_code) {when (val obj: Any =
data.getParcelableExtra(ScanUtil.RESULT)!!) {
is HmsScan -> {
if (!TextUtils.isEmpty(obj.getOriginalValue())) {
binding!!.SN?.setText(obj.getOriginalValue())
binding!!.loading.visibility = View.VISIBLE
QueryAllData()
}return}}}}
```

During debugging, it is found that ScanUtil After startScan scans the code, onActivityResult has not returned any data. I used to call ScanUtil in Fragment startScan returns the onActivityResult in the activity first. In order to receive barcode recognition results in Fragment, you need to rewrite the onActivityResult function in activity. When rewriting, be sure to add super OnActivityResult (requestCode, resultCode, data), and then iterate over each fragment to forward the onActivityResult. Therefore, each fragment will receive the onActivityResult results of all fragments. Therefore, the request code: permission_ Code is very important. It is the only basis to distinguish different fragments. The code is as follows:

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
super. onActivityResult(requestCode, resultCode, data)
supportFragmentManager.fragments
if (supportFragmentManager.fragments.size > 0) {
val fragments = supportFragmentManager.fragments
for (mFragment in fragments) {
mFragment. onActivityResult(requestCode, resultCode, data) }}}
```

In this way, the onActivityResult in the Fragment can receive the result after the code scanning identification.

4. Application optimization and expansion

4.1 Optimization of code scanning speed

In order to improve the scanning recognition speed of the application, certain code types specific to the code can be set in the parameters to improve the bar code recognition speed. For example, the code types commonly used in our daily life are commodity barcode EAN-13, QR Code and 128 code. We can set parameters as follows: val options=HmsScanAnalyzerOptions. Creator(). setHmsScanTypes (HmsScan. QRCODE_SCAN_TYPE, HmsScan. CODE128_SCAN_TYPE, HmsScan. EAN13_SCAN_TYPE) create().

4.2 User defined code scanning interface

Sometimes, we need to add a title and set the barcode scanning area in the barcode scanning interface. Then we can use the Customized View mode of the unified scanning service. In this mode, we can customize the title, return button, flash button, code scanning interface and code scanning border. So as to meet the personalized requirements of the application.

4.3 Image scanning

In life, we often encounter such a scenario: someone who is not around sends you a payment code to make payment, and the company notifies you of a QR code for information statistics. These codes cannot be scanned when there is only one mobile phone. It can only be identified by this QR code image. Bitmap mode of unified scanning service can realize image scanning.

5. Result verification

In Figure 1, there are horizontal and vertical lines in the dirty QR code. After testing, the application can quickly and accurately identify them.

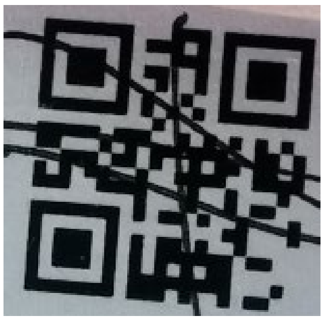

Figure 1: Dirty QR code

In Figure 2, there is a large piece of strong reflection in the QR code, which can be quickly and accurately identified by the application through testing.

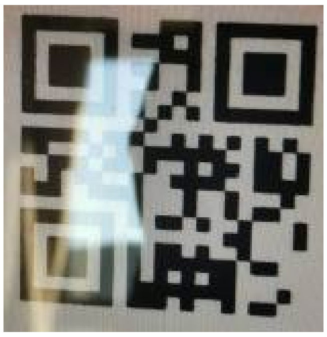

Figure 2 Reflective QR code

With the Huawei P30 mobile phone, the QR QR code with a size of 3.3cm * 3.3cm can be accurately identified by focusing and scanning at a distance of 5.6m. The two-dimensional code can also be accurately recognized in the dark environment.

6. Conclusion

Through development and testing, it is verified that the Android barcode recognition scheme based on Huawei's Scan Kit can really deal with complex scenarios such as barcode blurring, multiple barcodes being close together, long-distance scanning, dim light, reflection, and dirt. Compared with American technologies such as Google Play Service and ZXing library, Huawei's unified code scanning service is more "secure" and is not afraid of American bottlenecks. Compared with barcodescanner library and ZBar library, developers can also easily call the unified scanning service and customize the personalized scanning interface to save development costs and improve application quality.

Acknowledgment

Science and Technology Plan Project of Heyuan City in 2020 (Heke 166).

References

[1] Tian Z, Wang M, Liu Z, Guo R, Lv K. Research and design of two-dimensional code recognition system based on intelligent display line [J]. Manufacturing Automation, 2022, 44 (02): 159-163
[2] Du G, Liu Y, Yang X, Wang J. Interactive query of equipment information based on Huawei unified code scanning service programming [J]. Computer Measurement and Control 2021; 29 (06): 147-152+168. DOI:10.16526/j.cnki.11-4762/tp. 2021.06.030
[3] Huawei. 2022.08. Business Introduction - Unified Code Scanning Service, Huawei Developer Alliance from https://developer.huawei.com/consumer/cn/hms/huawei-scankit
[4] Chen F. Android Studio Application [J]. Computer Knowledge and Technology, 2014, 10 (24): 5659-5661+5666. DOI: 10.14004/j.cnki.ckt. 2014; 0028.