华南农业大学

课程大作业实验报告

基于 Android 平台的雾滴喷施分布分析系统

课程名称：数字图像处理

组　长：张凯旋　学号：201721010325　年级专业班级：17 自动化 3 班

组员一：林钊圣　学号：201721010311　年级专业班级：17 自动化 3 班

组员二：翁健豪　学号：201721010321　年级专业班级：17 自动化 3 班

大作业主要分工

张凯旋：图像预处理、雾滴沉积分析处理

林钊圣：智能裁剪（神经网络识别）、App 功能框架

翁健豪：App 界面设计、数据流分析

报告提交日期　2019 年 12 月 05 日

摘　　要

雾滴图像分析是检测农药喷雾效果的重要技术，主要使用水敏纸作为雾滴收集器代替叶片承接雾滴，人工采集并做好密封措施，再带回到实验室中使用专业的系统设施对水敏纸中雾滴进行分析统计。对水敏纸须密封包装以防止污染，否则会导致测量结果出现较大误差。为了减少人为造成的污染，及时方便地在田间测量雾滴分布情况，本文开发了一款基于 Android 平台的雾滴喷施分布分析仪。

基于 Android 平台的雾滴喷施分布分析仪充分利用 Android 移动端平台的便捷性以及可移植性，对田间采样后的水敏纸进行实时的图像采集，同时通过基于神经网络的智能识别功能解决传统方案中移动端拍摄中必须与纸张持平的问题，随后对雾滴图像进行相应的数据分析，测量计算雾滴图像中的雾滴数量、最小包围圆的直径等参数，并根据参数分析雾滴粒径、覆盖率及变异系数等雾滴分布信息。通过模拟测试和实际测试，本文设计的雾滴分析仪有较高的测量精度。

关键词：　　安卓应用　OpenCV　　雾滴图像处理　　神经网络

目录

1 前言

1.1 课题的研究背景

农药的不合理使用容易导致农作物产量降低，主要因为农药的利用率低，在施用过程中大部分农药受植保无人机喷洒过程中的环境因素如风力、阻力等影响导致飘移，部分流失进入地表径流，降低有效利用率的同时，还容易污染土壤和水源，影响生态环境。同时极低的农药利用率容易造成农药的滥用，极大增加了农药的残留，严重威胁着农作物的质量安全和人类的健康。

因此需要对农药的使用量进行严格的把控，而农药雾滴图像中雾滴的分布情况，是分析农药喷洒效果的重要参考指标。目前来说，雾滴分布情况的分析主要是基于数字图像的分析技术，与人眼相比起来，数字图像分析技术在处理速度和识别精确度来说有了显著的提高，但是，在前期的叶片采样上，仍需要花费大量的人力、时间和精力。

为了节省在叶片采样、包装以及转移所耗费的时间、精力和人力，设计开发了一款雾滴喷施分布分析仪 App，能够对取样后的水敏纸进行实时雾滴检测。同时利用 Android 手机的便利性实现拍照、截图、裁剪、保存等功能，接入 OpenCV 库对图像进行分析，并在 Android 端开发相应的功能，将图像分析的结果以文本方式进行展示和保存，能够有效减少劳动力浪费和节省时间成本，提高水敏纸采样和图像分析效率。

1.2 雾滴图像分析的基本原理

App 通过手机拍照或者读手机内存中的图像，然后对图像进行二值化、滤波等一系列的图像预处理之后利用图像分析算法求得雾滴在像素域的一些基本参数例如：粒径大小、雾滴面积等，然后通过输入水敏纸实际的长宽物理尺寸来标定雾滴粒径的实际大小，最后再做雾滴分析处理求得雾滴覆盖率、雾滴变异系数等一系列与雾滴相关的信息。

1.3 项目目标和技术路线规划

在 Android 手机上开发一款雾滴处理的 App，能够对收集农药雾滴的水敏纸卡拍摄图像，并分析图像中包含的雾滴数据信息，这些信息包括雾滴数量、雾滴粒径（即雾滴直径）、雾滴覆盖率、雾滴变异系数等。

首先选定基于 Android 平台的雾滴喷施分布分析系统项目，对项目有所了解后与老师交流确定软件具体的功能需求，接下来就是 Android 平台、雾滴分析技术以及图像处理方法等相关资料的查阅以及学习，并根据查阅的资料对应用的整体架构进行构思，对框架进行选择，确认好框架以及架构后即可开始界面的设计以及功能的编写。完成基础的功能以及界面后进行初步测试，测试成功后与老师进行后续功能的讨论。再将功能进行完善并改善界面，使界面变得更为美观，然后进行单元测试以及代码调试，不断优化代码，最后在真机上运行查看最终的效果是否理想，如图 1 所示。

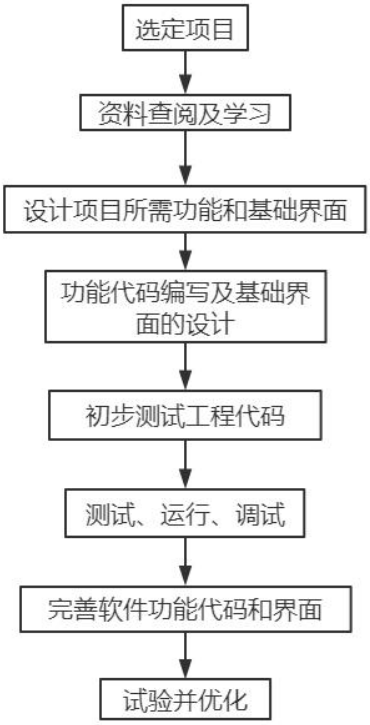


图 1　技术路线

2 Android 应用程序开发

2.1 Android 概述

Android 是一个基于 Linux 内核的开源操作系统，主要运行在智能手机等移动设备上。对于 Android 开发人员而言，Android 提供了一套完整的应用程序开发规范，开发者根据规范使用 Android 支持的 Java、Kotlin 等语言进行程序编写，开发完成后即可在不同的 Android 移动设备上运行。

2.2 Android 开发工具

2.3.1 开发环境 IDE

有 Eclipse 和 Android Studio 两种，本文使用的是 Android Studio。

Android Studio 是一款基于 IntelliJ IDEA 的 Android 集成开发工具，为 Android 的开发和调试提供了集成的 Android 开发工具。

作为一款优秀的编辑器，Android studio 提供了代码重构和快速修复、支持 UI 拖曳和预览布局、支持 gradle 编译构建、支持常用模板和组件快速生成等功能。

2.3.2 软件开发工具包 Android SDK

Android SDK（Android Software Development Kit）是安卓工程师用于在 Android 平台上创建应用程序的专属 Android 开发工具包，为应用程序设计语言 Java 和 Kotlin 提供了相应的 API（Application Programming Interface，应用编程接口）。并提供兼容 Windows/Linux/Mac 三大操作系统的 Android 移动应用开发组件，方便开发者使用 Java 或 Kotlin 编程语言开发应用程序，将编写完成的应用编译打包成 apk 文件，快速运行在模拟器或者 Android 移动设备上进行代码调试以及效果查看。

2.3.3 构建工具 Gradle

Gradle 是一个基于 Apache Ant 和 Apache Maven 概念的项目自动化建构工具。其功能优势主要有以下几点。

（1）多语言构建：支持多种语言的构建，包括了 Java、Scala、Python、C/C++、Android、IOS 等。

（2）工具集成：第三方插件库及各种 IDE 集成。

（3）构建报告：强大的构建分析功能。

（4）简明强大的逻辑：约定优于配置，没有过多的限制。

（5）强大的依赖管理：可以跨约多个存储库处理依赖传递包括 Maven 库、Ivy 以及本地文件库。

2.4 Android 应用程序框架设计

基于 Android 喷雾雾滴分析仪的应用开发为应用层程序开发，在 Android 应用层上接入 OpenCV 库，通过使用 OpenCV4Android 库中提供的相关图像处理的 Java 接口，并结合 Android SDK 中提供的应用程序接口 API，对雾滴分布数字图像进行相应的图像分析处理。

低量雾滴分析 App 主要包含三个部分：分别是图像采集、图像处理以及雾滴分析，如图 2 所示。其中图像采集主要通过两种方式：相册图片选择、相机拍摄，两者获取到的图片均需经过裁剪。图像处理主要为图像预处理部分，对打开的图像实现二值化、闭运算、中值滤波、轮廓检测以及最小外边框绘制预处理，为后面的雾滴分析做准备。雾滴分析中，主要对预处理后的图像进行分析，得出雾滴图像中的雾滴轮廓数量、轮廓面积以及最小外边距的宽高，通过分析得出的这些数据可以计算相应的雾滴密度、沉积分布等。

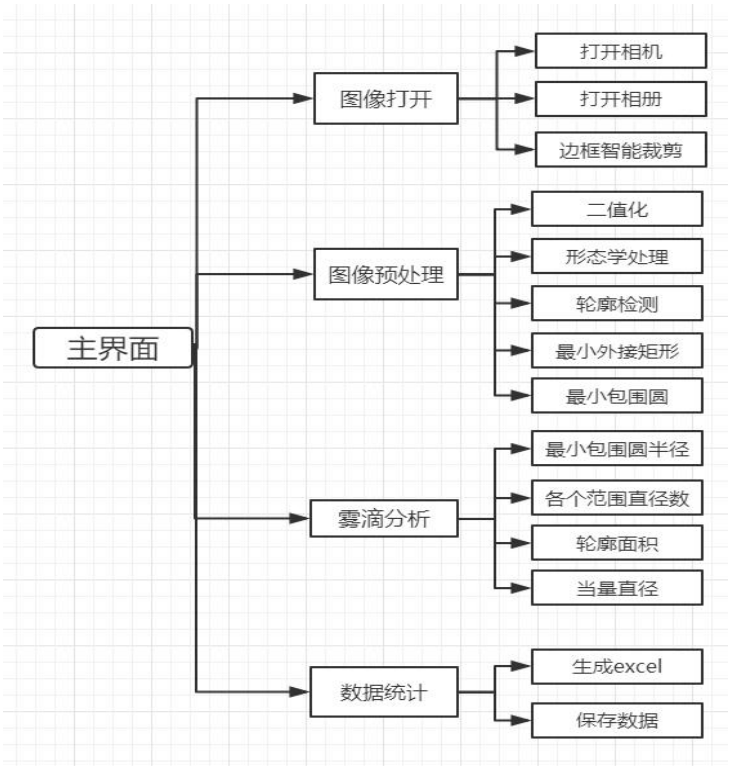


图 2 框架设计

3 雾滴分析 App 的设计

根据实际需求，雾滴分析仪应具备图像采集、图像预处理、雾滴分析、导出数据等功能，本文按功能要求设计了主界面和整体流程。

3.1 主界面

将雾滴分析应用安装到手机后，启动应用，主界面如图 3 所示。

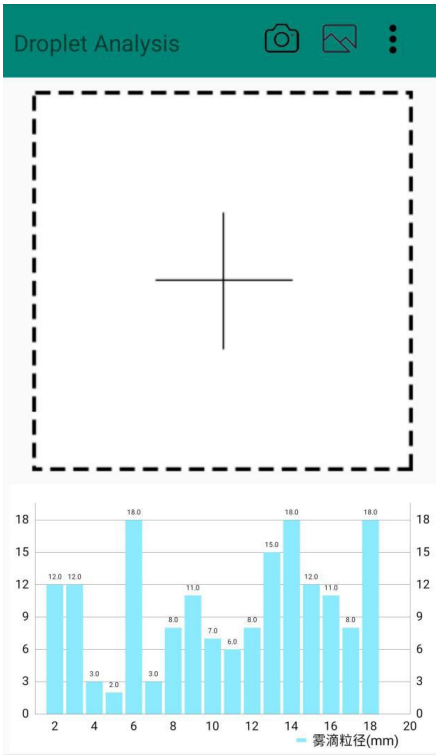


图 3 App 主界面

3.2 整体流程设计

从启动应用开始，用户可选择是否打开图片，如是，可以通过相册和拍照两种方式获取需要分析的图片，在 App 界面的标题栏点击相册图片可以打开系统相册选择图片，点击相机图片可以启动系统相机进行拍摄并按照自己的需要从拍摄的图片中裁剪出部分区域进行分析。如否，则显示默认的测试图片。

打开相应的图片后，用户即可在图像处理的区域中对图像进行一键分析，其中该功能中包含对图片进行二值化、闭运算、中值滤波、轮廓检测以及最小边框绘制等预处理。经过预处理后的图片可以在雾滴分析区域中点击数据分析求出相应的雾滴分析数据，如图 4 所示。

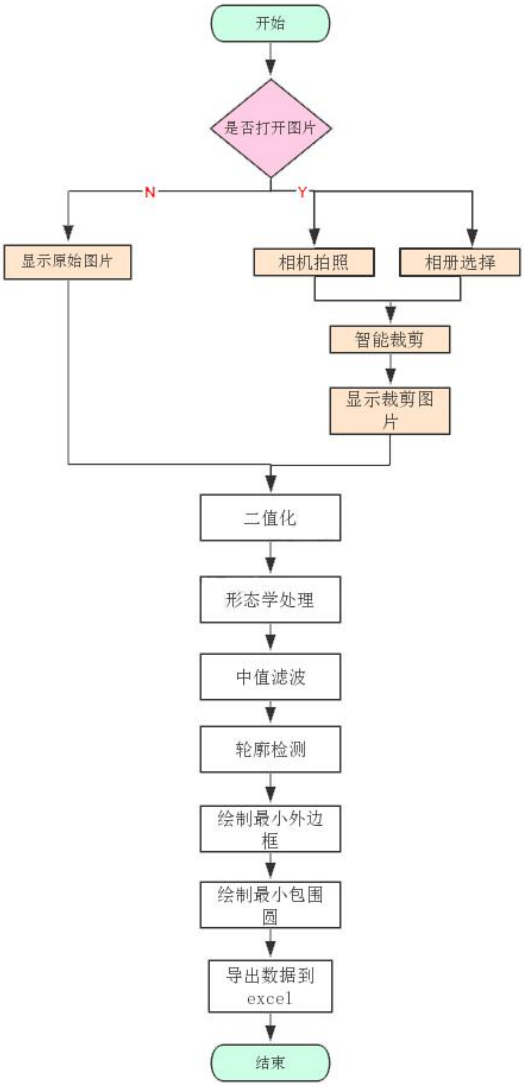


图 4 整体流程设计图

3.3 OpenCV 库的配置以及接入

3.3.1 OpenCV 概述

OpenCV 是一个支持 Linux、Windows、Android 和 Mac OS 等操作系统的跨平台计算机视觉库，实现了图像处理和计算机视觉方面的很多通用算法，并为开发者提供了 Java、MATLAB、Python 等语言的接口。

3.3.2 Android 平台接入 OpenCV

OpenCV 官网上提供了封装好的 Java 依赖库版本，可以直接从网上将相应版本的

Java 库下载到本地，将有需要的部分导入到 Android IDE 中，并在 Android IDE 的 Project 配置中导入 OpenCV 提供的 Java 依赖，最后对 build.gradle 文件进行相应的配置后即可使用 OpenCV 提供的 Java 接口进行编程。

缺点在于配置比较繁琐，需要注意的地方比较多，容易因为不够细心漏掉某些细节导致导入失败，程序报错等问题。而且完成配置后还需要下载 OpenCV 的管理应用 OpenCV Manager 才能运行编写的程序。使用一个应用还需要另外一个管理应用才能启动，这对于用户来说是很麻烦的，这对于本来为了便捷简约的移动应用来说是致命的。下面我们介绍使用一种新的方法避免这种方法带来的缺陷。

3.3.3 Android Studio 接入 OpenCV4Android 库

（1）下载 OpenCV 的 Android 支持包。

在 OpenCV 官网的 OpenCV4Android 支持库下载页面中，有很多不同版本的 Android 支持包。可以根据自己的需要选择不同的版本进行下载，本文选用的是 3.4.8 版本。下载完成的支持库是压缩包的形式，解压完成后可以看到目录下有两个配置文件以及 sdk 等三个文件夹，如图 5 所示。

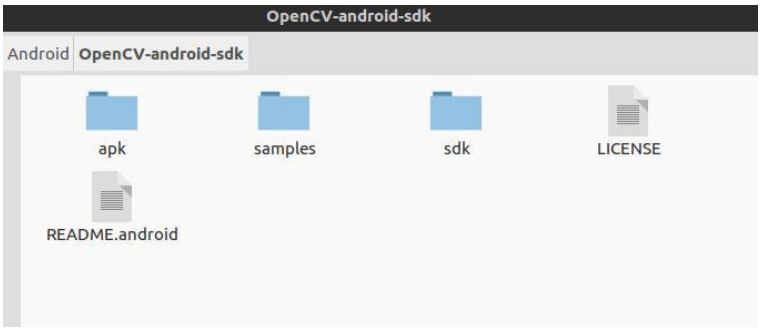


图 5 OpenCV4Android 压缩包解压后的文件

其中 apk 文件夹下提供的是相应版本的 OpenCVManager，samples 文件夹下提供的是 OpenCV 官方的一些例子，方便开发者进行学习，sdk 文件下的才是我们需要的。在 sdk 文件下有 etc、java、native 三个文件夹和一个 build.gradle 文件。其中 java 文件夹中的内容是最终需要导入到 Android 应用程序中 OpenCV 封装的 Java 库（后文简称为 OpenCV4Android 库），记录下这个 Java 库所在路径，下面的步骤需要通过这个路径将 OpenCV4Android 库导进来。

（2）Android Studio 引入 OpenCV4Android 库。

在 Android Studio 中新建一个工程，全部为默认设置即可。在新建好的工程中，

点击“File --> New --> Import Module”导入 OpenCV4Android 库，然后选择 OpenCV 库的路径。Module name 可以自由定义，本文自动识别为 OpenCVLibrary348，然后 Next --> Finish 即可。

（3）修改模块 OpenCVLibrary348 中 build.gradle 文件的配置。

OpenCVLibrary341 模块的 build.gradle 文件中的 compileSDKVersion、builtToolsVersion、minSDKVersion、targetSDKVersion 几个版本号需要修改为与 app 模块的 build.gradle 文件中的版本号一致，本文的 app 模块下 build.gradle 文件的 compileSDKVersion、builtToolsVersion、minSDKVersion、targetSDKVersion 分别为 28、28.0.3、19、28。修改后 OpenCVLibrary348 中 build.gradle 文件内容如下所示：

```
android {
    compileSdkVersion 28
    buildToolsVersion '28.0.3'
        defaultConfig {
            minSdkVersion 19
            targetSdkVersion 28
        }
}
```

（4）为 app 主模块添加 OpenCVLibrary348 模块依赖。

点击“File --> Project Structure”打开项目结构。选中 app 模块，选择标签栏的“Dependencies”，点击右边“+”出现下拉框，在下拉框中选择“Module dependency”，即可看到 OpenCVLibrary348，选中 OpenCVLibrary348 后完成整个 OpenCV4Android 库的导入以及配置，如图 6 所示。

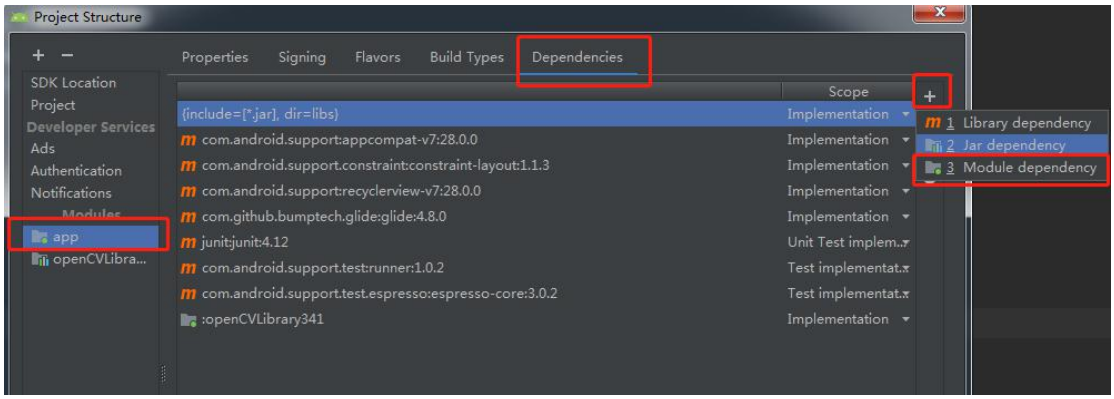


图 6 导入 OpenCVLibrary348 模块的 Java 依赖

（5）设置免安装 OpenCVManager。

OpenCVManager 保存了为 Java 库提供底层支持的 libOpenCV_java3.so 库，因此我们只要手动地将 libOpenCV_java3.so 这个库添加到 Android 应用程序后即可实现相同的效果，就不需要额外再下载 OpenCVManager 管理应用了。

libOpenCV_java3.so 放在图 7 中 sdk 文件下的 native/libs 文件夹，点开可以看到 armeabi-v7a、armeabi、arm64-v8a、x8 等多个不同的架构，每个架构文件夹下就是对应架构的 libOpenCV_java3.so。一般来说，市面上的 Android 手机基本都可以兼容 armeabi-v7a 架构，因此我们将 armeabi-v7a 文件夹下的 libOpenCV_java3.so 导入到应用中即可。为了兼容性的问题也可以全部导入进去，但考虑到应用体积优化的问题，因此在低量雾滴分析应用中只加入了 armeabi-v7a 架构的 libOpenCV_java3.so。

Android Studio 中切换到 Android 视图，在 app 目录下创建一个 jniLibs 文件夹，并在 jniLibs 下创建 armeabi-v7a 文件夹，将 libOpenCV_java3.so 导入到 armeabi-v7a 文件夹中，如图 7 所示。

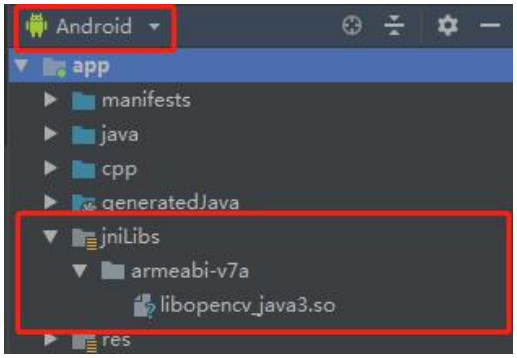


图 7　Android 视图下导入 libOpenCV_java3.so

最后在 MainActivity.class 文件中将 libOpenCV_java3.so 加载进来，代码如下所示：

```
static {
    System.loadLibrary("OpenCV_java3");
}
```

需要注意的是加载库的名称为“OpenCV_java3”，前缀“lib”和后缀“.so”都需要去掉。配置完成后即可在应用程序的工程下使用 OpenCV4Android 库中提供的 API。

4 雾滴分析仪 App 的功能

雾滴分析仪开发的功能主要有四个部分，分别为照片输入功能、图像预处理功能、雾滴分析功能以及分析报告生成功能。

4.1 照片输入功能

照片输入功能主要包含开启相机拍照、相册图片选择、图片智能裁剪和图片显示四部分。

4.1.1 开启相机拍照

（1）功能展示

拍照功能由 Android 应用层开发，点击雾滴分析仪导航栏中的相机图标，如图 8 所示。

图 8　相机图标

点击图标后即可启动 Android 手机端的系统相机进行拍摄，如图 9 所示。

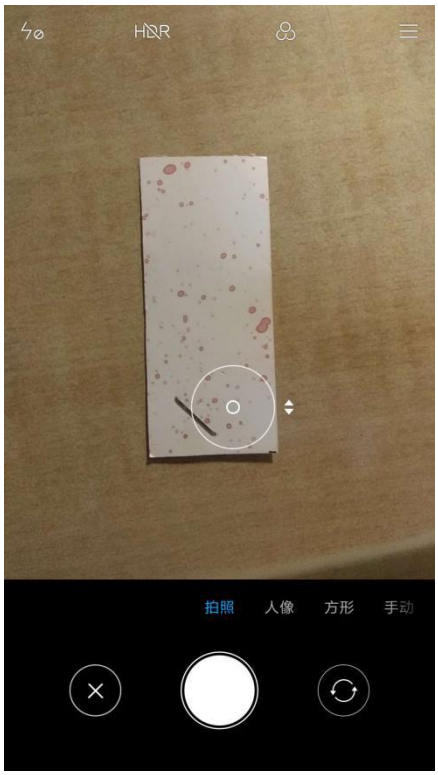


图 9　App 拍照

（2）功能实现

开启系统相机的功能主要通过 Android API 中的 File、URI（Uniform Resource Identifier）和 Intent 类来实现。

URI 是统一资源定位符。在 Android 7.0 以下，通过调用 URI 的 fromFile(File file)方法可以获取到参数中 file 文件对应的资源引用 URI，用来定位到图片拍摄后放置的文件位置。而在 Android 7.0 以上加入了文件保护的功能，需要通过 FileProvider 类的 getUriForFile（File file）方法来获取 file 对应的 URI。考虑到 Android 版本的兼容性，这里封装为 getMediaFileUri 方法，代码如下所示：

首先声明 File 文件的位置，通过 getMediaFileUri 方法获取到 mediaFile 文件对应的 imageUri。然后声明一个调用系统拍照功能的 takePhotoIntent，将对应的 imageUri 传入到 takePhotoIntent 的 putExtra 方法中进行传递。最后将 takePhotoIntent 传入到 startActivityForResult 中实现启动系统相机的拍照功能，这里封装为 OpenCamera 方法，代码如下所示：

```
public void openCamera(Activity activity) {
    int currentapiVersion = Build.VERSION.SDK_INT;// 获取系統版本
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);// 申请调用系统相机
    // 判断存储卡是否可以用，可用进行存储
    if (hasSdcard()) {
      //设置一个日期格式
      SimpleDateFormat timeStampFormat = new
SimpleDateFormat("yyyy_MM_dd_HH_mm_ss");
      //图片的命名为日期时间
      string_camera = timeStampFormat.format(new Date());
      tempFile = new File(Environment.getExternalStorageDirectory(),string_camera + ".jpg");
      //判断系统版本号,兼容 Android 7.0
      //如果系统版本号小于 Android 7.0
      if (currentapiVersion < Build.VERSION_CODES.N) {
        // 从文件中创建 uri
        imageUri = Uri.fromFile(tempFile);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
      } else {
        //当系统版本号大于 Android 7.0 使用共享文件的形式
        ContentValues contentValues = new ContentValues(1);
        contentValues.put(MediaStore.Images.Media.DATA, tempFile.getAbsolutePath());
        //检查是否有存储权限，以免崩溃
        if (ContextCompat.checkSelfPermission(this,
```

```
Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    != PackageManager.PERMISSION_GRANTED) {
                //申请内存写权限
                Toast.makeText(this, "请开启存储权限", Toast.LENGTH_SHORT).show();
                return;
            }
            imageUri =
activity.getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
contentValues);
            intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
        }
    }
```

处理事件：

```
if (resultCode == RESULT_OK) {
    Intent Camera_intent=new Intent(MainActivity.this,SmartCrop_Activity.class);
    Camera_intent.setData(imageUri);
    startActivityForResult(Camera_intent,PHONE_CODE_CUT);
}
```

4.1.2 选择相册照片

（1）功能展示

拍照功能由 Android 应用层开发，点击雾滴分析仪导航栏中的相册图标，如图 10 所示。

图 10　App 相册图标

点击图标后即可启动 Android 手机端的系统相册进行图片选择，如图 11 所示。

图 11　打开相册

（2）功能实现

开启系统相机的功能主要由相册图标启动，根据图标接收到的点击事件触发。启动部分代码如下：

```
public void openAlbum() {
        Intent intent = new Intent(Intent.ACTION_PICK);
        intent.setType("image/*");
        // 开启一个带有返回值的 Activity，请求码为 GET_PHOTO
        startActivityForResult(intent, CODE_GET_PHOTO);
}
```

处理事件：

```
if (resultCode == RESULT_OK) {
      if (data != null) {
          Uri uri = data.getData();
          imageUri = uri;
      }
      Intent Camera_intent=new Intent(MainActivity.this,SmartCrop_Activity.class);
      Camera_intent.setData(imageUri);
      startActivityForResult(Camera_intent,PHONE_CODE_CUT);
}
```

4.1.3 图片裁剪

（1）功能简介以及设计流程

裁剪功能在拍照或相册照片选择完成后触发，可由用户按需对图片实现部分区域截取，便于对截取部分图片进行针对性的分析。在设计这个功能模块的时候我们迭代设计了三个版本，下面详细介绍每个版本的特点。

（2）系统自带裁剪模块

裁剪功能由 Android 应用层开发，根据相机拍照后返回的回调结果来决定是否开启裁剪功能。在这一版本中用户可以拖动裁剪框对图片进行区域裁剪，实现功能如图 12 所示。

图 12　裁剪功能版本一 App 界面

回调结果返回在 onActivityResult 中处理，代码如下：

```
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data){
          super.onActivityResult(requestCode,resultCode,data);
    switch(requestCode){
          case CODE_TAKE_PHOTO:
          if(resultCode == RESULT_OK){
           cropPhoto();
          }
          break;
}
```

其中 cropPhoto 方法实现裁剪功能。代码如下：

```
private void cropPhoto() {
    File cropFile = new File(getExternalCacheDir(),"crop_image.jpg");
    cropUri = getMediaFileUri(cropFile);
    Intent cropIntent = new Intent("com.android.camera.action.CROP");
          cropIntent.setDataAndType(imageUri,"image/*");
    cropIntent.putExtra("crop",true);
    cropIntent.putExtra("return-data", false);
    cropIntent.putExtra("outputFormat", Bitmap.CompressFormat.JPEG.toString());
    if (Build.VERSION.SDK_INT >=  Build.VERSION_CODES.N) {
     cropIntent.setFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
     cropIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
     List<ResolveInfo> resInfoList =
    getPackageManager().queryIntentActivities(cropIntent,PackageManager.MATCH_DEFAULT_
ONLY);
     for (ResolveInfo resolveInfo : resInfoList) {
       String packageName = resolveInfo.activityInfo.packageName;
       grantUriPermission(packageName, cropUri,
    Intent.FLAG_GRANT_WRITE_URI_PERMISSION |
```

```
        Intent.FLAG_GRANT_READ_URI_PERMISSION);
        }
}
```

（3）uCrop 裁剪库

uCrop 是一个让你可以裁剪图片以进一步使用的安卓库。主要功能包括：

1）缩放图片

2）旋转图片

3）改变裁剪的宽高比

4）支持触摸手势：单手指滚动和平移图片，双手指旋转图片，捏图变焦（放大缩小），双击变焦。

5）功能多样化的简便 Activity，有精确化缩放和旋转的空间以及一套预定义的宽高比例（1:1, 4:3, 3:4, 2:3, 3:2, 16:9, 9:16 +原始图片的比例）。

实现功能如图 13。

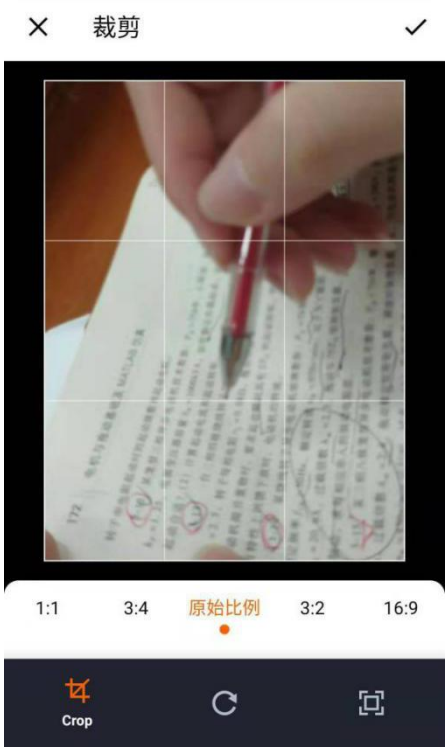


图 13　裁剪功能版本二—APP 界面

使用方法：

在项目中引入库：compile 'com.yalantis:ucrop:1.0.1'

在 AndroidManifest.xml 中添加 UCropActivity：

```
<activity
android:name="com.yalantis.ucrop.UCropActivity"
android:screenOrientation="portrait"/>
```

uCrop 的配置是使用的 builder 模式：

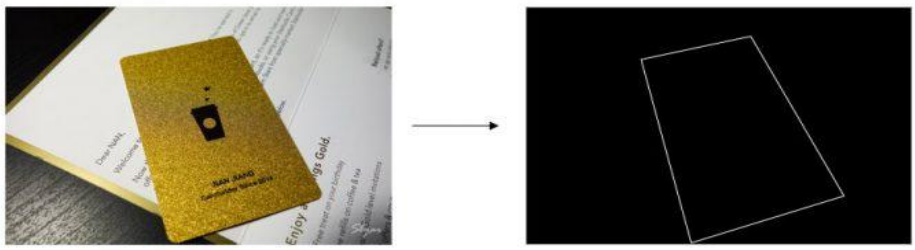
```
UCrop.of(sourceUri, destinationUri).withAspectRatio(16, 9)
.withMaxResultSize(maxWidth, maxHeight).start(context);
```

重写 onActivityResult 方法并处理裁剪的结果：

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK || requestCode == UCrop.REQUEST_CROP) {
        final Uri resultUri = UCrop.getOutput(data);
    } else if (resultCode == UCrop.RESULT_ERROR) {
        final Throwable cropError = UCrop.getError(data);
        }
}
```

（3）基于机器学习 HED 网络优化的 SmartCrop 裁剪

最初 SmartCrop 是通过 OpenCV 的 Canny 算法识别出照片的边缘线条，然后进行后续处理的，我们理想中的 Canny 算法效果应该是这样的，输入一张图片能精准的识别出我们想要的边缘线条，后续再配合 OpenCV 的线条检测功能可以很容易得识别出目标物体的位置：



如果背景稍微复杂一点就会夹杂着很多非识别物体的边缘线条， Canny 算法并不能很好的提取出我们想要的边框，对后续的处理提出了很大的挑战。OpenCV 这种传统算法很快会进入到识别瓶颈，机器学习是一条新思路。

SmartCropper 是一个智能图片裁剪框架，能够智能识别图片中的文档边框，自动瞄点选中文档边框，用户只需按下裁剪即可获得选取区放大图片。支持特性：

1）使用智能算法识别图片中的边框

2）支持拖动锚点，手动调节选区

3）使用透视变换裁剪并矫正选区

智能选区：

透视变换裁剪并矫正选区：

使用方法：

1、根目录下的 build.gradle 添加

```
allprojects {
        repositories {
            ...
            maven { url 'https://jitpack.io' }
        }
}
```

2、添加依赖

```
dependencies {
    compile 'com.github.pqpo:SmartCropper:v2.1.3'
}
```

3、裁剪布局

```
<me.pqpo.smartcropperlib.view.CropImageView
    android:id="@+id/iv_crop"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

4、设置待裁剪图片

```
ivCrop.setImageToCrop(selectedBitmap);
```

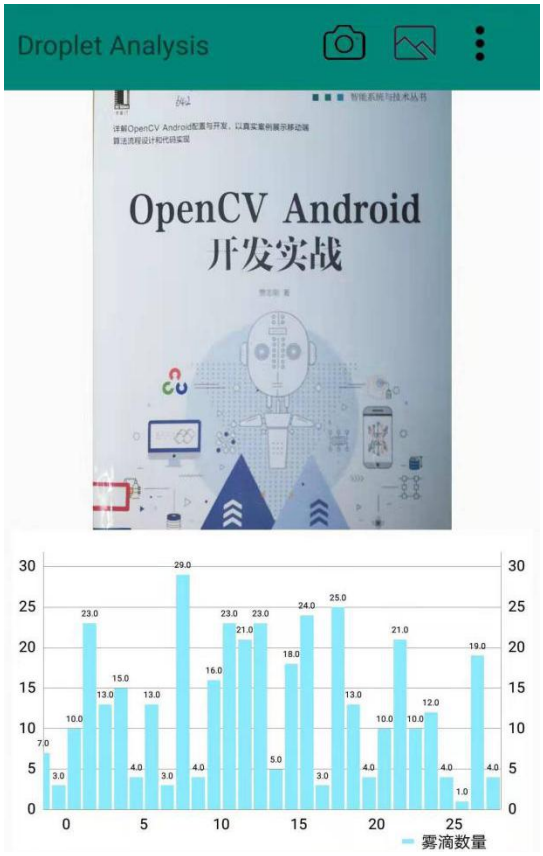
5、裁剪选区内的图片

```
Bitmap crop = ivCrop.crop();
```

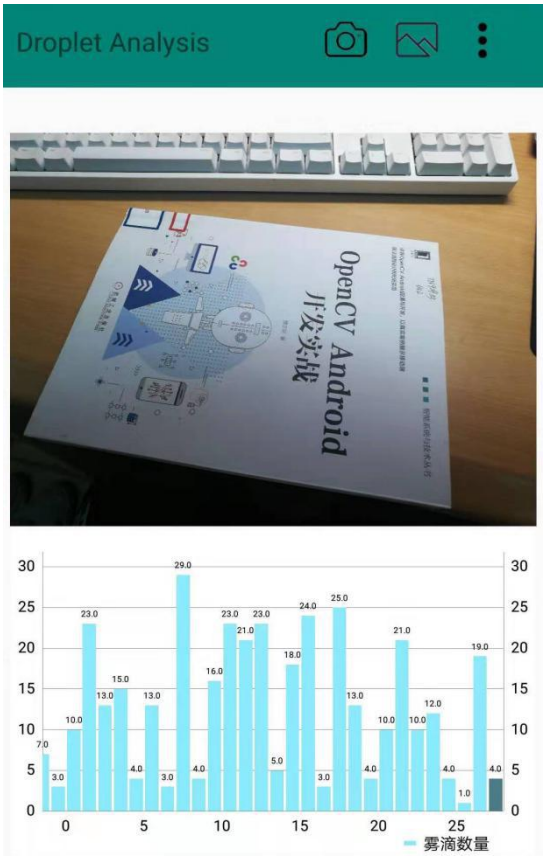
4.1.4 图片显示

（1）功能展示

图片显示功能由 Android 应用层开发，主要将拍照裁剪后的照片或从系统相册选择的照片显示在 App 主界面的显示区域，裁剪图片和相册选择图片分别如图 14（a）（b）所示。



（a） 显示裁剪图片



（b） 显示相册图片

图 14 裁剪及相册图片显示

（2）功能实现

通过 Android API 中的 BitmapFactory 类来实现。

BitmapFactory 的 decodeStream 方法对裁剪完成或相册选择图片完成后返回的 Uri 进行输入流解析，最终转换成图片显示在相应区域，代码如下：

```
if (resultCode == RESULT_OK) {
                PHOTO.setImageBitmap(crop_bitmap);
                //getPhotoSize();
}
```

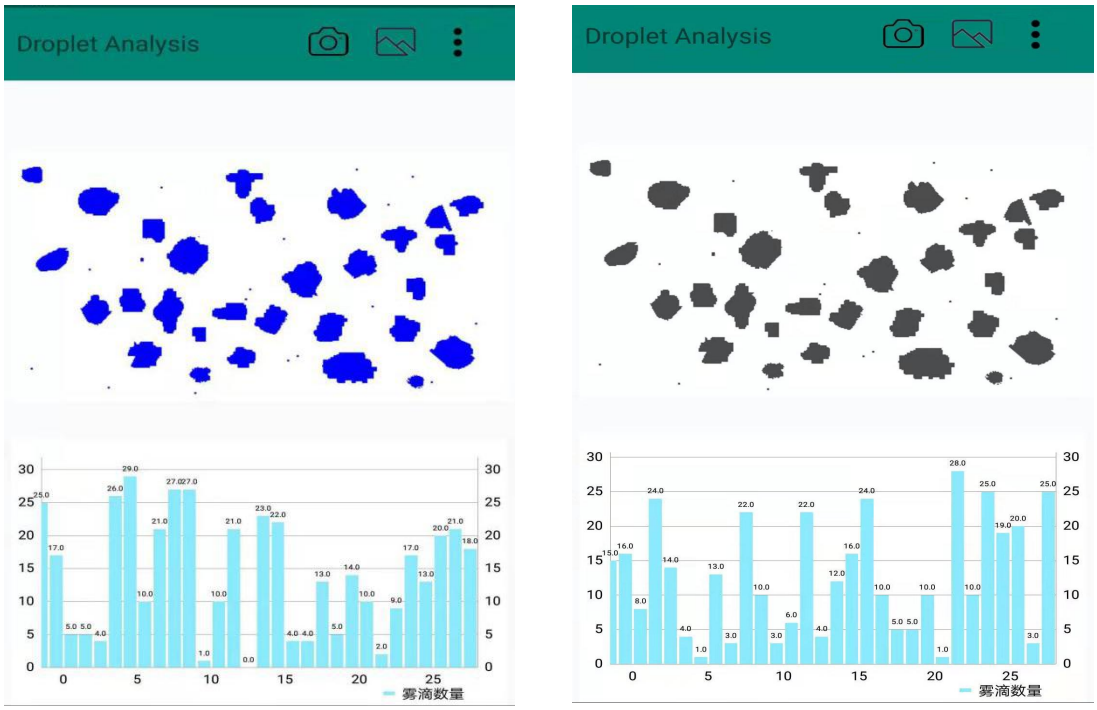
4.2 图像预处理

图像预处理部分包括图像的灰度化、二值化、中值滤波、形态学处理和雾滴的轮廓检测、最小外边框绘制、最小闭圆绘制。

4.2.1 灰度化

灰度化是指将彩色图像中每个像素的 R、G、B 三个分量平均化，使图片呈现出灰色的效果。在二值化前，需先提前将图像进行灰度化处理，并将处理后的灰度图像作为二值化的输入图像。

灰度化前后的效果图如图 15（a）（b）所示。



（a）灰度化前　　（b） 灰度化后

图 15 灰度化前后

在雾滴分析仪中，主要通过 OpenCV4Android 库中提供的 Utils 和 Imgproc 类实现，具体使用了 Utils 的 bitmapToMat 和 matToBitmap 方法以及 Imgproc 的 cvtColor 方法，这里封装为 toGray 方法，代码如下所示：

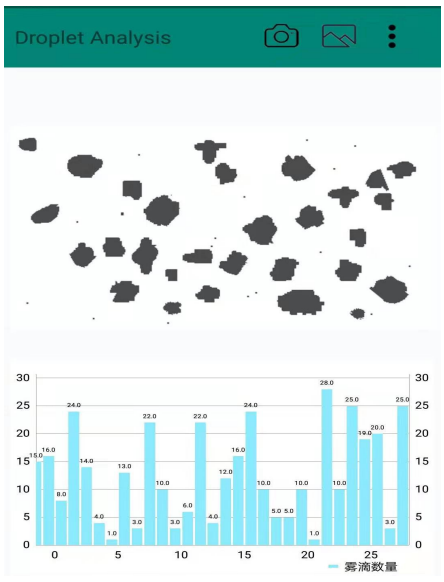
```
public void Grayscale(Mat rgb_Mat , Mat dst_Mat) {
Mat grayMat = new Mat();
 if (PHOTO.getDrawable() == null) {
  //避免闪退问题
 } else {
//灰度化
  Imgproc.cvtColor(rgb_Mat, grayMat, Imgproc.COLOR_RGB2GRAY);
 grayMat.release();
        }
}
```

4.2.2 二值化

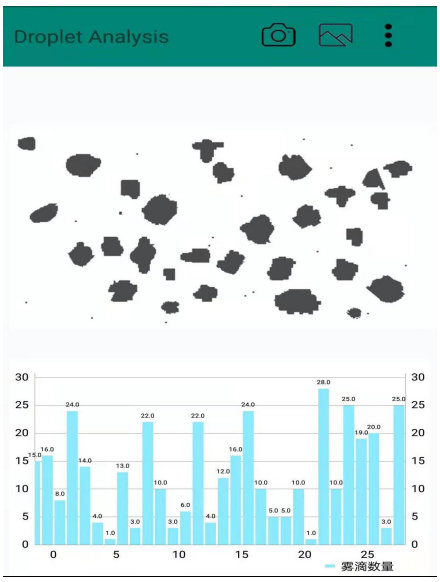
二值化是将图像上的像素点的灰度值设置为 0 或 255，使处理后的图像呈现出明显的只有黑和白的视觉效果。

为了减少光照不均匀带来的负面影响，这里使用了 OpenCV 提供的自适应二值化方法 adaptiveThreshold，能够根据光照的变化算出相应的阈值，并根据计算的阈值对图像进行二值化处理。

二值化前后的效果如图 16（a）（b）所示。



（c）二值化前



（d）二值化后

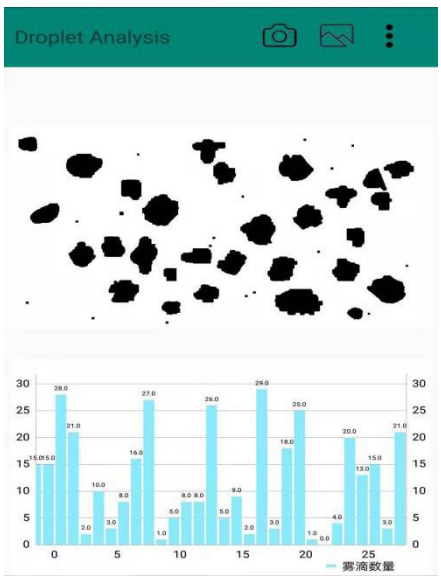
图 16 二值化前后

在雾滴分析仪中，主要通过 OpenCV 库中提供的 Utils 和 Imgproc 类实现，具体使用了 Utils 的 bitmapToMat 和 matToBitmap 方法以及 Imgproc 的 adaptiveThreshold 方法。这里封装为 AdptThreshold 方法，代码如下所示：

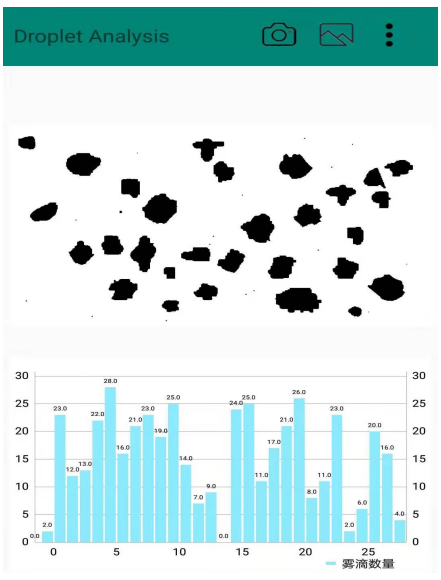
```
public void AdptThreshold(Mat rgb_Mat , Mat dst_Mat) {
      Mat grayMat = new Mat();
       if (PHOTO.getDrawable() == null) {
      //避免闪退问题
      } else {
            //自适应阈值二值化
            Imgproc.adaptiveThreshold(grayMat,threshold_Mat,255,
            Imgproc.ADAPTIVE_THRESH_GAUSSIAN_C, mgproc.THRESH_BINARY, 15, 10);
            grayMat.release();
             }
}
```

4.2.3 形态学处理

经过二值化后的图像能够较为清晰的观察到图像中的黑色雾滴点，但是雾滴之间存在粘连现象，故需对图像进行形态学处理，通过形态学闭运算处理，讲二值化后的雾滴进行“先膨胀后腐蚀”处理，从而减少雾滴粘连现象并尽可能保证雾滴轮廓信息。



（a）形态学处理前



（b）形态学处理后

图 17 形态学处理前后

在雾滴分析仪中，主要通过 OpenCV 库中提供的 Utils 和 Imgproc 类实现，具体

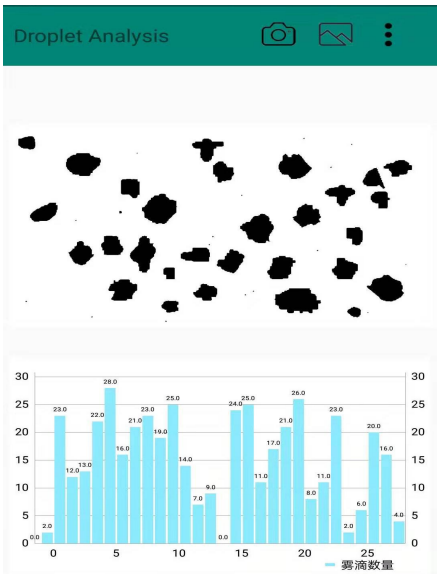
使用了 Utils 的 bitmapToMat 和 matToBitmap 方法以及 Imgproc 的 getStructuringElement 方法，这里封装为 Structuring 方法，代码如下所示：

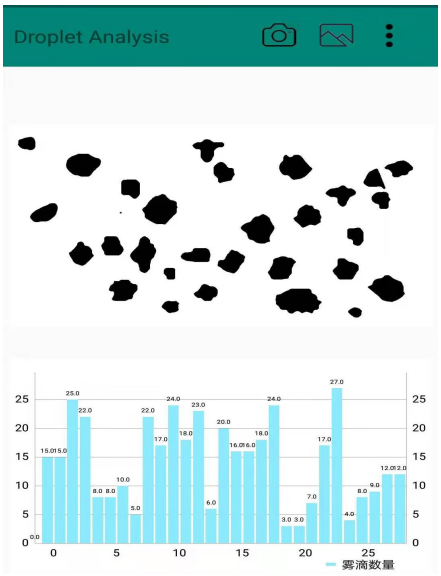
```
private void Structuring(Mat Filter_Mat, Mat dst_Mat) {
Mat kernel; //创建一算子核供给闭运算使用
Mat medianMat = new Mat();
 //形态学闭运算
kernel = Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(3, 3));
Imgproc.morphologyEx(threshold_Mat,Filter_Mat, Imgproc.MORPH_CLOSE,kernel);
 kernel.release();
}
```

4.2.4 中值滤波

中值滤波是把数字图像或数字序列中一点的值用该点的一个邻域中各点值的中值代替，让周围的像素值接近的真实值，从而消除孤立的噪声点。

中值滤波前后的效果图分别如图 18（a）（b）所示。





（c）中值滤波前　　（d）中值滤波后

图 18 中值滤波前后

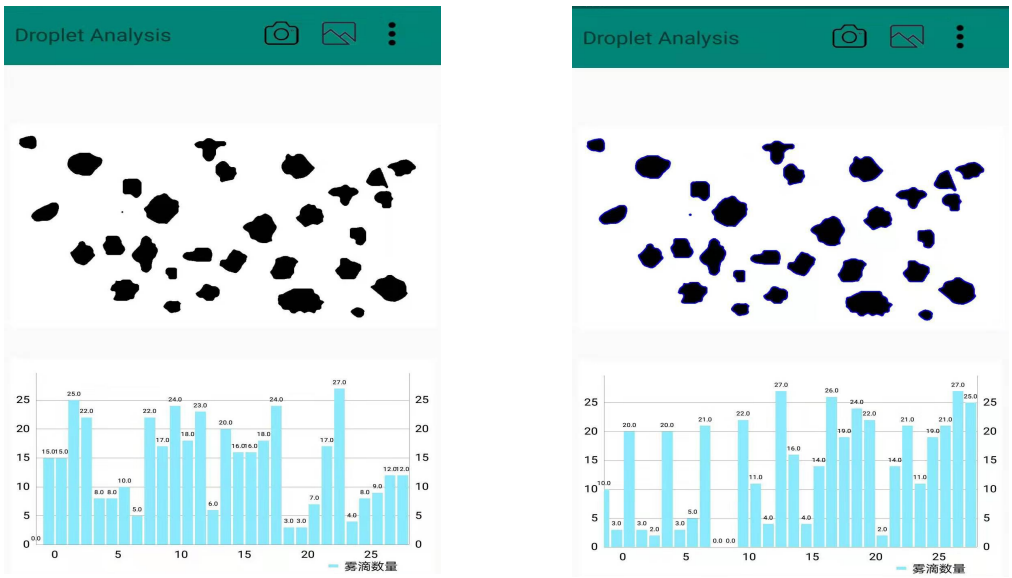
在雾滴分析仪中，主要通过 OpenCV 库中提供的 Utils 和 Imgproc 类实现，具体使用了 Utils 的 bitmapToMat 和 matToBitmap 方法以及 Imgproc 的 mediumBlur 方法，这里封装为 MedianFilter 方法，代码如下所示：

```
private void MedianFilter(Mat threshold_Mat, Mat Filter_Mat) {
Mat medianMat = new Mat();
        //中值滤波
         Imgproc.medianBlur(threshold_Mat, medianMat, 5);
        medianMat.release();
}
```

4.2.5 轮廓检测

经过二值化后的图像能够较为清晰的观察到图像中的黑色雾滴点，接下来需要对图像中的雾滴点进行轮廓检测，通过检测可以获取到雾滴图像中雾滴的轮廓数量以及层级关系，同时使用鲜色画笔绘制出检测到的轮廓可以更直观的观察到轮廓的分布。

轮廓检测前后的效果图分别如图 19（a）（b）所示。



（a）轮廓检测前　　（b）轮廓检测后

图 19 轮廓检测前后

在雾滴分析仪中，主要通过 OpenCV 库中的 Utils 和 Imgproc 类实现，具体使用了 Utils 的 bitmapToMat 和 matToBitmap 方法以及 Imgproc 的 cvtColor 和 findContours 方法，这里封装为 findContours 方法，需要注意的是 findContours 方法返回的是 MatOfPoint 类型的轮廓 contours，每一个 contours 对应图像中的相应的一个雾滴点的轮廓信息，通过遍历 List 列表中的 contours，使用 Imgproc 的

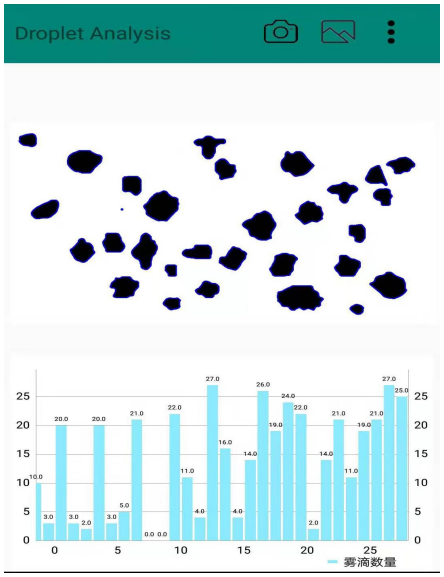
drawContours 方法对每个 contours 进行绘制，从而完成图像中所有雾滴点的轮廓绘制，绘制轮廓使用蓝色线条，代码如下所示：

```
private void findContours(Mat src, Mat dst) {
  Mat gray= new Mat();
  Mat binary = new Mat();
// 轮廓发现
List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
Mat hierarchy = new Mat();
Imgproc.findContours(binary, contours, hierarchy, Imgproc.RETR_EXTERNAL,
Imgproc.CHAIN_APPROX_SIMPLE, new Point(0, 0));
 // 绘制轮廓
dst.create(src.size(), src.type());
for(int i=0; i<contours.size(); i++) {
        Imgproc.drawContours(dst, contours, i, new Scalar(0, 0, 255), 2);
}
// 释放内存
 gray.release();
 binary.release();
}
```

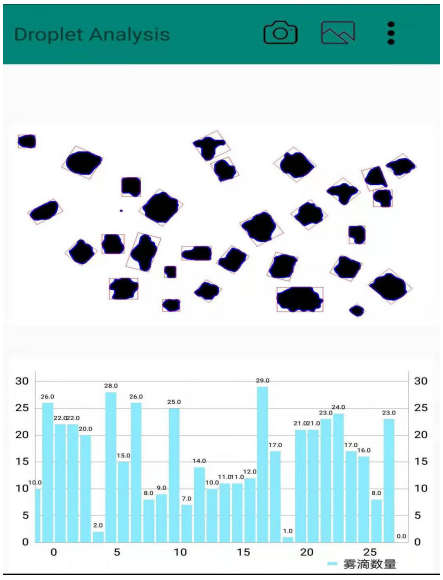
4.2.5 最小外边框绘制

轮廓检测完成后可以看到雾滴图像中有许多密密麻麻的红色填充雾滴点，为了更清晰的观察每个雾滴点所占的区域面积，也为了后面对雾滴粒径以及密度的计算，接下来对轮廓检测后雾滴图像上的雾滴轮廓进行最小外边框的绘制。

最小外边框绘制效果前后如图 20（a）（b）所示，最小闭圆绘制效果前后如图 21（a）（b）所示。

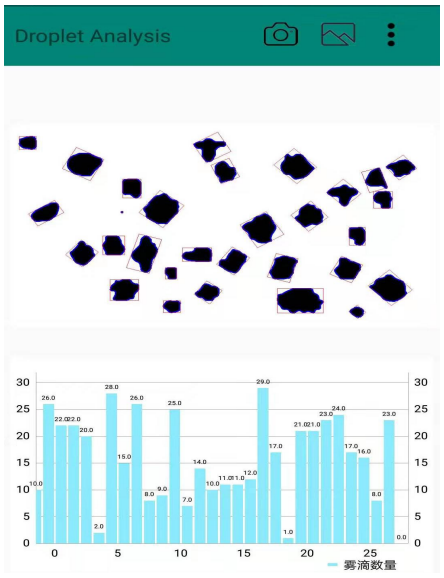


（a）最小外边框绘制前

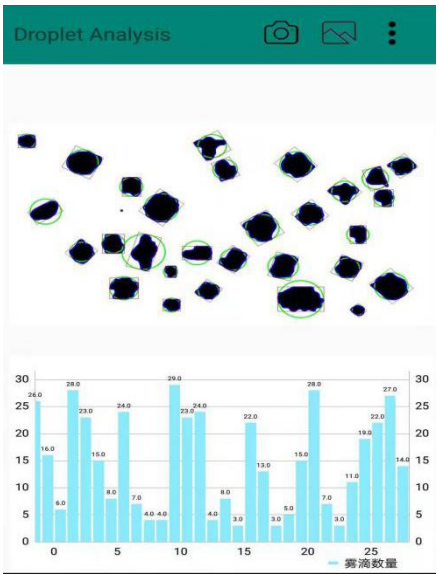


（b）最小外边框绘制后

图 20　最小外边框绘制前后



（c）最小闭圆绘制前



（d）最小闭圆绘制后

图 21 最小外边框绘制前后

在雾滴分析仪中，主要通过 OpenCV 库中的 Utils 和 Imgproc 类实现，具体使用了 Utils 的 bitmapToMat 和 matToBitmap 方法以及 Imgproc 的 minAreaRect、circle 和 line 方法，这里封装为 searchMinArea 方法，代码如下所示：

```
 private void SearchMinArea(Mat src,Mat Dst) {
Mat gray= new Mat();
// 轮廓发现
List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
Mat hierarchy = new Mat();
Imgproc.findContours(Dst, contours, hierarchy, Imgproc.RETR_EXTERNAL,
Imgproc.CHAIN_APPROX_SIMPLE, new Point(0, 0));
// 测量轮廓
 for(int i=0; i<contours.size(); i++) {
 //定义一个矩形的类使用
 Rect rect = Imgproc.boundingRect(contours.get(i));
double w = rect.width;
double h = rect.height;
double tl= rect.x;
double br= rect.y;
//【优选办法】通过调用 Rect 子函数.br()和.tl()分别获得宽高
Imgproc.rectangle(Dst,rect.br(), rect.tl(), new Scalar(255, 0, 0),1);
//定义一个带角度矩形的类使用
RotatedRectminRect=Imgproc.minAreaRect(new
MatOfPoint2f(contours.get(i).toArray()));
double w2 = minRect.size.width;
double h2 = minRect.size.height;
double tl2= minRect.center.x;
double br2= minRect.center.y;
//定义一个数组，供给最小外边框的角点
 Point[] minPoint = new Point[4];
minRect.points(minPoint);
//此处不能使用原生函数.rectangle，需要使用绘制直线的办法
for (int j = 0; j < 4; j++){
Imgproc.line(Dst, minPoint[j], minPoint[(j + 1) % 4], new Scalar(0, 255, 0), 1);
 }
}
```

通过最小外边框检测以及绘制可以检测出每个雾滴轮廓的最小外边距的宽高，可用于对雾滴密度的计算，同时对最小外边框的绘制更直观的观察每个轮廓的大小以及所占区域。

4.3 分析报告 excel 表生成

图像经过一键分析后，能够获取雾滴的当量直径，雾滴最小包围圆的直径，雾滴的面积以及变异系数。所得这些数据为了方便试验后分析，故需将之导出到 excel。

在雾滴分析仪中需要用到导出 excel 数据的库，需添加依赖 implementation group: 'net.sourceforge.jexcelapi', name: 'jxl', version: '2.6.12'，然后编辑导出数据的格式，打包成一个类，再在主函数中使用该类，先指定保存路径以及需要保存文件的名字，然后创建每列数据的标题，通过 for 循环以及全局变量数组里

的数据，按照雾滴直径大小顺序将数据一个个填写至表格对应的位置，最后保存到指定路径/sdcard/Droplet。

导出表格效果如下图。

雾滴分析

	A	B	C
1	序号	雾滴直径	数量
2	1	0~3	2.0
3	2	3~6	0.0
4	3	6~9	0.0
5	4	9~12	0.0
6	5	12~15	4.0
7	6	15~18	5.0
8	7	18~21	19.0
9	8	21~24	12.0
10	9	24~27	4.0
11	10	27~30	10.0
12		总计	56.0
13		覆盖率	0.0
14		密度	0.0
15		变异系数	0.0

图 22　数据分析图

使用方法：

在项目中引入库：

```
group: 'net.sourceforge.jexcelapi', name: 'jxl', version: '2.6.12'
```

该库用来读写 excel 文件。编辑导入 excel 格式并打包成 ExcelUtil 类方便调用。

4.4 雾滴分析

雾滴分析功能主要包含轮廓数、轮廓面积和周长、最小外边框宽高的计算。

4.3.1 轮廓数

轮廓数主要由 4.2.4 中轮廓检测得到的轮廓 contours 计算得到，因为 contours 中记录着雾滴图像中分析得到的所有雾滴轮廓的信息，所以 contours 列表中的数量就是图像中的轮廓总数。可以直接通过 List 类型的 size 方法获取到 contours 的数量，

也就是轮廓总数。

轮廓总数显示如图 23 所示。

雾滴分析

序号	雾滴直径(mm)	数量(个)
1	2.336448580	1.0
2	3.271028012	1.0
3	5.286780580	1.0
4	3.414378332	1.0
5	2.336448580	1.0
6	5.437487860	1.0
7	1.401869148	1.0
8	5.433436081	1.0
9	3.426791250	1.0
10	5.295950114	1.0
11	5.286779392	1.0
12	5.295950114	1.0
13	2.336448580	1.0
14	1.246105909	1.0
15	1.401869148	1.0
16	3.426791250	1.0
17	1.401869148	1.0
18	1.246105909	1.0
19	5.295950114	1.0
20	5.295950114	1.0
21	2.336448580	1.0
22	2.180685341	1.0
23	1.321694848	1.0
24	5.295950114	1.0
25	3.304237565	1.0
26	5.295950114	1.0
27	3.271028012	1.0
28	1.401869148	1.0
29	5.140186876	1.0
	总计(个)	29.0
	覆盖率(%)	5.57
	平均粒径(mm)	3.43
	变异系数	26.26

图 23 轮廓总数

在雾滴分析仪中，主要通过 List 类型的 contours 实现，代码如下所示：

```
int circlenum=contours.size();
```

其中变量 contoursNum 表示雾滴总数量。

4.3.2 最小外边框的宽高

雾滴最小外边距宽高主要由 3.2.4 中轮廓检测得到的轮廓 contours 计算得到。通过 for 循环对 contours 列表进行遍历，并在遍历中根据雾滴图像中每个雾滴轮廓对应的 contour 获取到最小外边框的四个顶点，根据这四个顶点的大小及位置关系求出相应的宽高。

在雾滴分析仪中，主要通过 List 类型的 contours 以及 Imgproc 类实现，具体使用了 Imgproc 的 minAreaRect 方法，代码如下所示：

```
//获得宽度
double w2 = minRect.size.width;
//获得长度
double h2 = minRect.size.height;
```

5 雾滴统计分析

5.1 雾滴数量分析

雾滴数量是雾滴分析前应该获取的一项基础指标，为雾滴粒径、沉积分布等分析提供数据支撑。

由 4.3.1 雾滴分析中求得的轮廓数，就是雾滴图像中的雾滴总数量。

5.2 雾滴粒径分析

农药使用过程中，经过农药喷雾器喷头的雾滴会受到雾化部件的影响，喷洒出的雾滴并不是均匀分布的。而雾滴粒径的大小，对喷雾效果起着重要的作用。粒径越大的雾滴，拥有更大的质量与动能，受外界的影响较小，不容易产生位置偏移，比较稳定。但相对而言，缺点也比较明显，附着性能差，喷雾效果不佳，而且对环境污染较大。而粒径越小的雾滴，在作物表面沉降和覆盖的效果更佳，而且附着性能好，不易流失，有效提高农药的利用率。因此，在实际应用中，对雾滴粒径的分析，并选用大小合适的雾滴粒径，对农药喷洒效果起着至关重要的作用。

雾滴粒径的表示方法目前有四种，其中比较常用的是体积中值粒径和数量中值粒径两种方法。

体积中值粒径（Volume median diameter，VMD）:在一次喷雾中，将全部雾滴的体积从小到大顺序累加，当累加值等于全部雾滴体积的 50%时，所对应的雾滴粒径为体积中值粒径，简称体积中径[14]。

数量中值粒径（Number median diameter，NMD）:在一次喷雾中，将全部雾滴从小到大顺序累加，当累加的雾滴数目为雾滴总数的 50%时，所对应的雾滴粒径为数量中值粒径，简称数量中径[14]。

体积中值粒径与数量中值粒径的关系如图 24 所示。

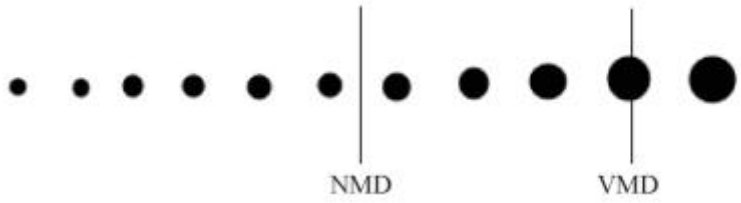


图 24 雾滴的数量中值粒径（NMD）和体积中值粒径（VMD）示意图

根据体积中值粒径的定义，需先算出雾滴总体积，再对雾滴体积按从小到大的顺序进行累加，当累加的体积为总体积的一半，即与总体积的比为 1/2 时，取出所对应的雾滴粒径即为体积中径。

对于数量中值粒径的计算，相对比较方便，先对图像中的雾滴面积按照从小到大的顺序进行排序，然后取出最中间的雾滴粒径即为数量中径。

5.3 雾滴沉积分布分析

在靶标作物上的喷洒的农药雾滴会附着在作物表面上，形成沉积效果。经过喷雾系统喷洒的雾滴沉积分布具有较大的不确定性，而且容易受到硬件以及环境如风向等影响。

雾滴的不同沉积效果会影响到喷雾的成效，是评价农田作业效果的一项重要指标。其主要的评价指标包括雾滴覆盖率、密度以及分布均匀性等。

5.3.1 雾滴覆盖率

雾滴覆盖率用雾滴覆盖区域面积占统计总面积的百分比表示，可用于分析雾滴的整体喷洒效果，计算方法如公式（2）所示。

$$C = \frac{A_s}{A_p} * \ 100 \qquad (2)$$

式中 C 为喷雾覆盖率，%；A_s为雾滴总面积；A_p为试纸总面积[16]。

其中参数中的 dropletArea 表示所有雾滴总面积，paperArea 表示试纸裁剪部分的总面积，均以手机屏幕像素点为单位。

5.3.2 雾滴分布均匀性

雾滴分布均匀性是指雾滴在靶标作物上的分布是否均匀，主要使用雾滴变异系数来衡量雾滴的分布均匀性。变异系数的计算方法如如公式（4）、（5）所示。

$$CV = \frac{S}{X} * 100\% \quad (4)$$

$$S = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n-1}} \quad (5)$$

式中：S——标准差；X_i——每个采样卡上雾滴沉积参数（如覆盖率、沉积密度、当量直径等）；Ẋ——所有采样卡上雾滴沉积参数的平均值（如覆盖率、沉积密度、当量直径等）；n——每层采样卡总数[17]。

6 验证试验

验证试验包含雾滴覆盖率测试和分析仪测试试验两部分。雾滴覆盖率测试部分主要通过营造模拟环境，创建已知的模拟量进行测试，将测试的结果与已知的模拟量进行对比，并两者对比求得的准确率来分析测试结果是否准确。分析仪测试试验部分主要根据整体流程设计框图使用雾滴喷施分布分析仪 App 对一张水敏纸进行雾滴分析，并将分析结果显示在 App 中。

6.1 雾滴覆盖率测试

本次验证实验主要检验雾滴喷施分布分析仪 App 的雾滴覆盖率的准确度。自制模拟的雾滴分布图，使用工具模拟已知的雾滴大小，根据已知模拟量计算出精确的雾滴覆盖率，并与雾滴分析仪 App 分析的相应结果进行对比，以验证雾滴分析仪 App 分析的准确性。

为了营造可靠的模拟环境，作者通过 word 图片编辑器绘制了一张 10*5cm 大小的雾滴分布模拟图，如图 25 所示。在模拟图中的不同位置绘制了四种不同半径的圆以及一个椭圆来模拟雾滴的分布，从上到下，雾滴直径依次为：1、2、3、5，以及长轴为 5，短轴为 2.5 单位均为 mm。

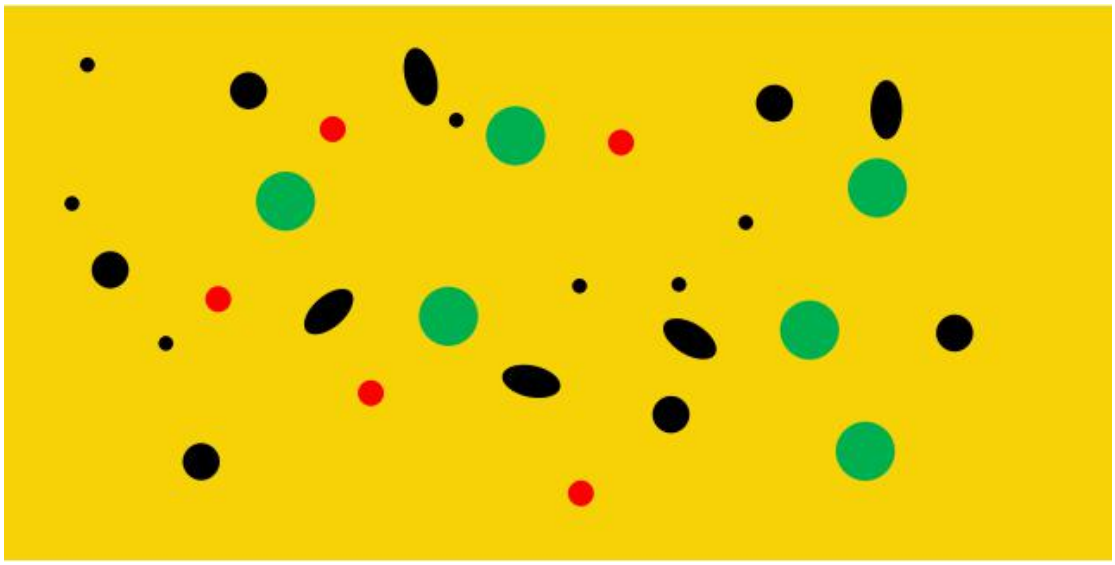


图 25　雾滴分布模拟图

由已知的模拟量中的模拟图尺寸、雾滴数量以及雾滴半径可以算出试纸区域面积、雾滴总面积，进而求得模拟图的雾滴覆盖率。根据 5.3.1 雾滴覆盖率中公式（2）计算雾滴覆盖率，计算过程如下所示。

$$C=\frac{A_S}{A_P}*100=\frac{\pi(7*0.5^2+5*1^2+6*1.5^2+6*2.5^2+5*2.5*1.25)}{100*50}*100\%\approx 6.57\%$$

根据已知模拟量求出雾滴分布模拟图的雾滴覆盖率为 18.44%。

接下来使用设计的低量雾滴分析仪 App 对雾滴分布模拟图进行拍照截图采集并进行雾滴分析，经过二值化、中值滤波、轮廓检测和最小外边框绘制等预处理后，求出的分析结果如图 26 所示。

平均误差(%)	23.08
变异系数	26.26
覆盖率(%)	12.12
平均粒径(mm)	4.56
总计(个)	29

图 26　分析仪 App 对雾滴分析模拟图的分析结果

根据分析的结果所展示的数据，分析仪 App 对模拟图分析得出的雾滴覆盖率约等

于 0.11619，即 11.62%。与根据模拟量精确计算出来的雾滴覆盖率 13.14%相比，准确率为 88.43%，同时分析仪 App 检测出的六个雾滴的圆形度依次为：0.989、1.028、1.007、1.003、1.003、0.997，与标准圆形度为 1 的圆形相比十分接近。由此可见经过分析处理后的六个雾滴基本都为标准的圆形，预处理对雾滴的影响较小。

6.2 分析仪测试试验

首先准备一张低量雾滴分布的图片作为测试样本，如图 27 所示。

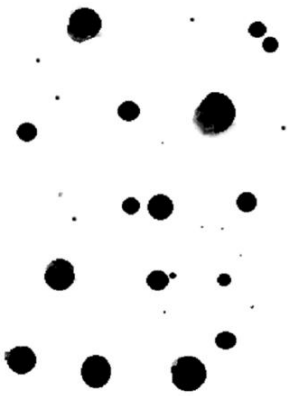


图 27　低量雾滴分布测试图

将该测试图载入到雾滴分析仪 App 中，效果如图 28 所示。

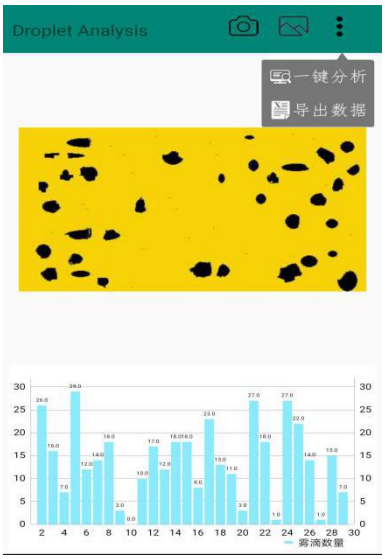


图 28　载入测试图后的 App 界面

按照 3.2 整体流程设计框图对测试图依次执行二值化、中值滤波、轮廓检测、最小外边框绘制等预处理，处理结果如图 29 所示。

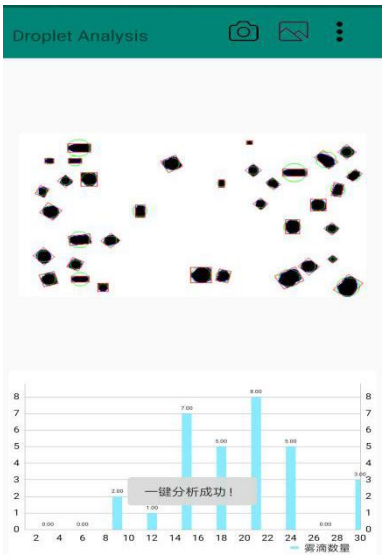


图 29　预处理结果

对预处理后的低量雾滴分布测试图进行数据分析，可获得每个雾滴的轮廓面积、周长、长轴、短轴和圆形度，分析结果如表 2 所示。

表 2 雾滴分析结果

实际粒径大小(mm)	雾滴直径(mm)	误差(%)
1	1.2461059093475342	24.61059093
1	1.2461059093475342	24.61059093
1	1.321694848107711	32.16948481
1	1.401869148015976	40.1869148
1	1.401869148015976	40.1869148
1	1.401869148015976	40.1869148
1	1.401869148015976	40.1869148
2	2.180685341358185	9.034267068
2	2.3364485800266266	16.822429
2	2.3364485800266266	16.822429
2	2.3364485800266266	16.822429
2	2.3364485800266266	16.822429
3	3.271028012037277	9.034267068
3	3.271028012037277	9.034267068
3	3.304237565911478	10.1412522
3	3.414378332853687	13.8126111
3	3.426791250705719	14.22637502
3	3.426791250705719	14.22637502
5	5.1401868760585785	2.803737521
5	5.286779392430844	5.735587849
5	5.286780580810046	5.735611616
5	5.29595011472702	5.919002295
5	5.29595011472702	5.919002295
5	5.29595011472702	5.919002295
5	5.29595011472702	5.919002295
5	5.29595011472702	5.919002295
5	5.29595011472702	5.919002295
5	5.433436081279979	8.668721626
5	5.437487860166698	8.749757203

根据上表中获取到的每个雾滴相关数据进行分析，计算求得雾滴分布测试图中的雾滴总面积，雾滴总数量、雾滴覆盖率以及雾滴密度，测试结果如图 30 所示。

平均误差(%)	15.729134
变异系数	26.26
覆盖率(%)	5.57
平均粒径(mm)	3.43
总计(个)	29.0

图 30　雾滴测试图最终分析结果

其中雾滴总面积是在手机图片显示区域部分检测到的所有轮廓面积总和，试纸区域面积为经过 Android 手机屏幕缩放并渲染完成，显示在屏幕上实际显示的宽高乘积，两者均以手机的像素点 px 为单位。而雾滴覆盖率为面积与面积的比例，与像素无关。

7 结论与探讨

7.1 结论

通过雾滴分析仪的验证与分析，总结了基于 Android 低量雾滴分析仪的结论如下所示。

（1）本文设计的雾滴分析仪基于移动端的 Android 系统开发，并使用 Android 手机系统自带的拍照、截图、相册图片选择等图像采集功能来代替人工采集图像方式，充分发挥了 Android 移动端设备的便捷性、实时性等优势。

（2）该雾滴分析仪不仅支持试纸部分的雾滴分析，同时也支持试纸局部区域的雾滴分析。使用者可以根据分析要求对试纸中需要的区域使用截取功能，并对截取的区域进行雾滴分析。

（3）通过模拟验证试验，雾滴分析仪 App 在图像预处理阶段中二值化以及中值滤波对原图像的雾滴的影响较小，在数据分析阶段中对雾滴覆盖率检测的准确性较高，与模拟量计算出的精确雾滴覆盖率相比准确率在 85%左右。根据实际试验结果显示，雾滴分析仪 App 能满足基本的低量雾滴分析。

7.2 探讨

虽然基于 Android 的低量雾滴分析仪节省了人工系统采集部分所需要的人力、时间和精力，降低了部分人为误差。但目前仍存在很多不足，主要有以下几点：

（1）不同厂商的 Android 手机系统相机拍摄效果无法保证。

由于 Android 的开源性，许多手机厂商如华为、小米、oppo、魅族等都根据开源协议定制了自己的 Android 系统，不同的定制系统有不同的系统定制相机。而相机拍摄的效果也不尽相同，这就导致拍摄出的效果无法保证。是否对光照敏感，能否近距离对焦，拍摄图片是否压缩等，这些问题无法统一，因此不能保证拍摄的作物照片与实际作物的误差。

（2）雾滴黏连的问题。

粘连雾滴区域会影响对雾滴图像中雾滴轮廓数的检测，从而影响到雾滴个数的统

计，目前雾滴喷施分布分析仪 App 还只能处理低量且没有相互黏连的雾滴图像，否则图像测量分析结果会出现较大的误差。

除了上述提到的问题之外，还需要考虑实际的情况，对水敏纸采集的雾滴分布图像部分进行容错处理，考虑光线较暗等环境影响因素。目前来说，基于 Android 的雾滴喷施分布分析系统尚未成熟，所以暂时定位在低量雾滴的处理上。

综上所述，本文所设计的雾滴喷施分布分析仪 App 暂时出于测试研发阶段，还不能完全满足实际生产的需求。为了能在实际情况下使用 Android 手机实时、快速、准确地检测，还需要对图像处理算法进行改进以适应环境的干扰，同时做好 App 手机兼容性，以及版本的更新和迭代。

8 对本门课程的建议

对于这学期的数字图像处理课授课，感觉老师的授课方式挺适合我们本科生的，能够将理论和实践相结合，通过演示程序来吸引我们的眼球，使我们能够更专注于去听这门课，并能够与学生互动，调动学生的积极性以及自主思考能力，使每个人都尽可能的积极融入课堂。其中授课过程中有些较为简单的知识讲的过为详细，如中值滤波以及均值滤波，建议老师可以将后面更为难理解的部分讲的详细一点。

参 考 文 献

[1]OpenCV Android 开发实战. 贾志刚著.北京机械工业出版社.2018.06.

[2]Java 数字图像处理：编程技巧与应用实践.贾志刚著. 机械工业出版社. 2015. 11.